Horizon3.ai User Documentation

None

Horizon3.ai

None

Table of contents

1. Welcome to Horizon3.ai Documentation

NodeZero provides continuous autonomous penetration testing as a true SaaS offering. With NodeZero, cybersecurity teams proactively find and fix internal and external attack vectors before attackers can exploit them.

Let's get started!

1.1 Download this documentation for offline use

- PDF

2. Getting Started

2.1 Introduction

This guide is designed for people just starting out with NodeZero. It will help you get the most out of the platform by walking you through four basic steps. If you're already an experienced user, check out the Reference section!

1. Registering an account on our user portal
2. Run an External Pentest to assess your internet-facing infrastructure
3. Set up a NodeZero Host that acts as a starting point for an internal pentest
4. Run an Internal Pentest to see your network from the eyes of an attacker

2.2 Register an Account

2.2.1 Register An Account

Follow the steps below to register an account with NodeZero.

How to Register A New Account

> **Note**
>
> New accounts are granted read-only access. To upgrade to free trial, refer to Upgrade your account to free trial.

1. Go to the NodeZero portal and click Sign Up
2. Select Google, LinkedIn, or Microsoft to register your account
3. Read and agree to the terms and conditions

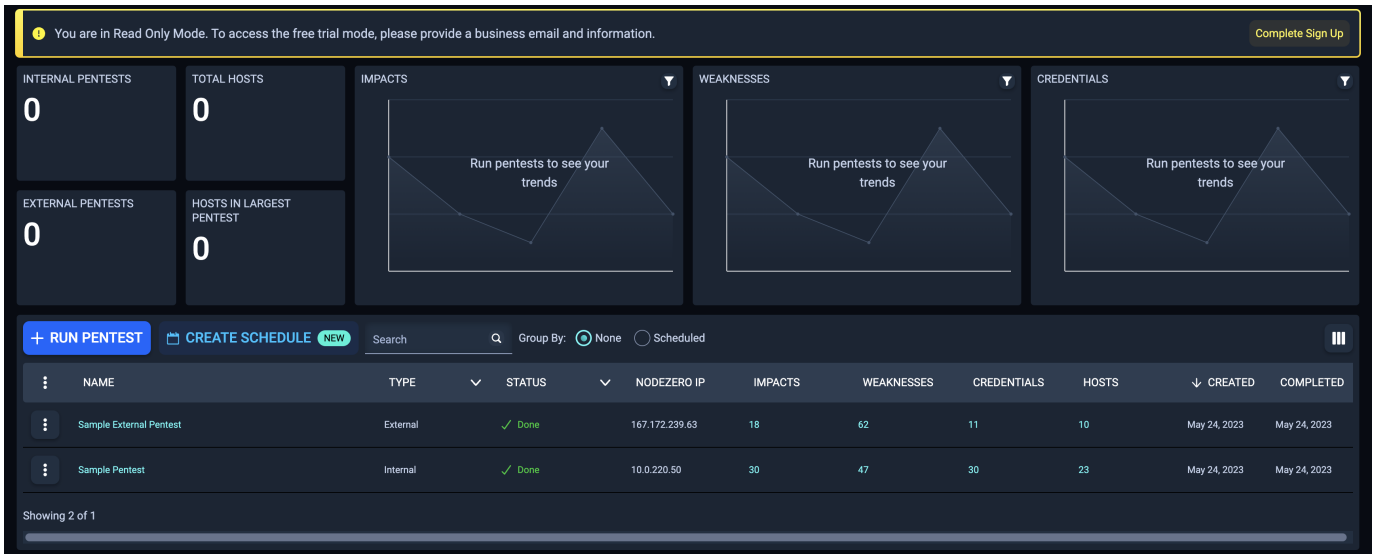
Upgrade Your Account to Free Trial

You can upgrade your account to a 30 day free trial at any time. To upgrade, you must fill out the company information form and provide a valid company email.

> **Warning**
>
> A contact list, also known as a distribution group (i.e. admin@.com) is not permitted.

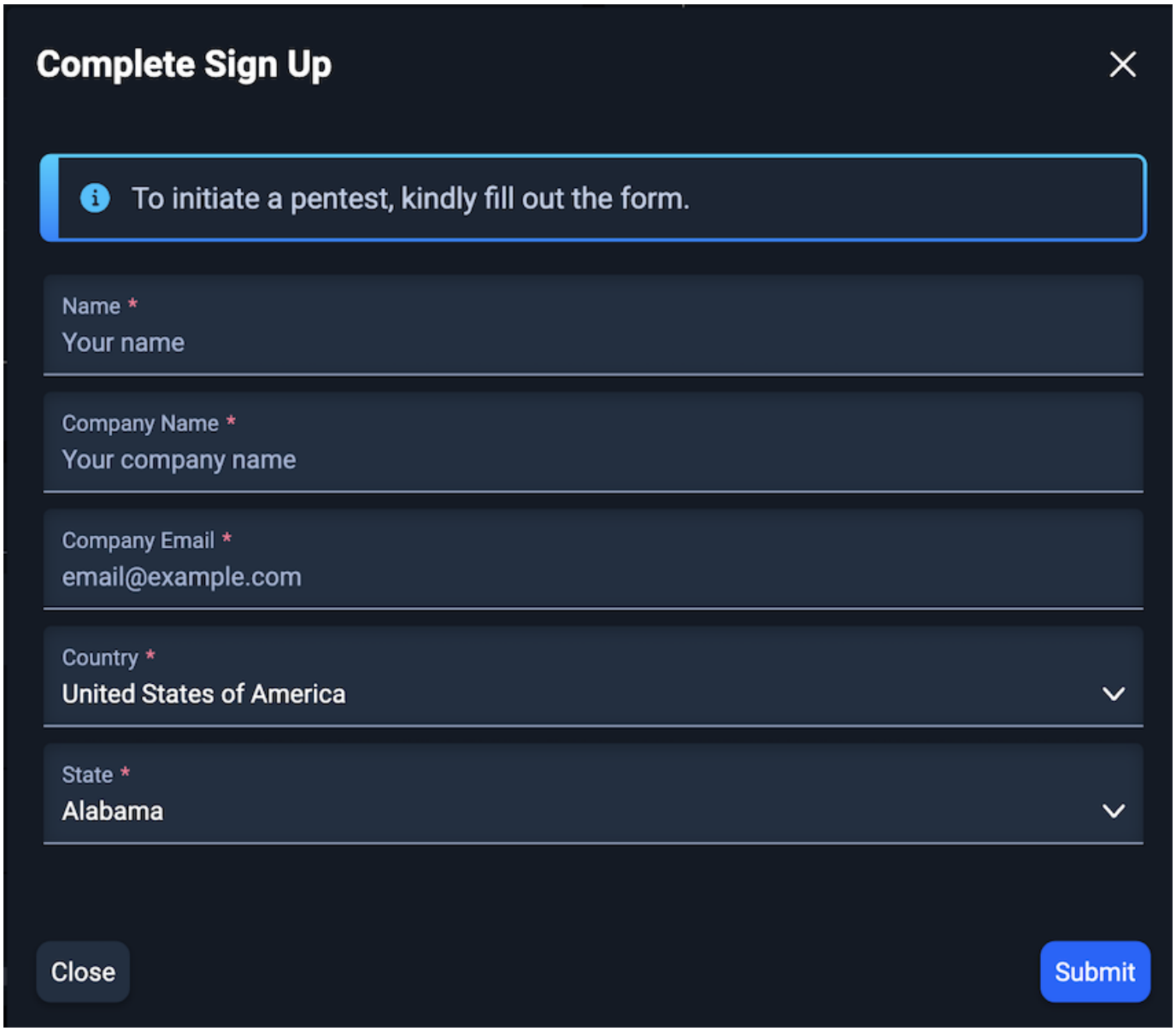
1. WHERE TO FIND THE FORM

Locate and click the "Complete Sign Up" button at the top right of the page.



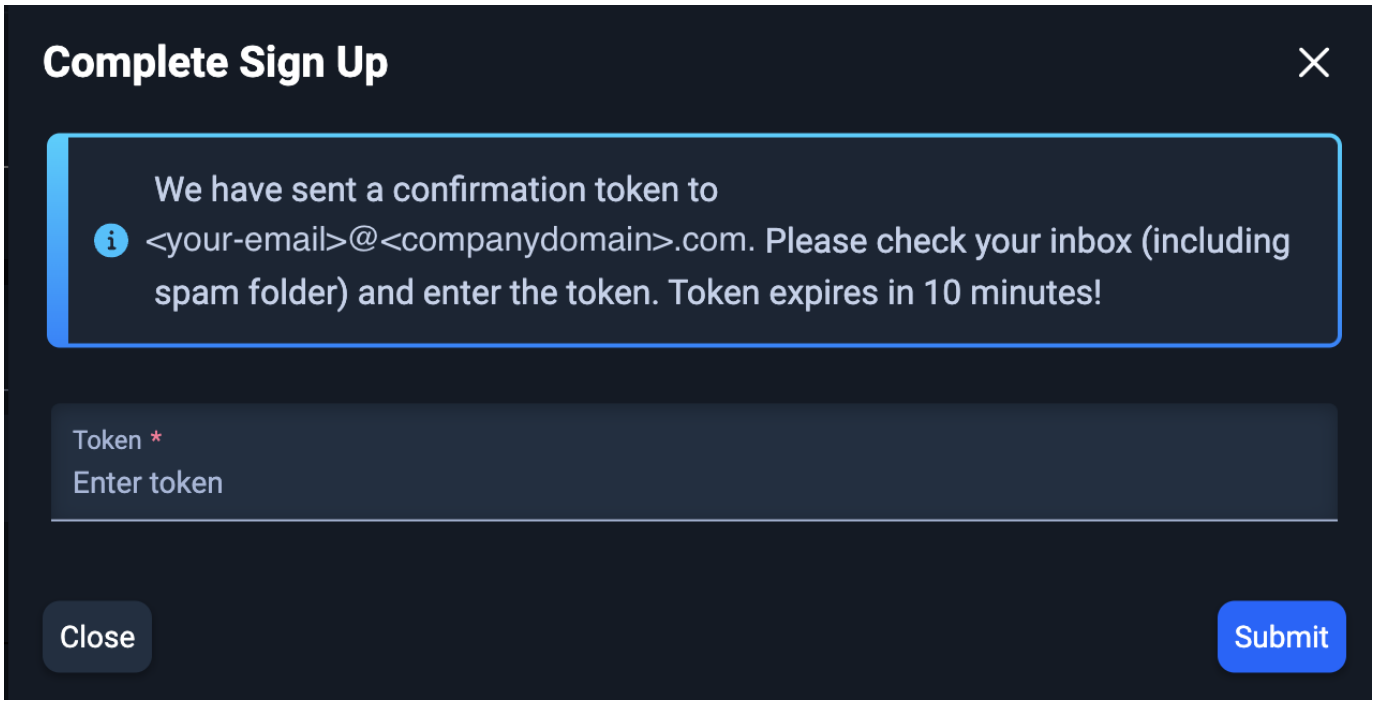
2. SUBMITTING THE FORM

Fill out the required fields. The email must be a valid company email address **(again, distribution lists emails are not permitted)**. If you do not provide a company email, you will receive an error and not be granted a free trial. The email can be the same as the one used to create the account, if a company email was used.



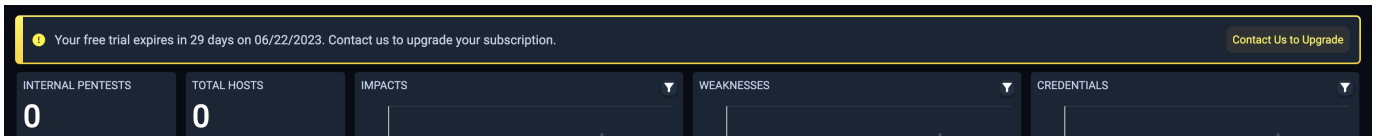
3. SUBMIT VERIFICATION TOKEN

A validation token will be sent to the email provided in the form from NodeZero. Ensure to check your spam folder.



4. CONFIRM ACCOUNT ACCESS LEVEL

Once you have verified your account, you will be granted a free trial. A banner will be displayed stating your account is now a free trial and the time left in the trial. Once your 30 day free trial is over, you will be placed back into read-only mode. You can only verify a company email for a free trial once.



Note

If you have any questions, reach out to us by clicking the "?" icon in the top right and selecting "Chat with support".

2.3 Run an External Pentest

2.3.1 Run an External Pentest

NodeZero External Pentest is an easy way to gain an additional perspective on your environment. Due to the nature of an external pentest, you must first authorize the discovered assets.

Before getting started, navigate to the NodeZero portal and login with your credentials.

How to Run an External Pentest

To run an external pentest, first create an Asset Group and run Asset Discovery to enumerate external facing assets. When the Asset Discovery completes, you can authorize assets for pentesting and start an external pentest. Follow the documentation step by step or jump to one of the following sections:

- Create an Asset Group
- Run Asset Discovery
- Authorize Assets
- Run an External Pentest

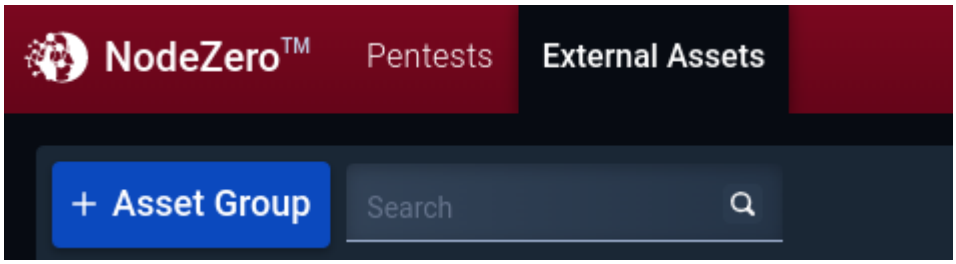
CREATE AN ASSET GROUP

1. Navigate to External Assets

To run an external pentest, first create an Asset Group. The Asset Group is used to scope the external pentest.

2. Click Create Asset Group

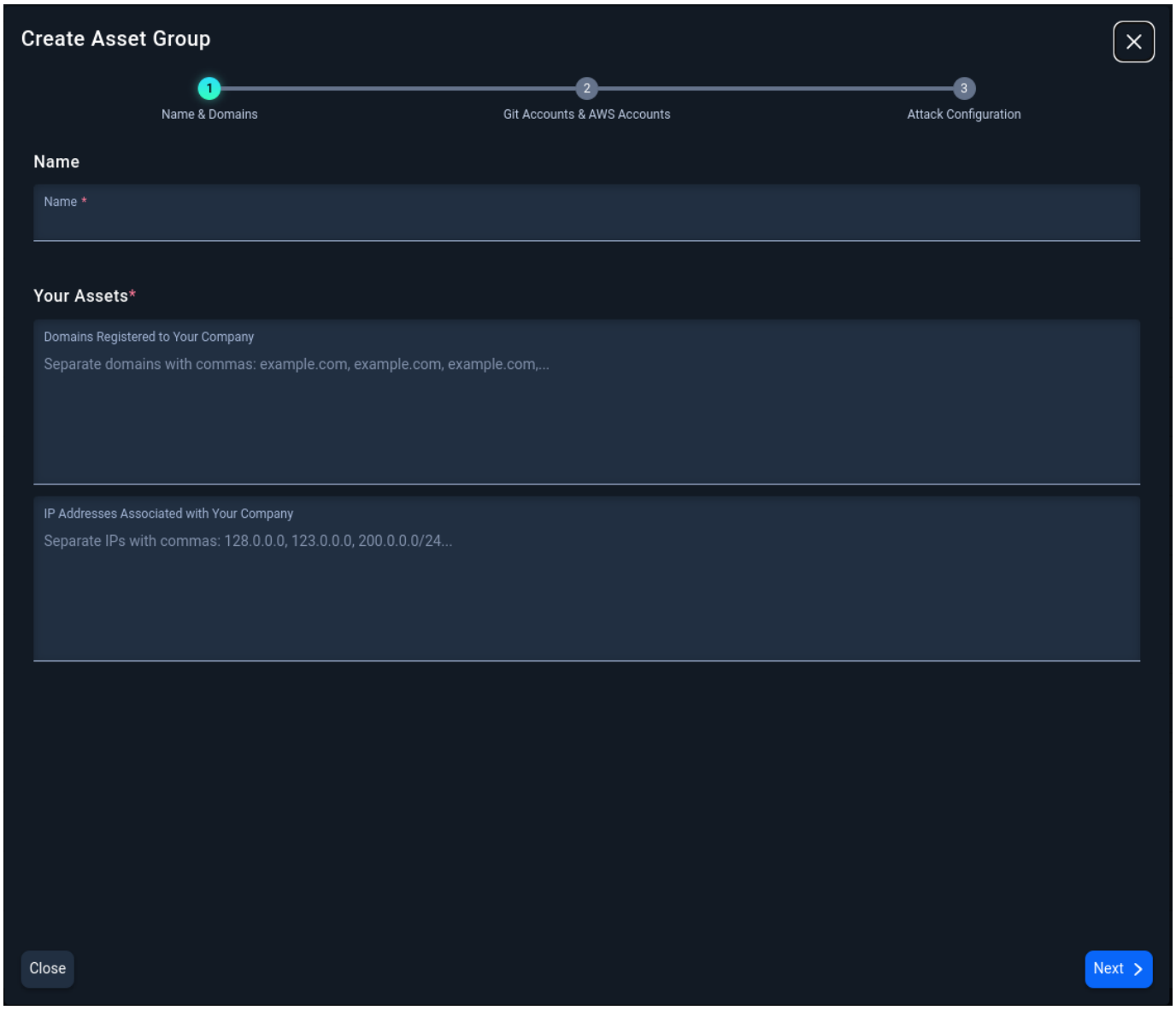
On External Assets, click + Asset Group to open the asset group configuration.



3. Configure the Asset Group

3.1 Set a Scope for the Asset Group

Name the Asset Group and provide company assets as domains, IP addresses, or a combination of domains and IP Addresses. A maximum of 700 domains can be added to the configuration. IP addresses need to be Public IPs only with a maximum input of 3000 entries and formatted using IP/CIDR notation. Please specify a network segment of /20 or smaller (e.g., /22, /23, /24). Both domains and IP addresses should be comma separated. Click Next.

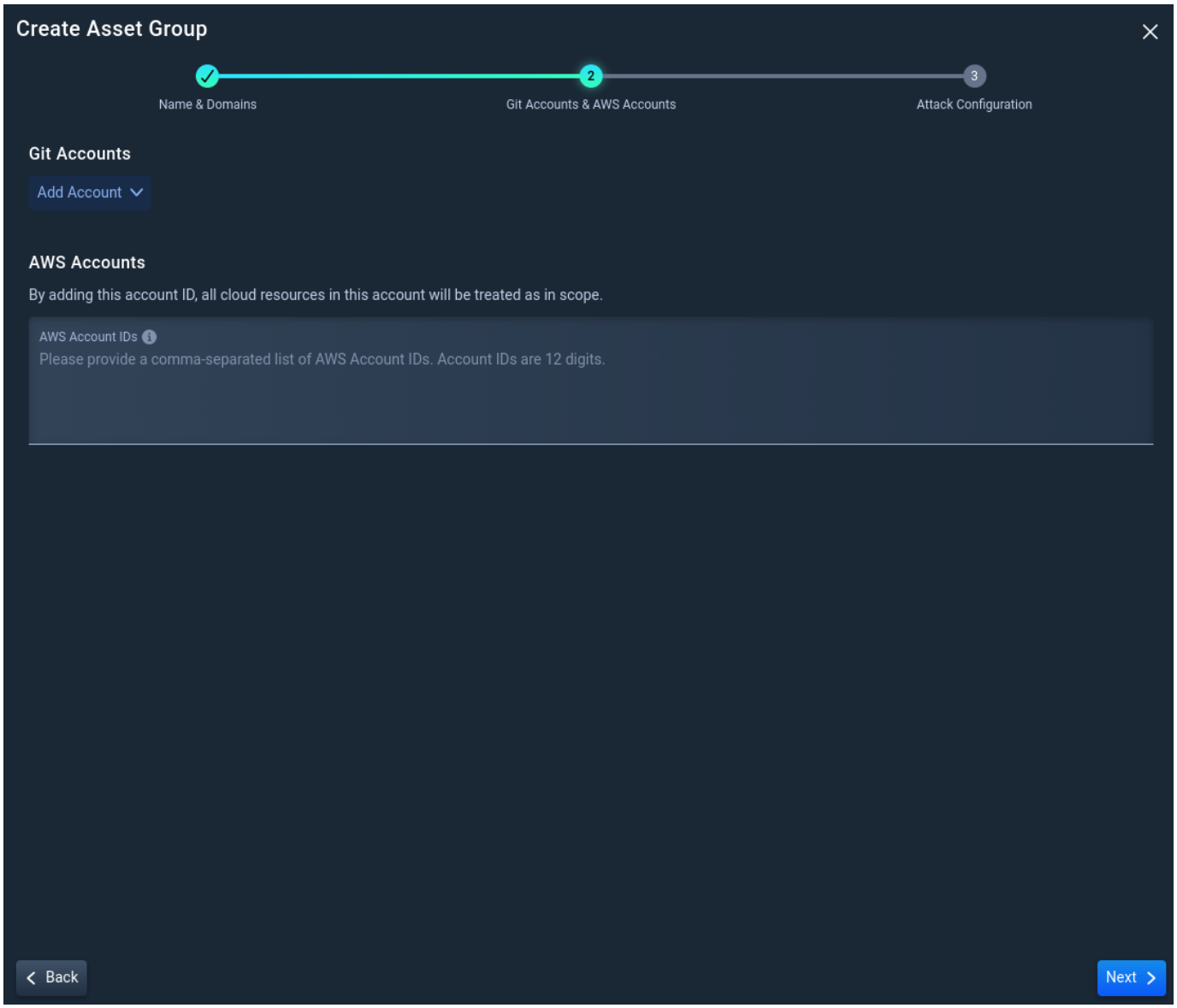


3.2 Optionally, add your Git and AWS Accounts

Listing Git and AWS accounts allows NodeZero to confirm ownership of these accounts and perform a more thorough enumeration of assets.

- To add a Git Account, select +Add Account, select a Git provider, and add the account name.
- To add an AWS Account, type the 12-digit AWS Account ID in the box listed below.

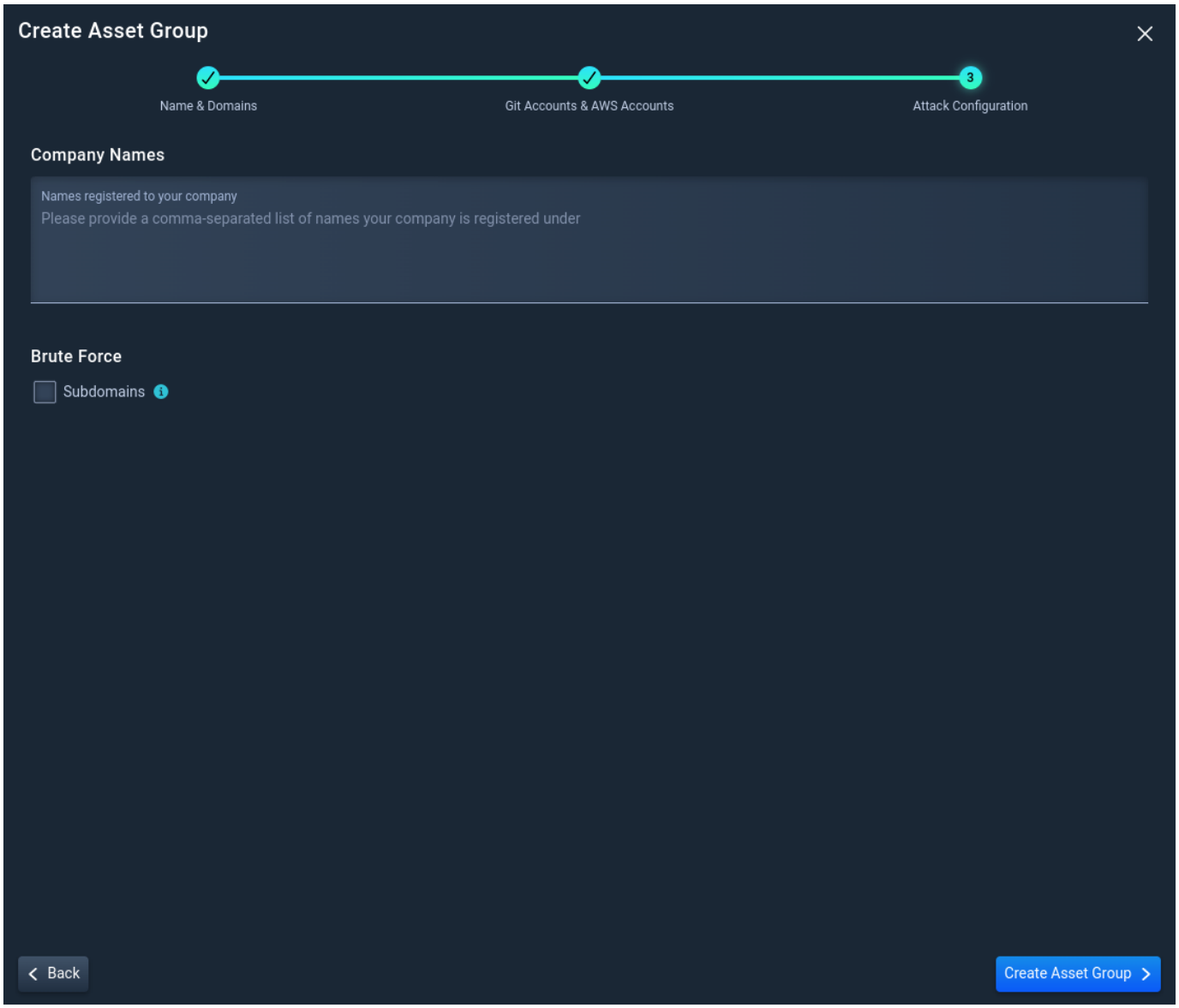
Once satisfied with the accounts, Click Next.



3.3 Attack Configuration Options

- Add company name(s) that NodeZero will use for Open Source INTelligence (OSINT) gathering tools and techniques to harvest company information.
- Enable brute force on subdomains to authorize NodeZero to search for well-known subdomains that may not surface through OSINT discovery.

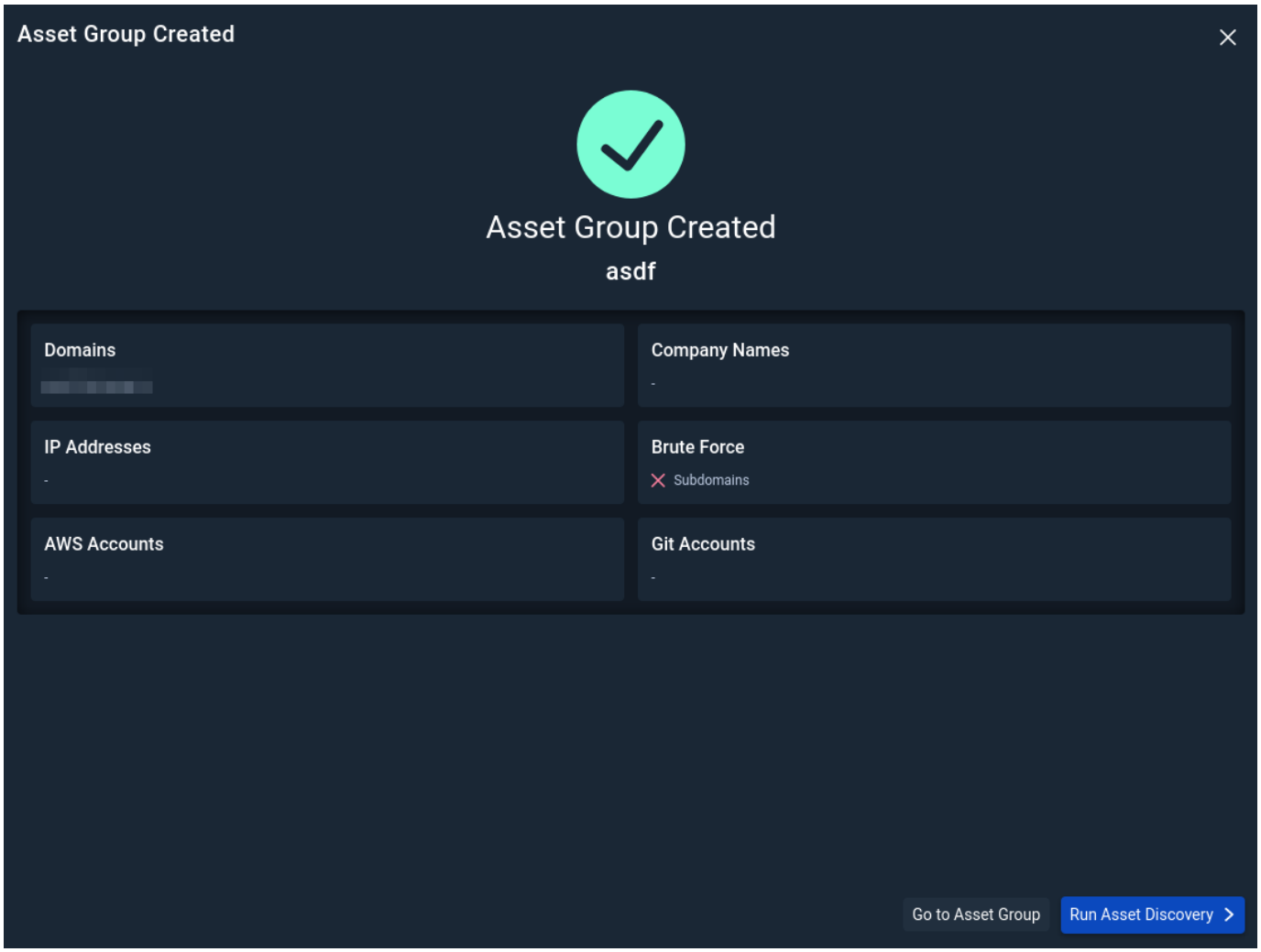
Once satisfied with the configuration options, click Create Asset Group.



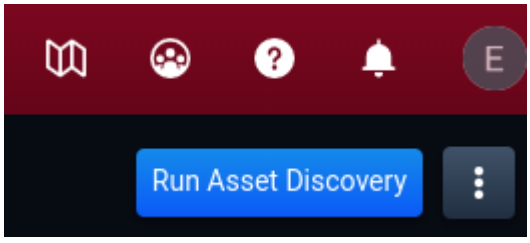
RUN ASSET DISCOVERY

NodeZero's External Asset Discovery is a passive enumeration capability that leverages DNS, passive website crawling, certificate scraping, and Open Source Intelligence (OSINT) gathering capabilities and services to find all the assets linked to your organization. No exploitation is performed during this operation.

If you just created your Asset Group, review the Asset Group configuration and click Run Asset Discovery. Alternatively, if changes are needed, click Go to Asset Group and click Edit Configuration in the top right.



If you are viewing your Asset Group, click Run Asset Discovery in the top right.



Asset Discovery is in progress

NodeZero sends an email once Asset Discovery finishes enumeration. Return to this documentation when your Asset Discovery is complete to learn how to authorize discovered assets for external pentesting.

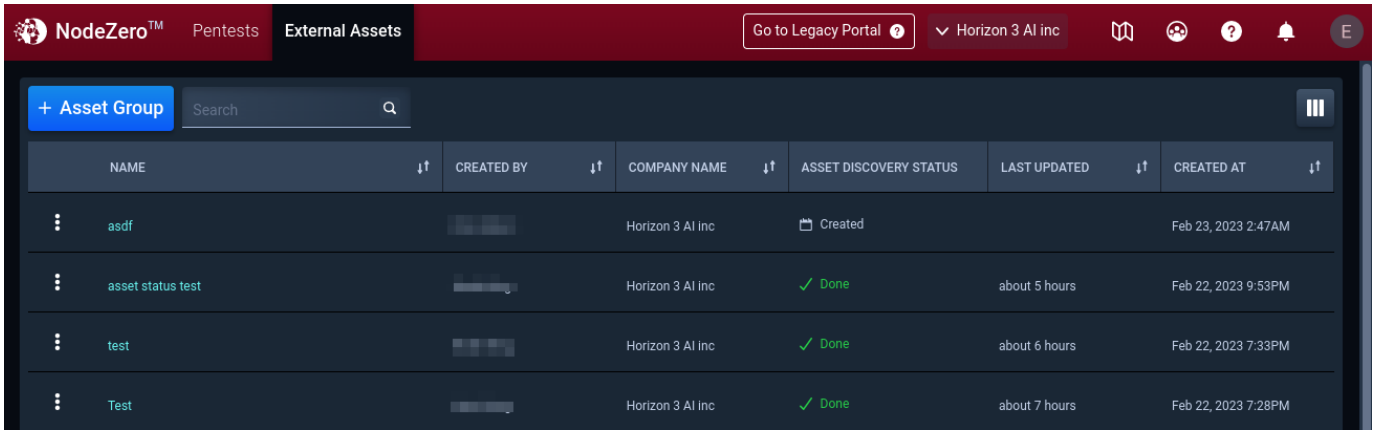
AUTHORIZE ASSETS

1. Navigate to External Assets

Now that Asset Discovery has been completed, navigate to External Assets to review and authorize assets for external pentesting.

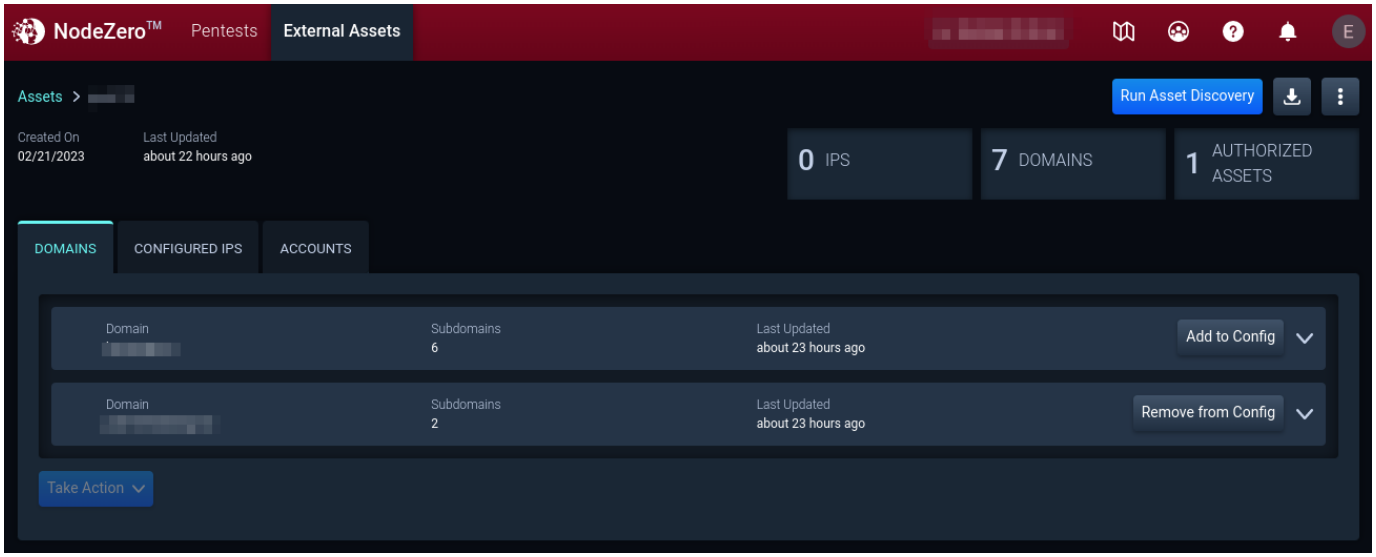
2. Click on the Asset Group

Click the external asset group with the “Done” Asset Discovery Status.



3. Review Discovered Domains

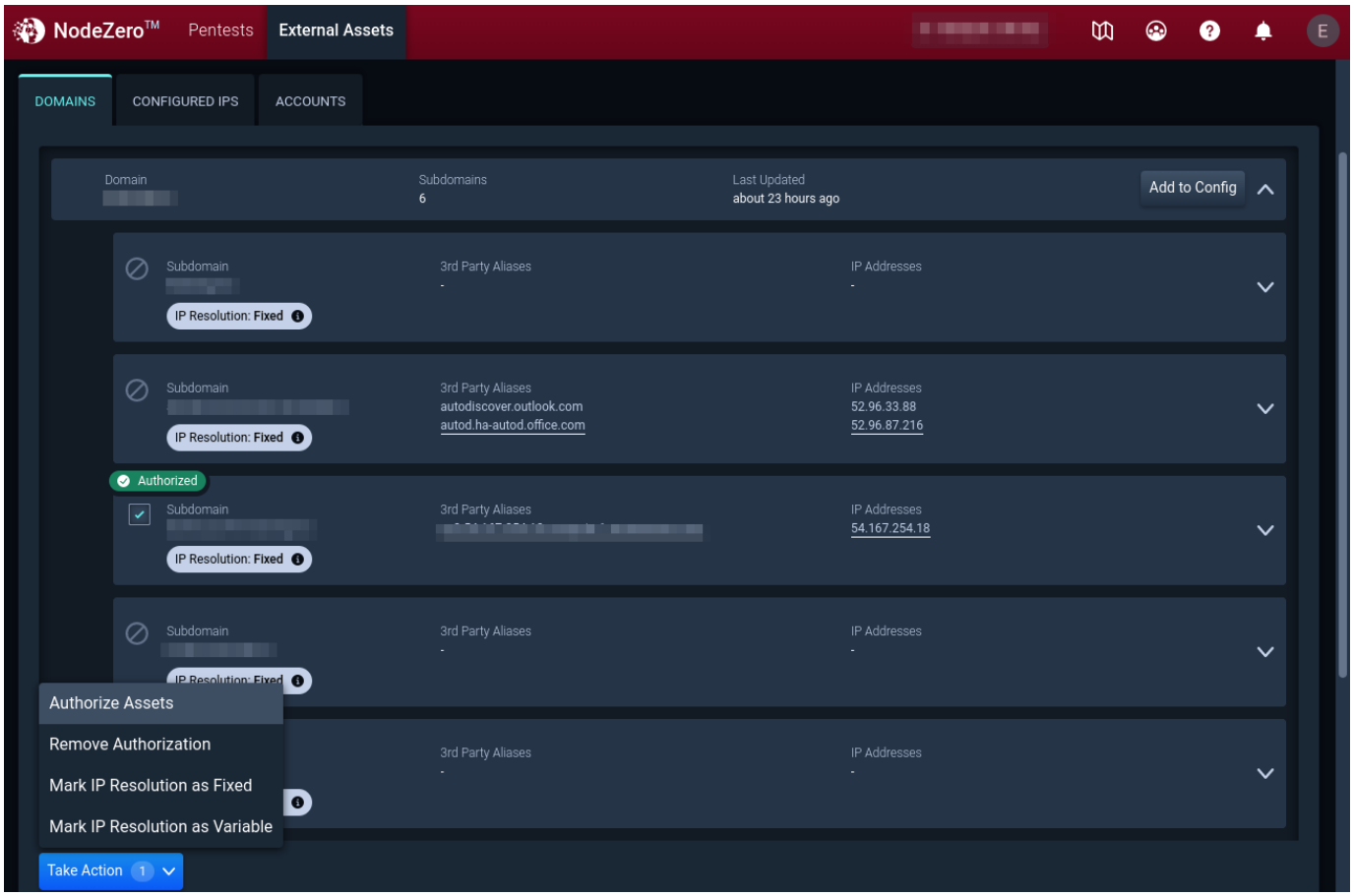
During Asset Discovery, NodeZero may find related company domains which can be added to the asset group’s configuration. To add these discovered domains to the asset group’s configuration, click Add to Config on the domain in the list. Otherwise, go to the next step.



> Rerun Asset Discovery on a routine basis to have the most up-to-date information on the status of your assets.

4. Review Subdomains to Authorize Assets

Expand a Domain to review discovered subdomains identified during Asset Discovery. To authorize assets for a pentest, select the asset, click Take Action, and then Authorize for Pentest.



Only authorize the assets you are legally permitted to pentest. You are responsible for accurately defining the scope of the Services for both internal and external testing. See Terms and Conditions for more information.

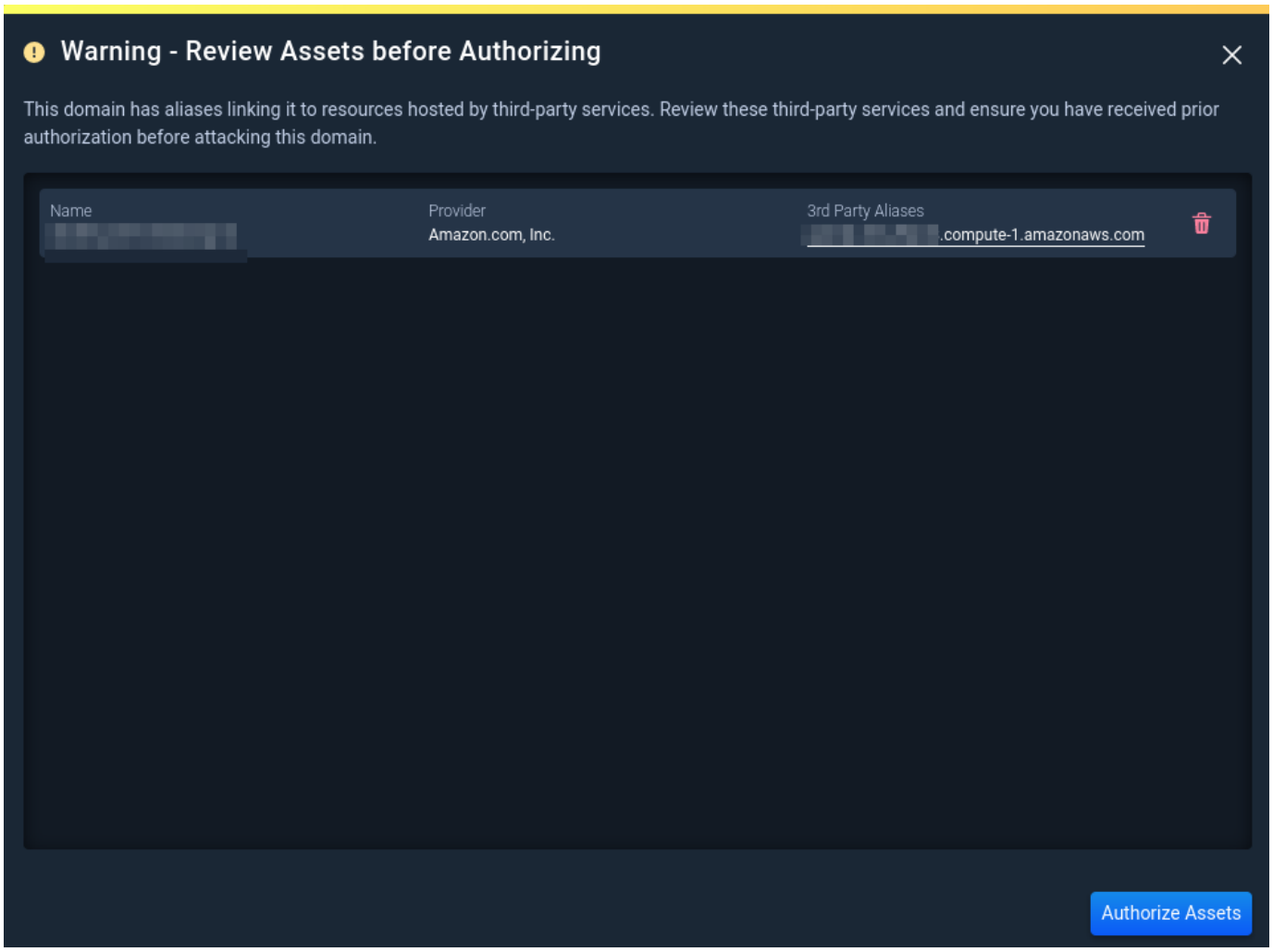
4.1 Assets with Warnings

Assets may have warnings indicating that they are hosted on third-party aliases. Before authorizing, verify you are legally allowed to pentest these assets. Possible reasons for the asset warnings:

- Asset links to services such as Rackspace and Digital Ocean don’t provide their pentesting guidance and NodeZero cannot confirm the allowance of pentesting their services
- Asset links to some unknown third-party service and you need to determine if external pentesting is allowed or not based on the terms and conditions of the service

4.2 When authorizing an asset with a warning...

When authorizing an asset with a warning, this pop-up requires verification that you are aware that you are authorizing assets that may fall outside of your domain. From this warning, you may remove assets from this list by clicking the trashcan. Click Authorize if you are legally allowed to pentest all the assets listed.



4.3 Asset IP Resolution

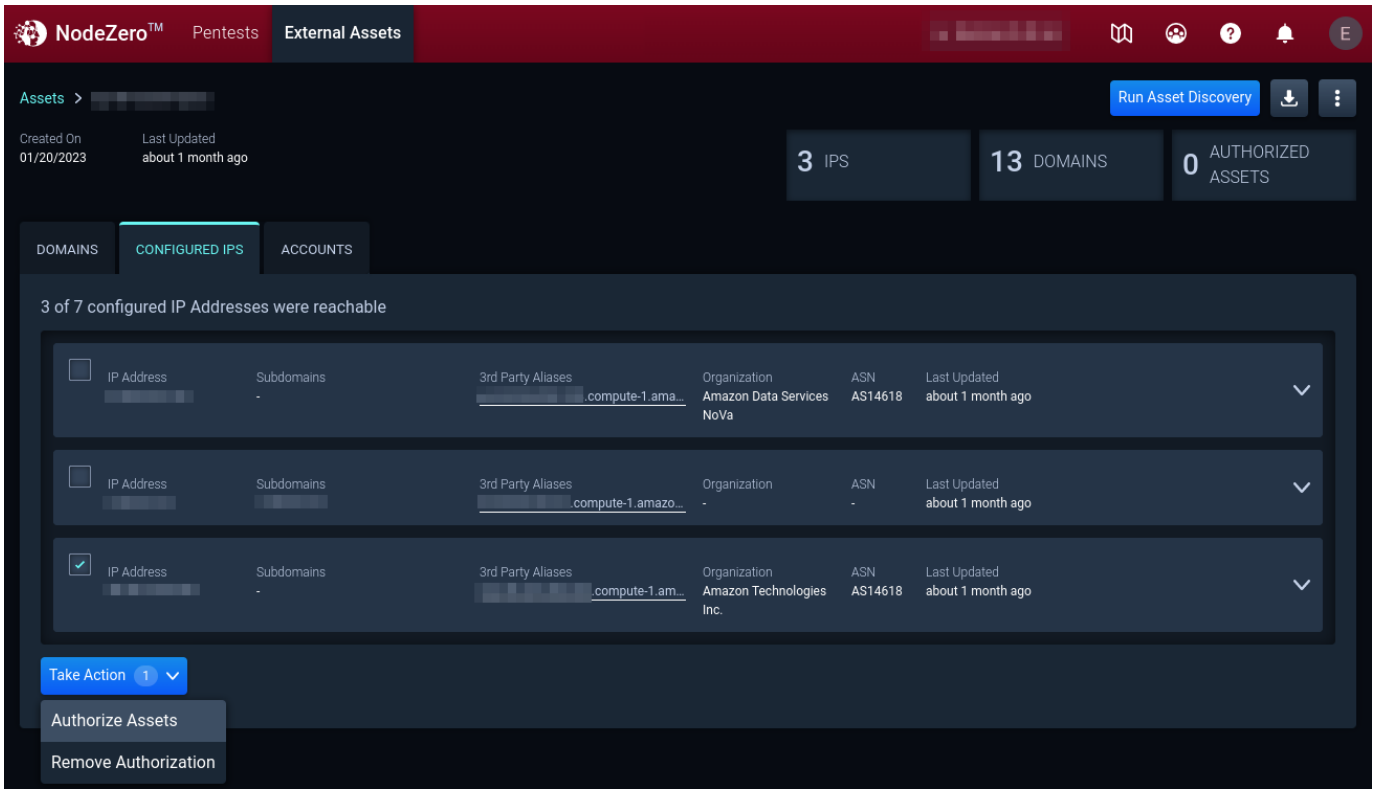
During an external pentest, NodeZero uses the asset's IP Resolution to determine if an asset stays in scope for the pentest. IP Resolution may be marked as Variable or Fixed. By default, assets are set to Fixed IP Resolution.

Fixed IP Resolution indicates the IP address resolved from the domain name is not expected to change. Assets labeled Fixed will be removed from scope if the asset resolves to a different IP address during the External Pentest than it did during the Asset Discovery.

Variable IP Resolution indicates the IP address resolved from the domain name may be expected to change over time. Assets labeled Variable will remain in scope even if the IP address changes between the Asset Discovery and the External Pentest. An example of a Variable asset IP would be AWS assets for which the resolved IP address is controlled by AWS and may change between pentests.

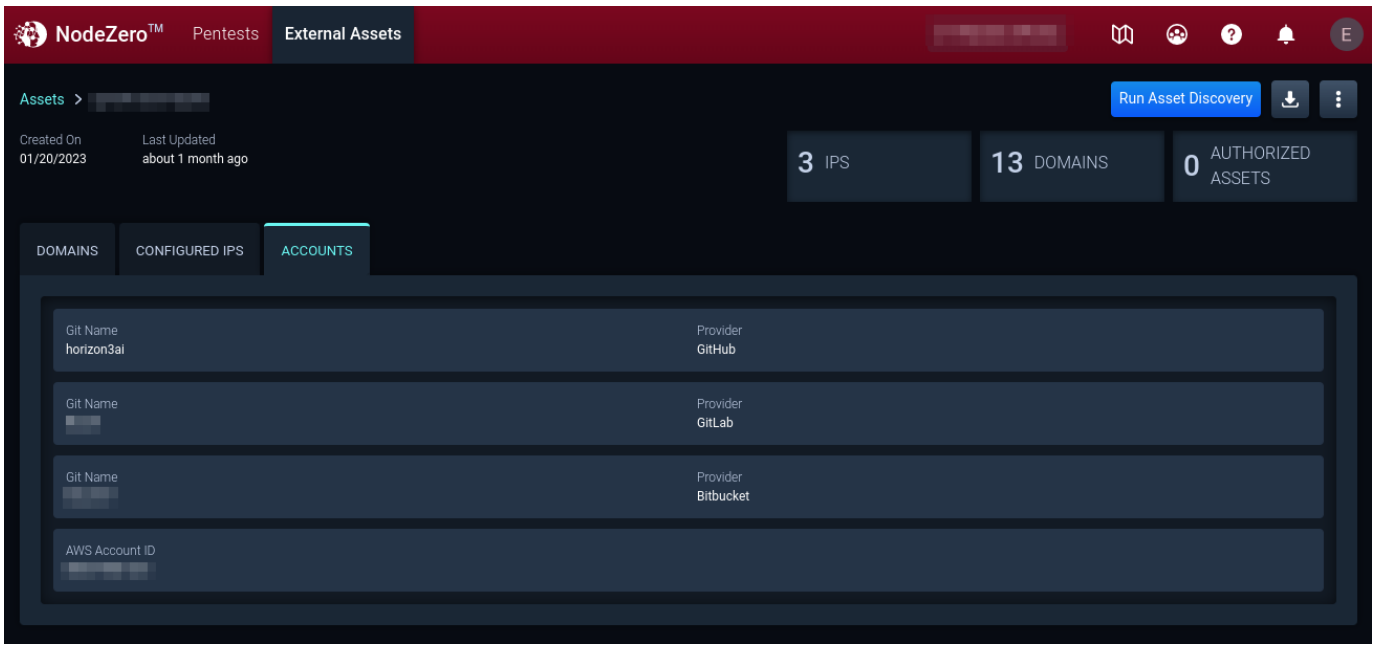
5. Configured IPs

On the Configured IPs tab, view reachable IP addresses in the Asset Group configuration. To authorize an IP address for a pentest, select the IP address, click Take Action, and then Authorize for Pentest.



6. Accounts Tab

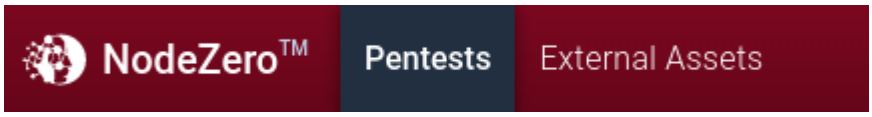
On the Accounts tab, view Git and AWS accounts added to the Asset Group configuration. To add or remove Git or AWS accounts, edit the asset group configuration by clicking the menu button in the top right of the asset group.



START AN EXTERNAL PENTEST

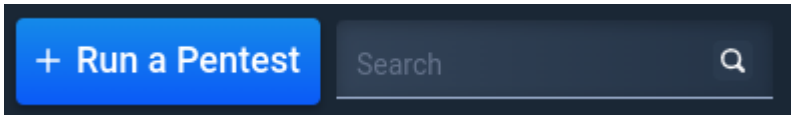
1. Navigate to Pentests to Run an External Pentest

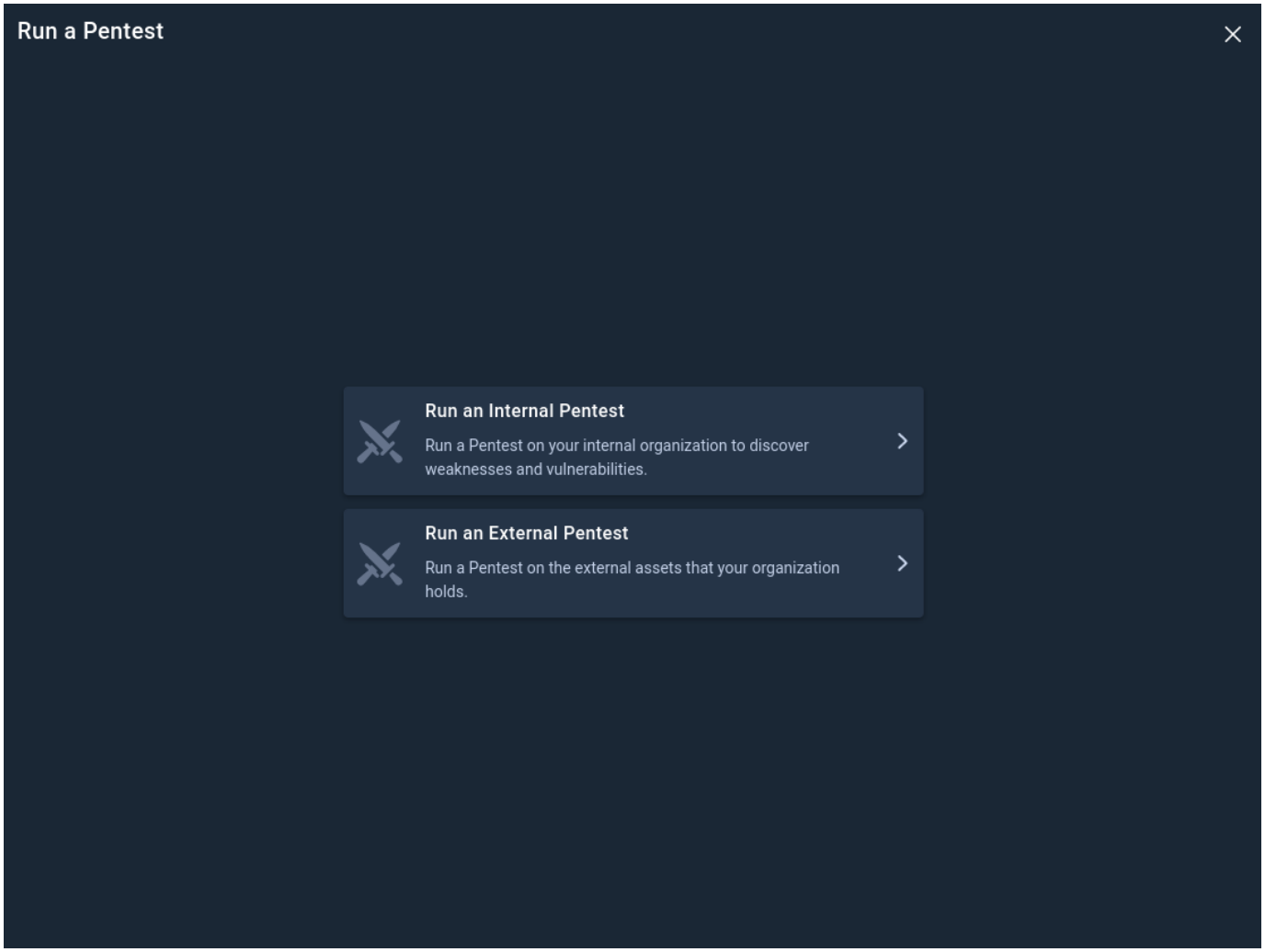
Once an Asset Group contains the assets authorized for pentesting, you may navigate to the Pentests page to start an external pentest.



2. Click + Run a Pentest

Click + Run a Pentest to open the Pentest Configuration and select External Pentest



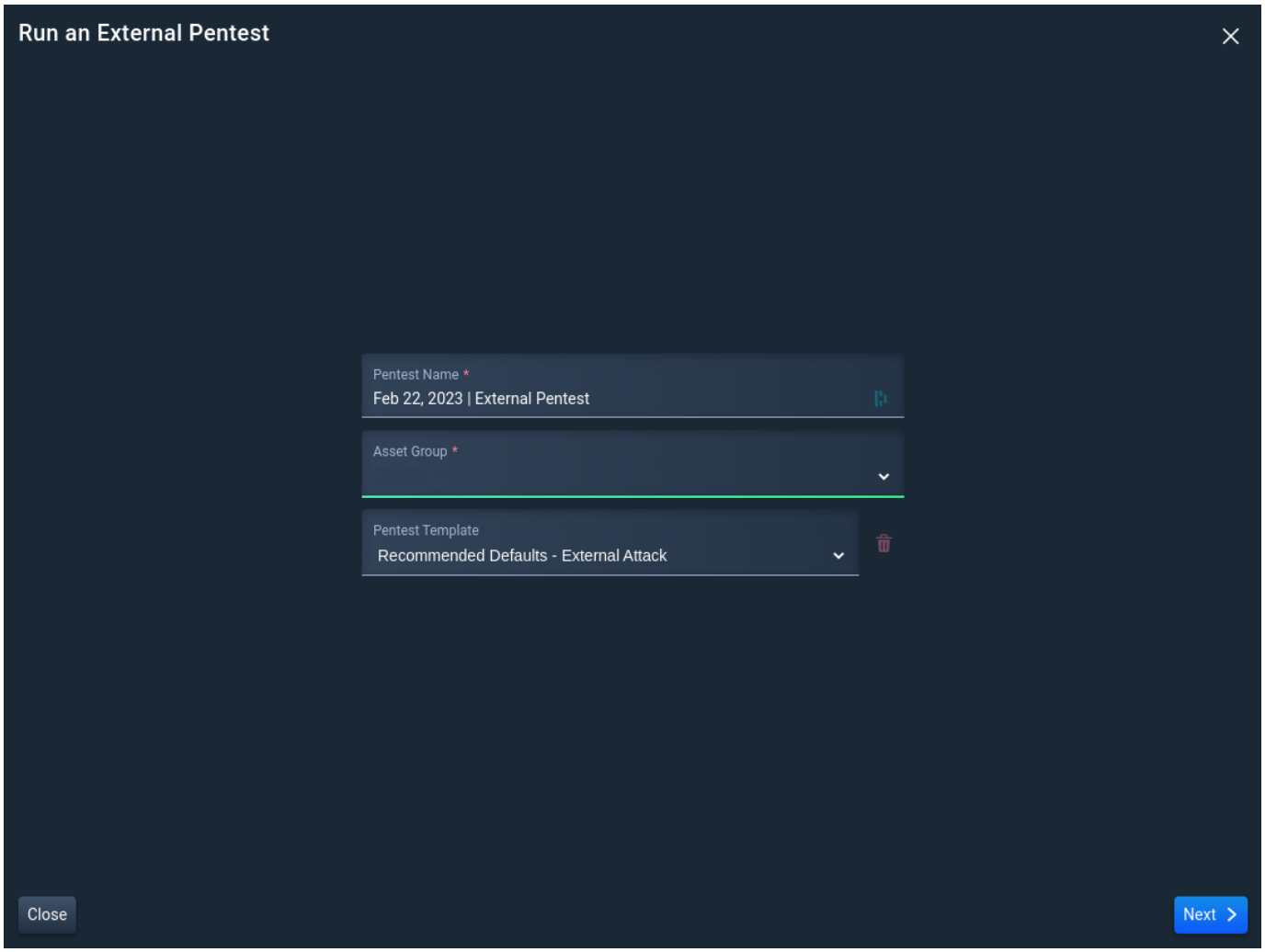


3. Configure the External Pentest

3.1 Set a Scope for the External Pentest

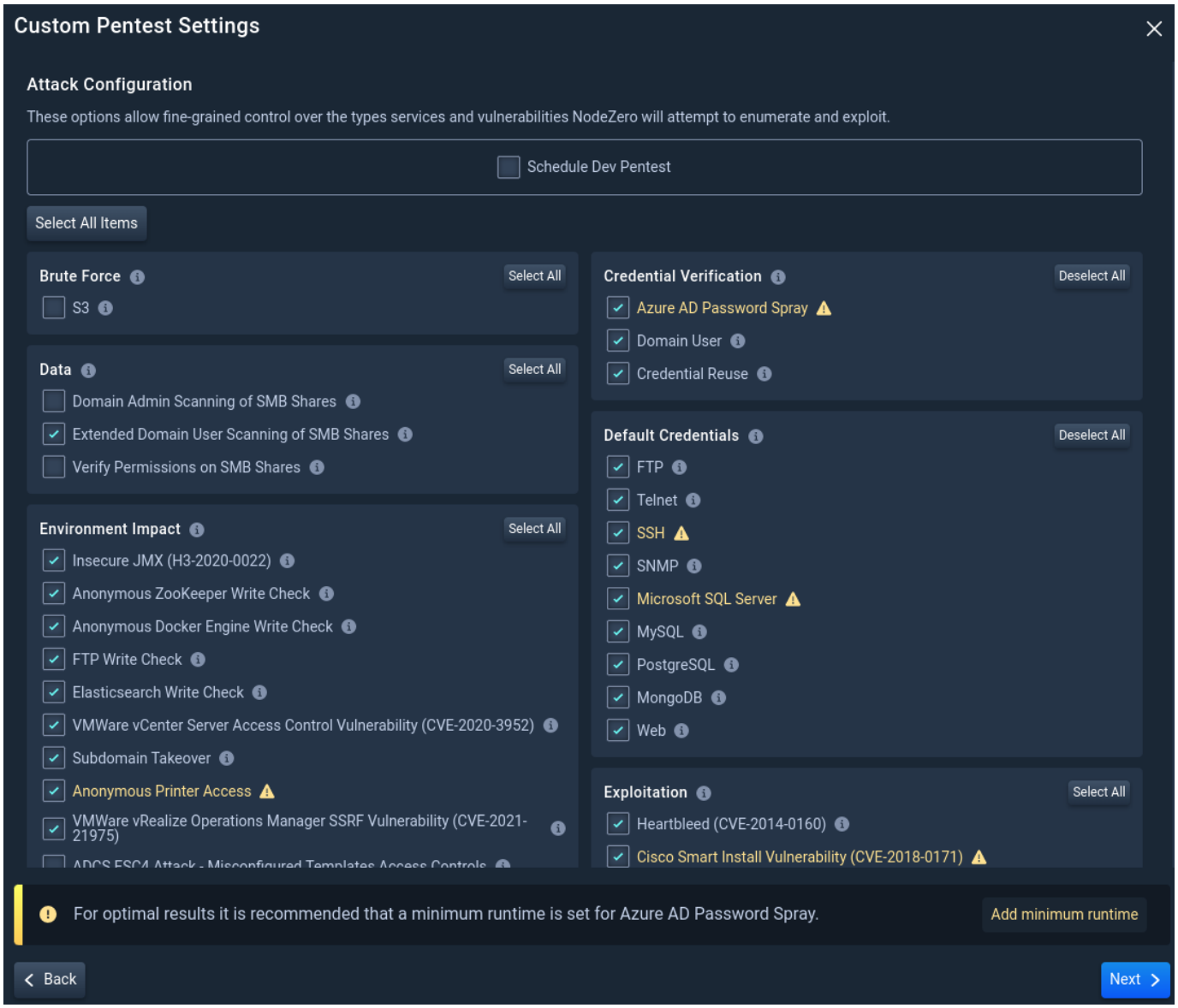
Name the External Pentest, select a pentest template, and select an Asset Group with authorized assets.

If no changes are needed to the template configuration, click Run Pentest to start the pentest now.



3.2 Advanced Configuration Options

Select the types of services and vulnerabilities NodeZero will attempt to enumerate and exploit. Click Next.

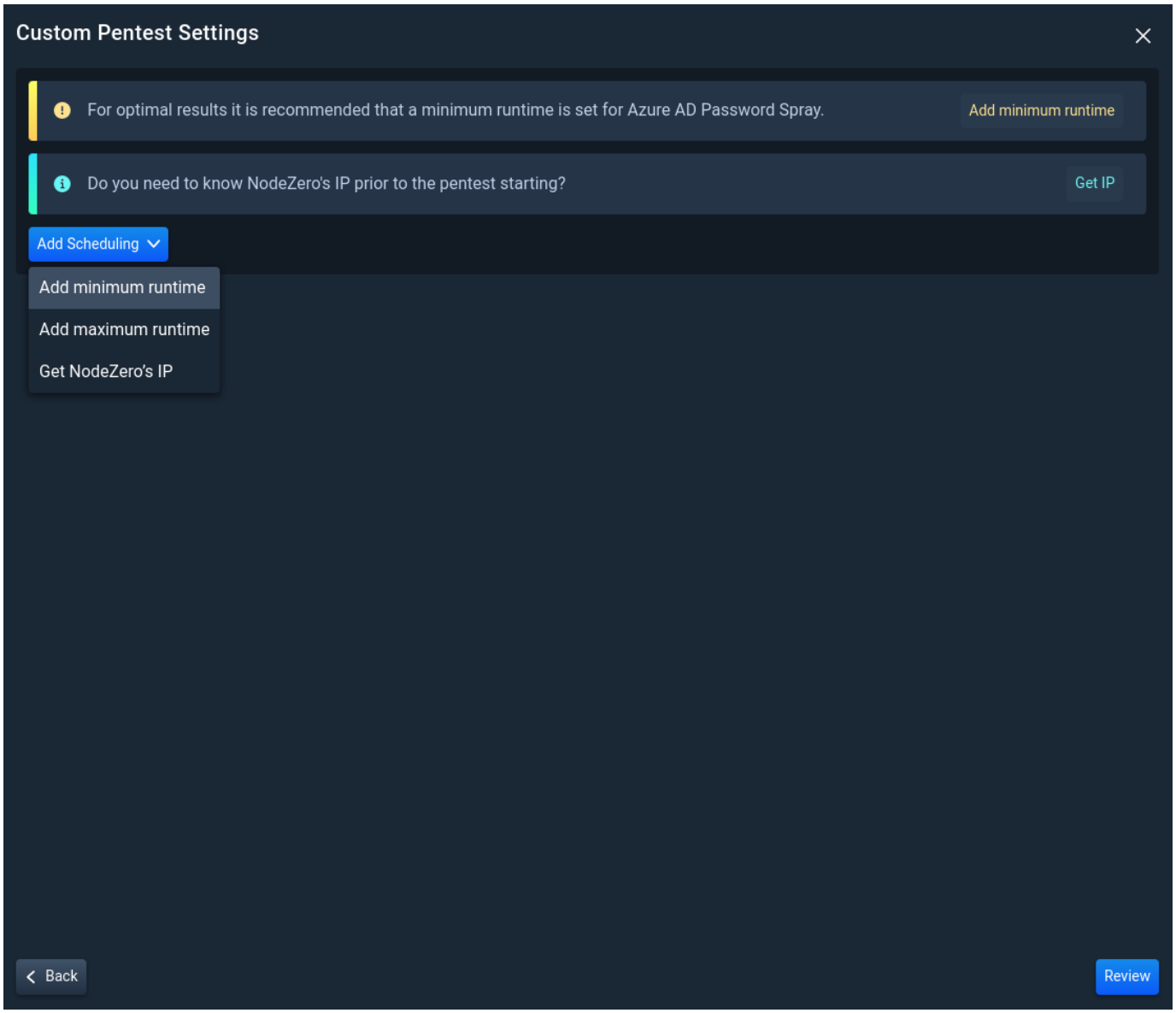


3.3 Additional Pentest Options

Add additional pentest options to further customize your pentest:

- Add a minimum or a maximum amount of time for a pentest
- Obtain the External Pentest's IP

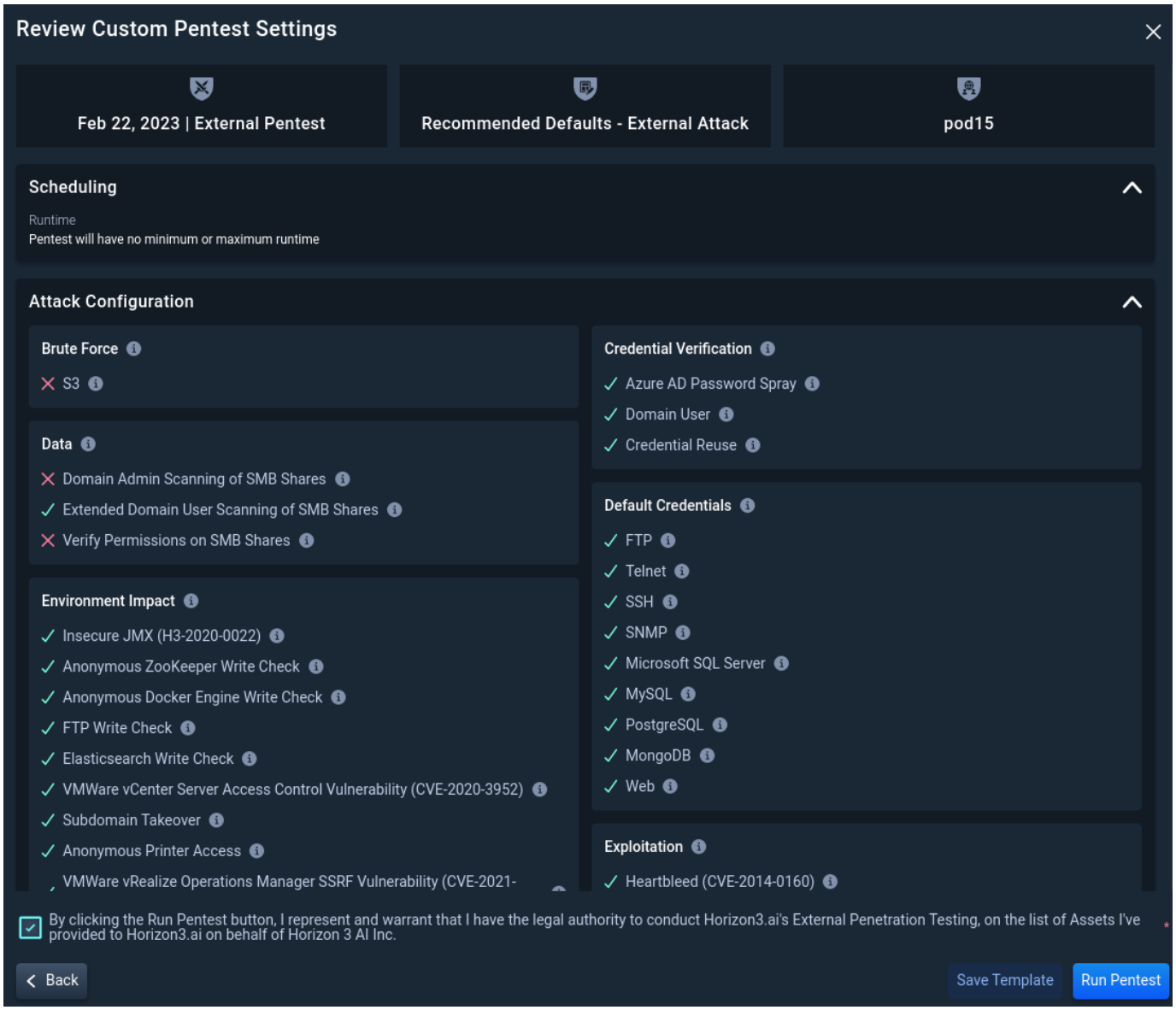
Then, click Review.



If you select to Get NodeZero's IP, NodeZero will acquire an IP, Pause the Pentest, and notify you that an IP has been acquired. After you have allow-listed the NodeZero IP where necessary, Resume the pentest from the Real-Time View.

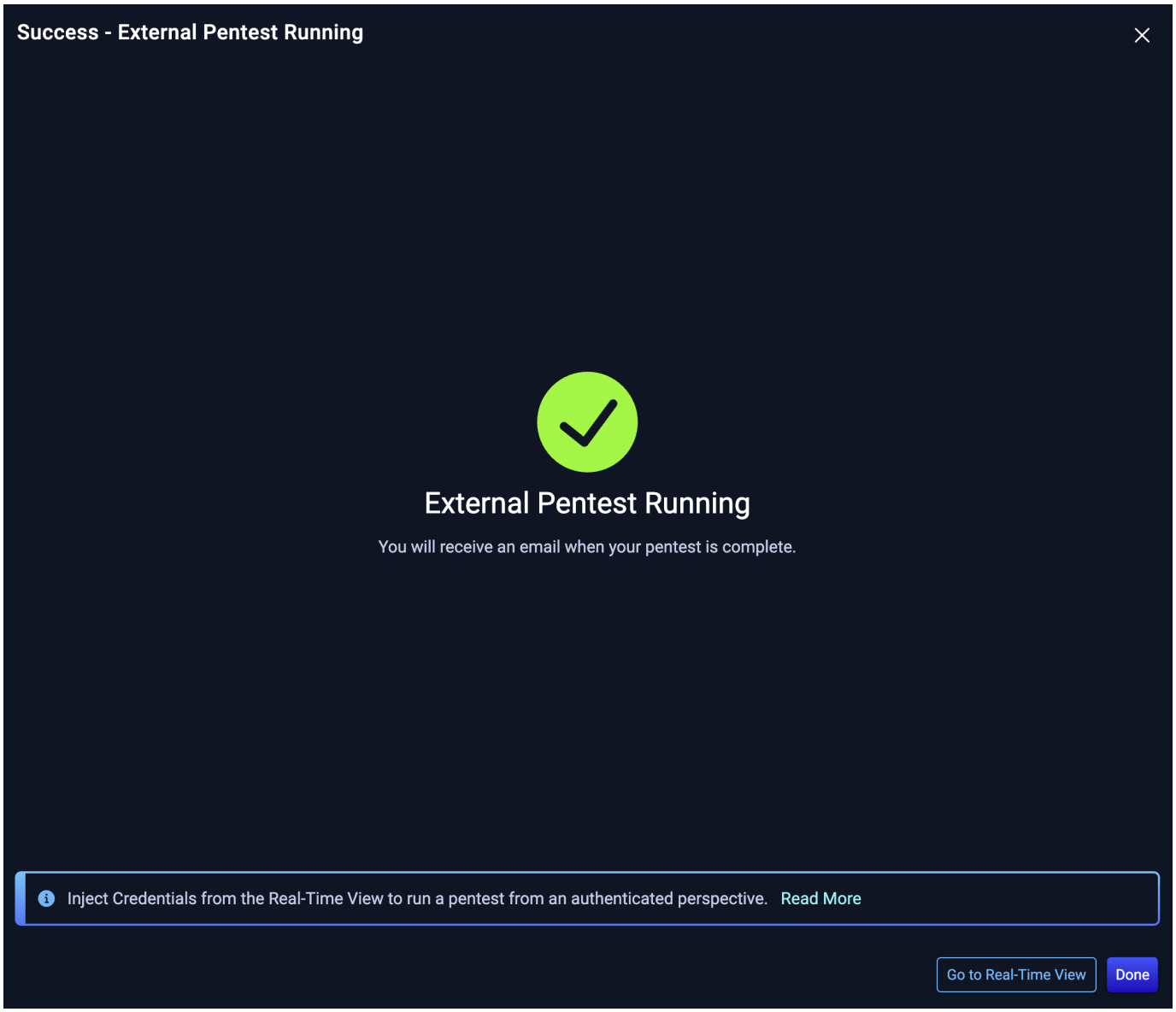
3.4 Review the External Pentest Configuration

Once satisfied with your pentest selections, check the box to indicate you represent and have the legal authority to conduct Horizon3.ai's External Penetration Testing on the list of authorized assets. Then click Run Pentest.



You've started an External Pentest

NodeZero sends an email once the External Pentest completes.



Tip

NodeZero can also run pentests from an authenticated perspective. Go to the Real-Time View and Inject Credentials to see the impact an attacker would have by leveraging compromised credentials!

2.4 Setup NodeZero Host

2.4.1 Setup NodeZero Host

NodeZero is our prepackaged software module that functions as an attacker within your network. To run NodeZero against your internal network, a host is needed within the network to act as the breach point

USING THE NODEZERO OVA

> **OVA is preferred**
>
> We highly recommend using our OVA for NodeZero installation, as it is the preferred method to ensure a seamless setup.

The OVA comes with the `h3-cli` preloaded and has the `n0` update utility to help with keeping the host up to date with the latest changes for the NodeZero host.

To setup your environment for use with the OVA:

1. Download the OVA
2. Setup and Configure Host
3. Deployment Options
4. Validate NodeZero Host

CREATING NODEZERO HOST FROM SCRATCH

Follow the steps below to learn more about the NodeZero Host, system requirements, deployment locations, and installation process. Ensure that if you need to build your own host, it meets the necessary specifications.

1. NodeZero Host Information

a. Host System Requirements

b. Connectivity Requirements

c. Proxy Setup

2. Docker Installation
3. Validate NodeZero Host
4. Deployment Options
5. (optional) NodeZero CLI

Once you have your NodeZero Host ready, log into a shell on it using your favorite method.

2.4.2 NodeZero Host Information

The NodeZero host serves as the operating platform and starting point for Horizon3.ai's autonomous pentesting solution. We highly recommend using Linux, but customers have successfully employed Windows Docker. Position the NodeZero host within the network segment where you want the pentest to begin. Before starting the operation, ensure that the host is running to download and execute NodeZero, and maintain its operation throughout the process. Once the operation is complete, you may shut down or remove the host from the network.

If you prefer a different distribution, please contact the Horizon3 team for compatibility assessment. Although most distributions should run NodeZero smoothly, we have not yet performed checks or validations for all of them.

Host System Recommendations

Minimum Specifications

- 2 CPU (x86-64, physical or virtual)
- 8GB RAM
- 20 GB free HDD space

Operating System

- Ubuntu Linux 18.x, 20.x, or higher
- Redhat Linux 7.x (EoL Jun 2024)

It is possible to run the NodeZero on other OSs, however, we will not provide support in the event issues arise while using them.

Network Access

- Various HTTPS (443/tcp) access to AWS services (SQS, Cognito, S3, ECR, etc)
- See Connectivity Requirements for more information

Docker installed

- Most recent version of Docker
- Minimum required version: 20.10
- See Docker installation instructions here

Connectivity Requirements

The `Core` service, which serves as the central intelligence for NodeZero, resides in a single-use architecture within the cloud. The NodeZero host needs access to Core via HTTPS on port 443 to facilitate communication. This connection can be likened to a central nervous system, with the brain sending messages to the hands and receiving feedback to analyze and determine the next appropriate action.

From a service perspective, NodeZero must be able to communicate with Core. We currently use AWS SQS, Cognito, and S3 over HTTPS on port 443.

Regarding assessment perspective, it is crucial not to modify your environment. NodeZero is a unique service and tool, and you should not alter your settings for it as you wouldn't for an attacker. For example, if your firewall is set to block the marketing VLAN from accessing the finance VLAN, leave it as is. NodeZero will verify that this configuration is in place.

> If your environment connects to the internet via a proxy, this will affect NodeZero's ability to communicate out. Directions to configure NodeZero for use with a proxy can be found here.

OUTBOUND NETWORK ACCESS

Network access requirements are based on what portal instance generates the test, and not where the NodeZero host is being run. Uninterrupted network access is required during the entire operation to the following endpoints:

For US-Based Logins: (portal.horizon3ai.com)

- HTTPS - 443/tcp

```
api.horizon3ai.com
cognito-identity.us-east-2.amazonaws.com
cognito-idp.us-east-2.amazonaws.com
downloads.horizon3ai.com
sqs.us-east-2.amazonaws.com
*.docker.com
*.docker.io
*.ecr.us-east-2.amazonaws.com
*.queue.amazonaws.com
*.s3.amazonaws.com
*.s3.us-east-2.amazonaws.com
*.s3-w.us-east-2.amazonaws.com
```

- HTTP - 80/tcp

```
*.interacth3.io
```

For EU-Based Logins: (portal.horizon3ai.eu)

- HTTPS - 443/tcp

```
api.horizon3ai.eu
cognito-identity.eu-central-1.amazonaws.com
cognito-idp.eu-central-1.amazonaws.com
downloads.horizon3ai.com
sqs.eu-central-1.amazonaws.com
*.docker.com
*.docker.io
*.ecr.eu-central-1.amazonaws.com
*.queue.amazonaws.com
*.s3.amazonaws.com
*.s3.eu-central-1.amazonaws.com
*.s3-w.eu-central-1.amazonaws.com
*.execute-api.eu-central-1.amazonaws.com
*.elb.eu-central-1.amazonaws.com
*.s3-r-w.eu-central-1.amazonaws.com
```

- HTTP - 80/tcp

```
*.interacth3.eu
```

For NodeZero Runner EU and US based

- HTTPS - 443/tcp

```
raw.githubusercontent.com
github.com
```

> **Note: HTTP use**
>
> No sensitive information of any kind is transmitted over this channel

Consolidated Endpoint

> **Beta**
>
> **This feature will need to be enabled for your account. Please contact your CS rep.**
> User experience is subject to change.

If you are operating the NodeZero host within a restricted network environment, the consolidated endpoint feature can simplify networking requirements. Instead of opening outbound network traffic to all the AWS services listed above, you will only need to allow traffic for the two static IP addresses associated with these domains:

US-Based

- Domains

```
gateway.horizon3ai.com
interact.gateway.horizon3ai.com
```

- IPs

```
15.197.206.82
3.33.191.122
```

EU-Based

- Domains

```
gateway.horizon3ai.eu
interact.gateway.horizon3ai.eu
```

- IPs

```
52.223.20.205
99.83.187.197
```

INBOUND NETWORK ACCESS

The following ports should be opened on the NodeZero host/VM to allow traffic in:

- TCP 21, 23, 25, 53, 80, 88, 110, 135, 139, 143, 389, 443, 445, 587, 1433, 3306, 3389, 5900, 5985, 8080, 8443, 8888, 28069, 45000-49999
- UDP 69

This is required on the NodeZero host and does not pertain to perimeter firewalls.

2.4.3 Installing Docker

We highly recommend using Linux for your NodeZero Host.

“Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and deploy it as one package. By doing so, thanks to the container, the developer can rest assured that the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code.”

Linux Guide

To host the NodeZero Docker container, Docker Community Edition is required and compatible with most Linux Operating Systems. For comprehensive installation instructions, we recommend referring directly to Docker's official documentation.

INSTALL DOCKER ON UBUNTU

For Ubuntu users, the best resource for installing Docker is the Docker Installation Page for Ubuntu. This page provides step-by-step guidance tailored to different versions of Ubuntu, ensuring you have the most relevant and up-to-date information for your installation.

Version Compatibility

Make sure you are using Docker version 20.10 or higher. You can verify your Docker version by running `docker --version` in your terminal.

ADDITIONAL LINUX DISTRIBUTIONS:

If you're using a different Linux distribution, you can find specific installation guides on Docker's official website:

- Install Docker Engine Information Page
- Install Docker on CentOS
- Install Docker on Debian
- Install Docker on Fedora

Success! Time to validate

Once you've installed Docker on your Linux OS, you're done! Now validate NodeZero is ready to operate.

2.4.4 Validate NodeZero Host

Once the installation is complete, validate that your NodeZero host is ready to run operations by running the NodeZero environment validation script.

Run the following command to run the script:

```
curl https://downloads.horizon3ai.com/utilities/checkenv.sh | bash
```

The output should look similar to the following:

```
# https://downloads.horizon3ai.com/utilities/checkenv.sh | bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 18646  100 18646    0     0  61134      0 --:--:-- --:--:-- --:--:-- 61134

[#] This script validates the environment is ready to run NodeZero.

[#] Checking Operating System:
[+] PASSED: Linux is a supported Operating System.

[#] Gathering environmental variables to conduct further checks:
[+] PASSED: All environmental variables set and proceeding with next checks.

[#] Checking Docker functionality by running the hello-world test container:
[+] PASSED: Docker is installed and functioning properly.

[#] Checking Docker permissions to volume mount files from /Users/test/test directory:
[+] PASSED: Docker permissions are correct for the /Users/test/test directory location.

[#] Checking connectivity to AWS resources:
[+] PASSED: s3.us-east-2.amazonaws.com is reachable.

[#] Checking 20GB HDD space requirements:
[+] PASSED: There is enough space for the NodeZero container: 34.0386GB

[#] Checking memory requirements:
[+] PASSED: This system has 32GB RAM which meets the recommended minimum to support NodeZero.

[#] Checking compute resource requirements:
[+] PASSED: This system has 20 CPUs which meets the minimum logical CPU requirements to run NodeZero.

[#] Script complete. If any checks failed, correct and re-run before attempting to 'Run a Pen Test'
```

Now you're ready to run an internal pentest!

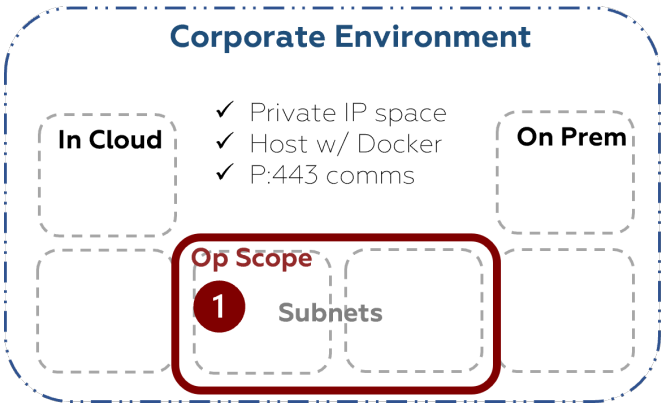
2.4.5 Deployment Options

The launch point of your NodeZero Host may change based on what you want to learn from the pentest. The following are a few options you may find helpful in making this decision.

	Custom Scope			Intelligent Scope	RFC 1918	OSINT
NodeZero Placement	1 Inside the Scope	2 Outside the Scope	3 Outside @ Endpoints (i.e., /32s)	4 Attack Starting Point	5 Full private scope	6 NOTE: CloudZero launches externally
Intent	I want to limit the scope and see what an attacker could exploit from inside that defined range	I want to limit the scope to see if an attacker can access hosts and data inside	I want to look at specific endpoints to check for host vulnerabilities and misconfigurations	I want to see what an attacker can discover and access from a specific starting point	I want to search every nook and cranny of private IP space accessible in my environment	I want to see what publicly available data makes our business vulnerable to an attacker
Will enumerate and exploit:	✓ in-scope hosts, services, domain, web, credentials, & data resources ✓ ProTip: ensure a DC is “in scope”	✓ in-scope hosts, services, web, credentials (except MITM and PTH attacks) and cloud assets	✓ specified hosts, ports, services, web and certs, exploitable vulnerabilities	✓ Discovered hosts, services, domain, web, credentials & data resources	✓ Discovered hosts, services, domain, web, credentials & data resources	✓ Publicly available user names, subdomains (from TLDs), and web-facing attack surface
Won’t execute	× On anything outside the prescribed scope	× Man-In-The-Middle attacks	× On infrastructure nor chained vulnerabilities and misconfigurations that could lead to compromise	× On inaccessible hosts, services, domain, web, credentials	× On inaccessible hosts, services, domain, web, credentials	× On internal assets * NOTE: when combined with an - internal op w/ access to a DC, will verify user/password access
Use Cases	❑ Internal Pentest ❑ SOC SLAs ❑ Verify policies ❑ Verify EDM/SIEM	❑ Internal Pentest ❑ Verify Segmentation ❑ Verify access to a sensitive VLAN ❑ Third party security assessment	❑ Test EDR ❑ Assess endpoint vulnerabilities	❑ Internal Pentest ❑ Verify Segmentation ❑ Verify Policies ❑ Verify EDR/SIEM ❑ Test Blast Radius ❑ Test ZeroTrust	❑ Internal Pentest ❑ Environment & Asset Discovery ❑ Assess hybird env ❑ Verify Policies ❑ Verify EDR/SIEM	❑ Public-facing Reconnaissance ❑ Company recon ❑ User recon ❑ Subdomain recon ❑ *Cred stuffing

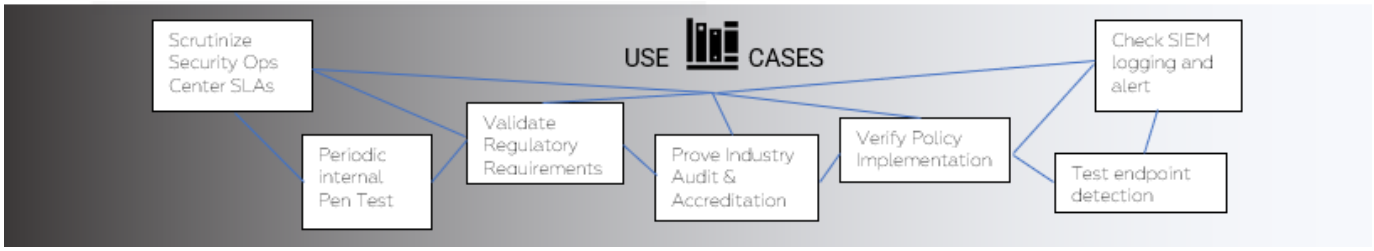
1. Inside Custom Scope

If you want to limit the scope and see what an attacker could exploit from inside a defined range, place the NodeZero host within the scope you want to test. When you configure the scope for your pentest, ensure the NodeZero Host is within one of the specified CIDR range(s) for the test.



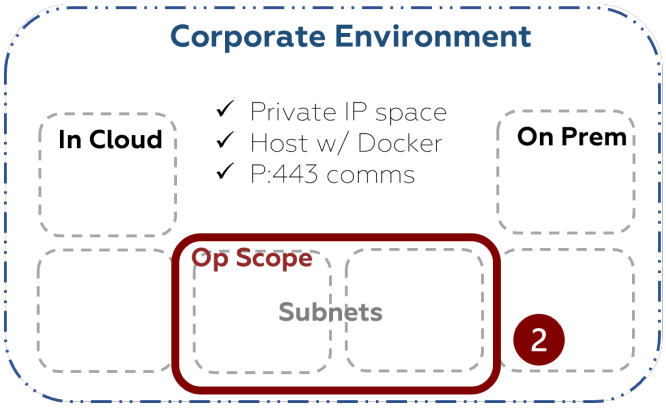
Ensure a Domain Controller is in-scope as well, and NodeZero will attempt to exploit any vulnerabilities or misconfigurations on this host, as well as verify weak domain defaults & credentials

This is your high-speed assessment; enabling a lean Find-Fix-Verify loop to initiate an agile security posture



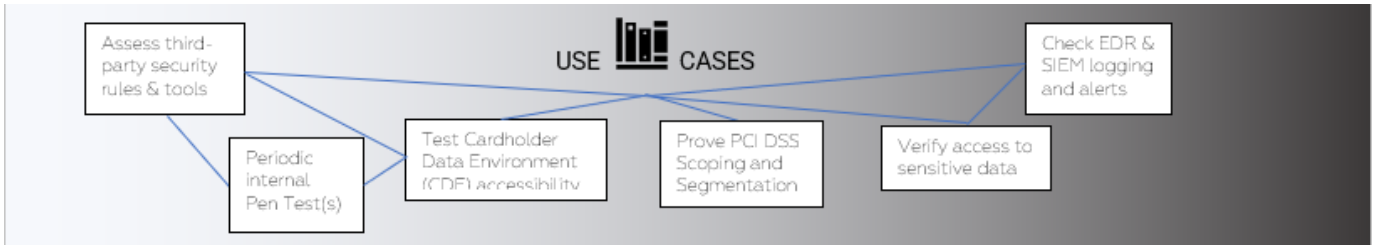
2. Outside Custom Scope

If you wanted an “outside-in” perspective to see if an attacker could access critical data and assets inside a specific scope, place the NodeZero Host outside the scope you want to test. When you configure the scope for your pentest, ensure the NodeZero Host is NOT within the specified CIDR range(s) for the test.



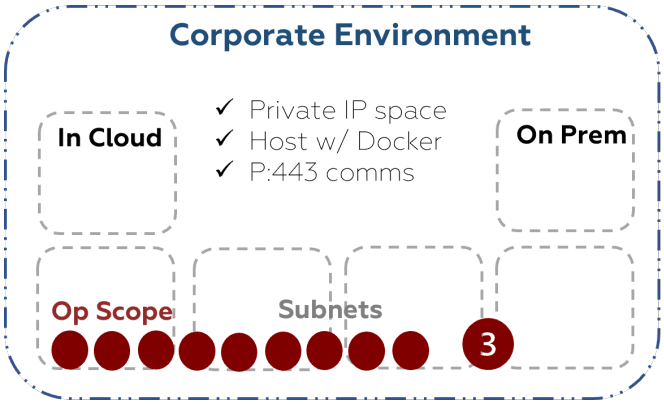
When NodeZero is not in the same IP range as the scope, it will not execute Man-In-The-Middle and pass-the-hash attacks

This is your unrestricted assessment; providing true insight into what is accessible, valuable, and vulnerable from any starting point



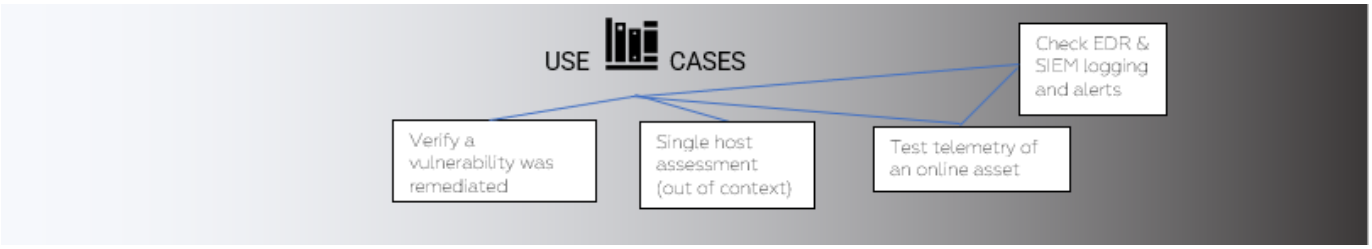
3. Endpoints Only Scope

If you want to quickly verify if the vulnerability you just remediated had the effect you desired, select a single host or range of hosts by /32s. When you configure the scope for your pentest, ensure the NodeZero host has access to the specific host identified by the /32 CIDR range(s) for the test but is NOT within the specified CIDR range(s) for the test.



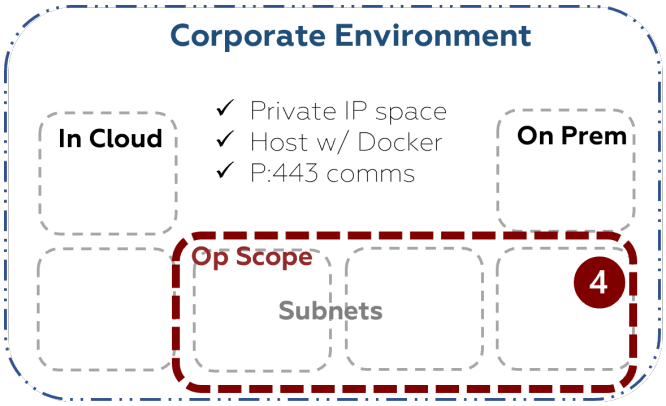
When NodeZero is not in the same IP range as the scope, it will not execute Man-In-The-Middle and pass-the-hash attacks. Further, with this restricted scope, NodeZero will chain neither weaknesses nor paths as you have limited the scope to a specific endpoint for this assessment

This is your restricted assessment; a quick turnaround op to verify your fix-action was implemented and a vulnerability is now presenting less severity to your attack surface

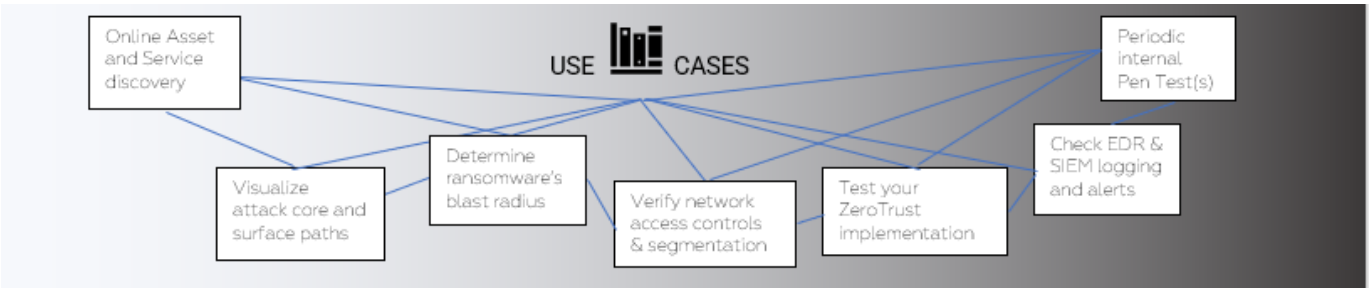


4. Intelligent Scope

If you wanted to see what an non-credentialed attacker could enumerate and exploit from a specific starting point in your network – a true “black box” pentest – use Intelligent Scope. When you configure the scope for your pentest, leave the “Include” box blank. NodeZero’s host subnet will provide the initial scope and it will expand organically during the pentest as more hosts and subnets are discovered similar to how an attacker would.



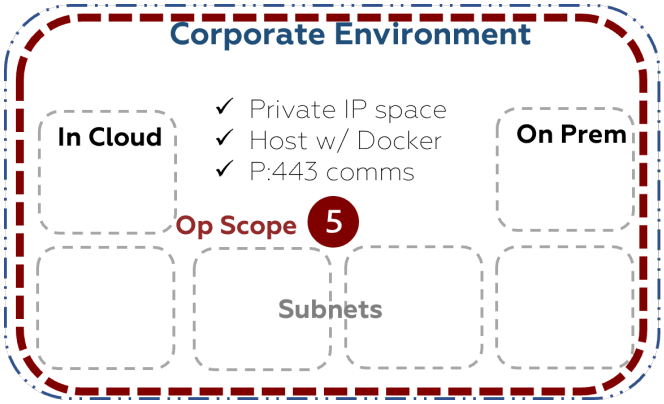
This is your proactive assessment; providing true insight into what is accessible, valuable, and vulnerable from any starting point



5. All Private IP Scope (i.e., RFC 1918)

Use RFC 1918 to run a private scope pentest, enumerating everything accessible quickly and safely.

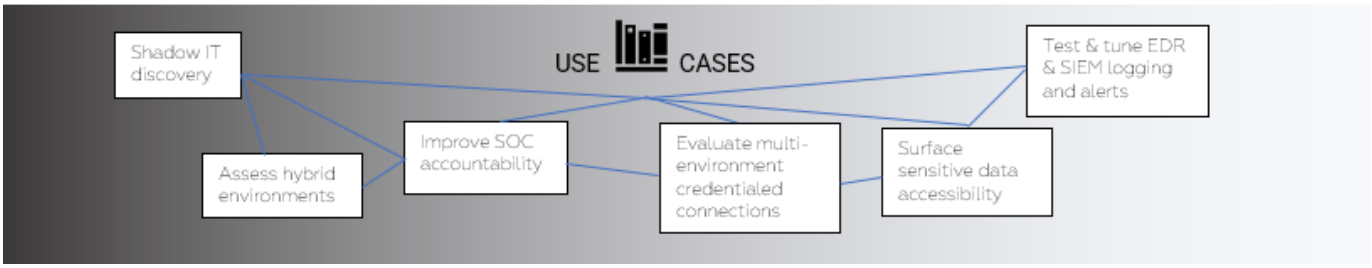
When you configure the scope for your pentest, check “Use RFC 1918” and NodeZero will take care of the rest. If there are IP addresses or ranges you do not want to be assessed, add them to the "Exclude" box when configuring the scope for this pentest.



This op may take a bit longer as NodeZero enumerates any IPs and DNS names it can access...including edge routers; if yours are misconfigured for routing private IPs, NodeZero may attempt to enumerate those external private IPs.

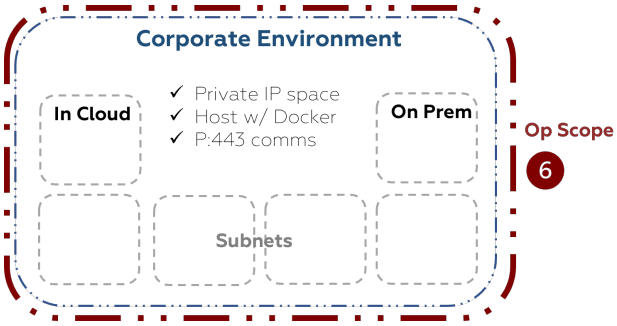
if you want to see EVERYTHING, put NodeZero in an unrestricted ACL so it can discover every nook and cranny online in your environment

This is your unrestricted and holistic enterprise assessment–and should be run regularly.



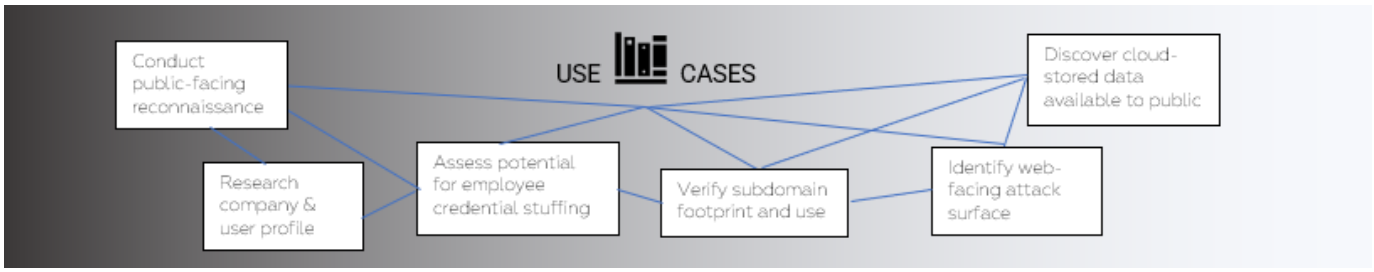
6. OSINT Focused

Available with any of the pentest types is our Open-Source Intelligence (OSINT) assessment, where NodeZero will gather publicly available information to use as part of the pen test. The second step of configuring your pentest offers you the ability to take a true external perspective; your company name will be auto-filled for you, and you can provide TLDs and weak password terms you’d like NodeZero to test with any discovered information.



NodeZero's OSINT gathering operates outside your environment so NodeZero placement isn't as critical, however, when combined with an internal pentest with a domain controller in-scope, NodeZero will verify domain users and passwords with those found publicly giving your further insight into your attack surface risk.

This is your external reconnaissance capability to see what attackers see and use to start their campaigns and establish a foothold in your environment



Use this table as a reference for all your pentest operations!

	Custom Scope			Intelligent Scope	RFC 1918	OSINT
NodeZero Placement	Inside the Scope	Outside the Scope	Outside @ Endpoints (i.e., /32s)	Attack Starting Point	Full private scope	NOTE: CloudZero launches externally
Intent	I want to limit the scope and see what an attacker could exploit from inside that defined range	I want to limit the scope to see if an attacker can access hosts and data inside	I want to look at specific endpoints to check for host vulnerabilities and misconfigurations	I want to see what an attacker can discover and access from a specific starting point	I want to search every nook and cranny of private IP space accessible in my environment	I want to see what publicly available data makes our business vulnerable to an attacker
Will enumerate and exploit:	✓ in-scope hosts, services, domain, web, credentials, & data resources ✓ ProTip: ensure a DC is "in scope"	✓ in-scope hosts, services, web, credentials (except MITM and PTH attacks) and cloud assets	✓ specified hosts, ports, services, web and certs, exploitable vulnerabilities	✓ Discovered hosts, services, domain, web, credentials & data resources	✓ Discovered hosts, services, domain, web, credentials & data resources	✓ Publicly available user names, subdomains (from TLDs), and web-facing attack surface
Won't execute	× On anything outside the prescribed scope	× Man-In-The-Middle attacks	× On infrastructure nor chained vulnerabilities and misconfigurations that could lead to compromise	× On inaccessible hosts, services, domain, web, credentials	× On inaccessible hosts, services, domain, web, credentials	× On internal assets * NOTE: when combined with an - internal op w/ access to a DC, will verify user/password access
Use Cases	❑ Internal Pentest ❑ SOC SLAs ❑ Verify policies ❑ Verify EDM/SIEM	❑ Internal Pentest ❑ Verify Segmentation ❑ Verify access to a sensitive VLAN ❑ Third party security assessment	❑ Test EDR ❑ Assess endpoint vulnerabilities	❑ Internal Pentest ❑ Verify Segmentation ❑ Verify Policies ❑ Verify EDR/SIEM ❑ Test Blast Radius ❑ Test ZeroTrust	❑ Internal Pentest ❑ Environment & Asset Discovery ❑ Assess hybird env ❑ Verify Policies ❑ Verify EDR/SIEM	❑ Public-facing Reconnaissance ❑ Company recon ❑ User recon ❑ Subdomain recon ❑ *Cred stuffing

2.5 Run an Internal Pentest

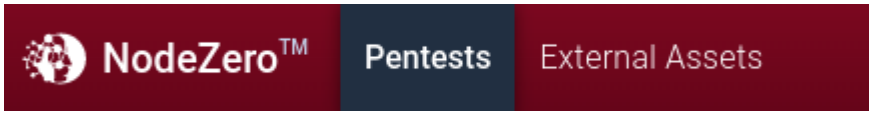
2.5.1 Run an Internal Pentest

NodeZero can assess all of your environments, from the attack surface of your hybrid cloud to your on-premise network infrastructure, helping you continuously find and fix your internal and external attack vectors before criminals exploit them. Follow these steps to run a pentest within your network.

How to Run an Internal Pentest

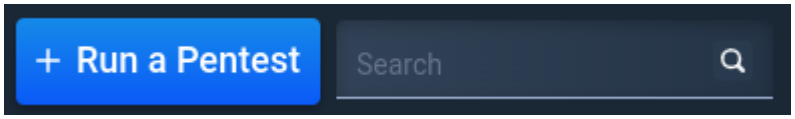
1. NAVIGATE TO PENTESTS TO RUN AN INTERNAL PENTEST

Once you've established a NodeZero Host that meets to requirements, navigate to Pentests to start a pentest.



2. CLICK + RUN A PENTEST

Click + Run a Pentest to open the Pentest Configuration and select Internal Pentest



3. CONFIGURE THE INTERNAL PENTEST

3.1 Name the Internal Pentest

Name the Internal Pentest and select a pentest template.

> Determine and follow a naming convention to allow you to quickly find a pentest from your pentest list.
>
> An example: [date]|[library]|[NodeZero Src]|[scope]
>
> 2021-09-01|NodeZero|East-Coast-Bizops|Full: This indicates that the NodeZero host was placed in the East Coast Bizops network and the scope was the entire enterprise.

3.2 Select a Scope

The pentest scope is the set of IPs and/or subnets (in CIDR notation) within which you want to run the pentest. The larger the scope, the better results you will get. This is not a “vulnerability scanner” that has a narrow focus. NodeZero assesses your environment and uses any data it finds, and the context around it, to identify and exploit your vulnerabilities, misconfigurations, and poor cybersecurity hygiene.

If you are unclear on CIDR notation, here is a reference and a calculator app to assist you:

- CIDR Notation Article
- CIDR Calculator

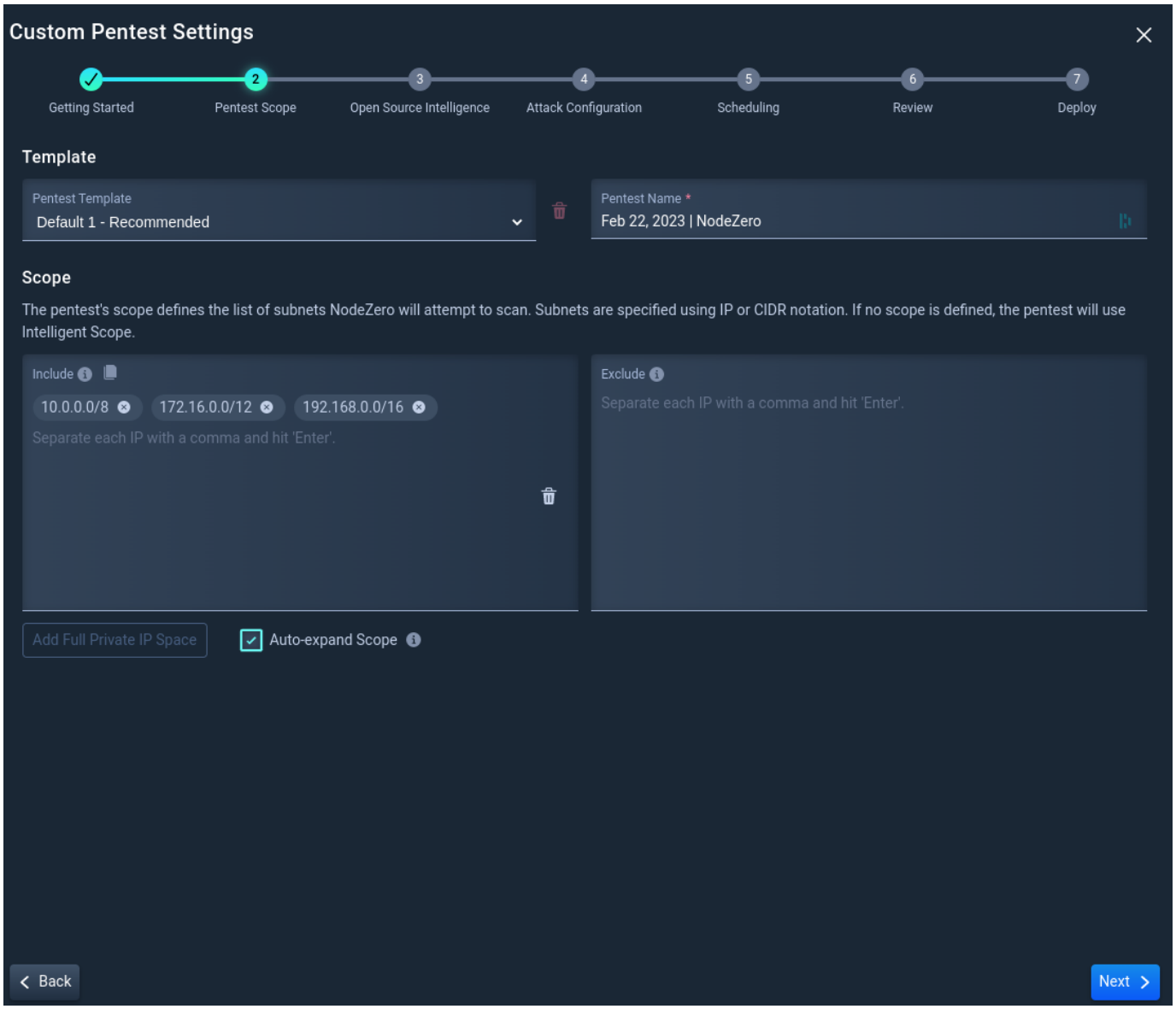
If your environment uses 192.168.0.1 and the subnet mask is 255.255.255.0, then you’ll add the following to the Include section: 192.168.0.0/24

For properly segmented environments, use comma-separated CIDR notation. For example: 192.168.0.0/16,172.16.10.0/24,10.0.0.0/8

If you are running NodeZero in a more complex environment, set the scope to cover as many subnets as possible. You should ask your Network Administrator for a list of CIDR annotated subnets.

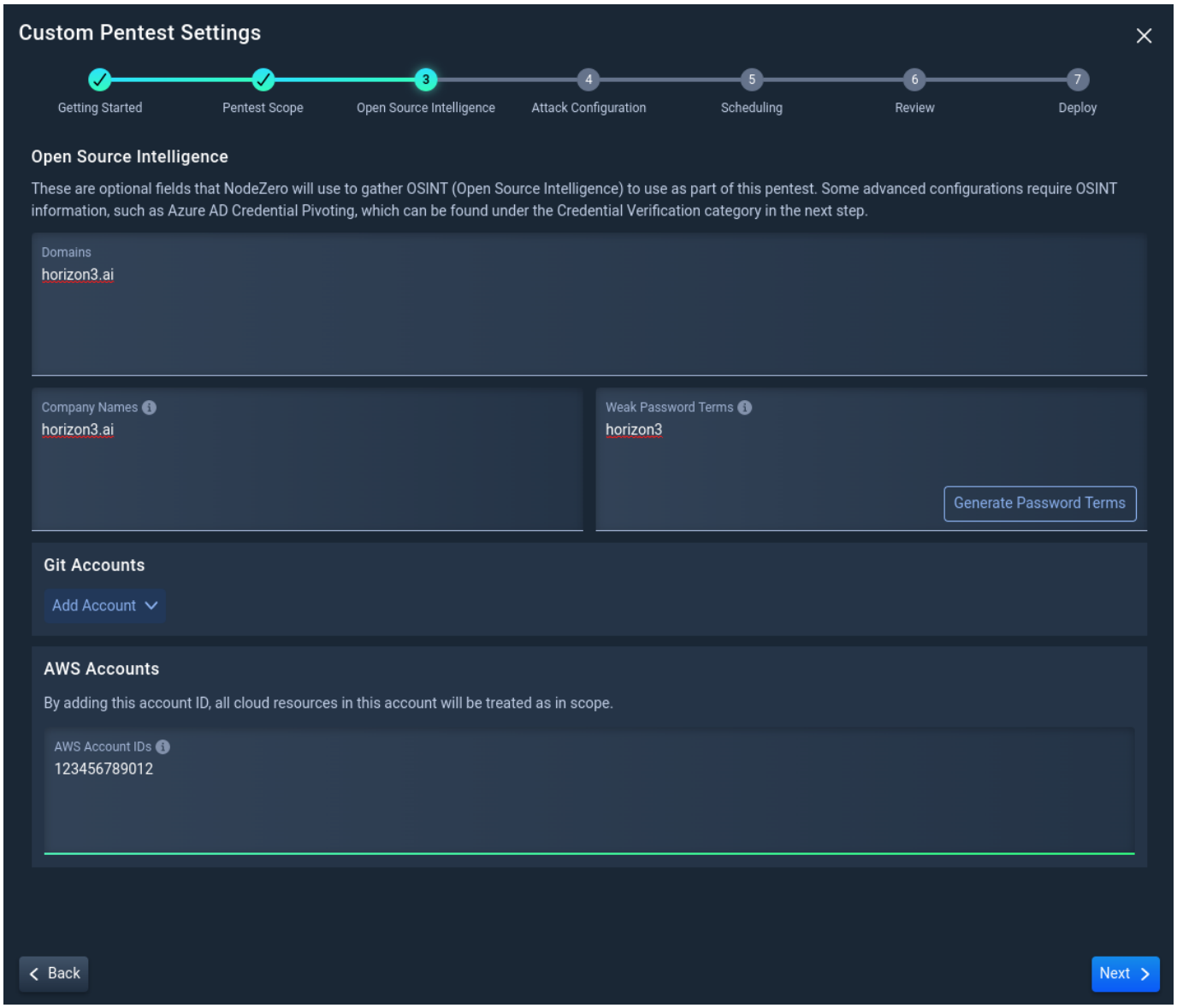
The Exclude section stops NodeZero from scanning or exploiting a set of IPs or subnets. The IPs within this section may be discovered by NodeZero via various techniques within the pentest, but NodeZero will not touch them. They may show up in the Out of Scope list within the pentest results. Note that this parameter also requires CIDR notation.

When satisfied with your scope, click Next.



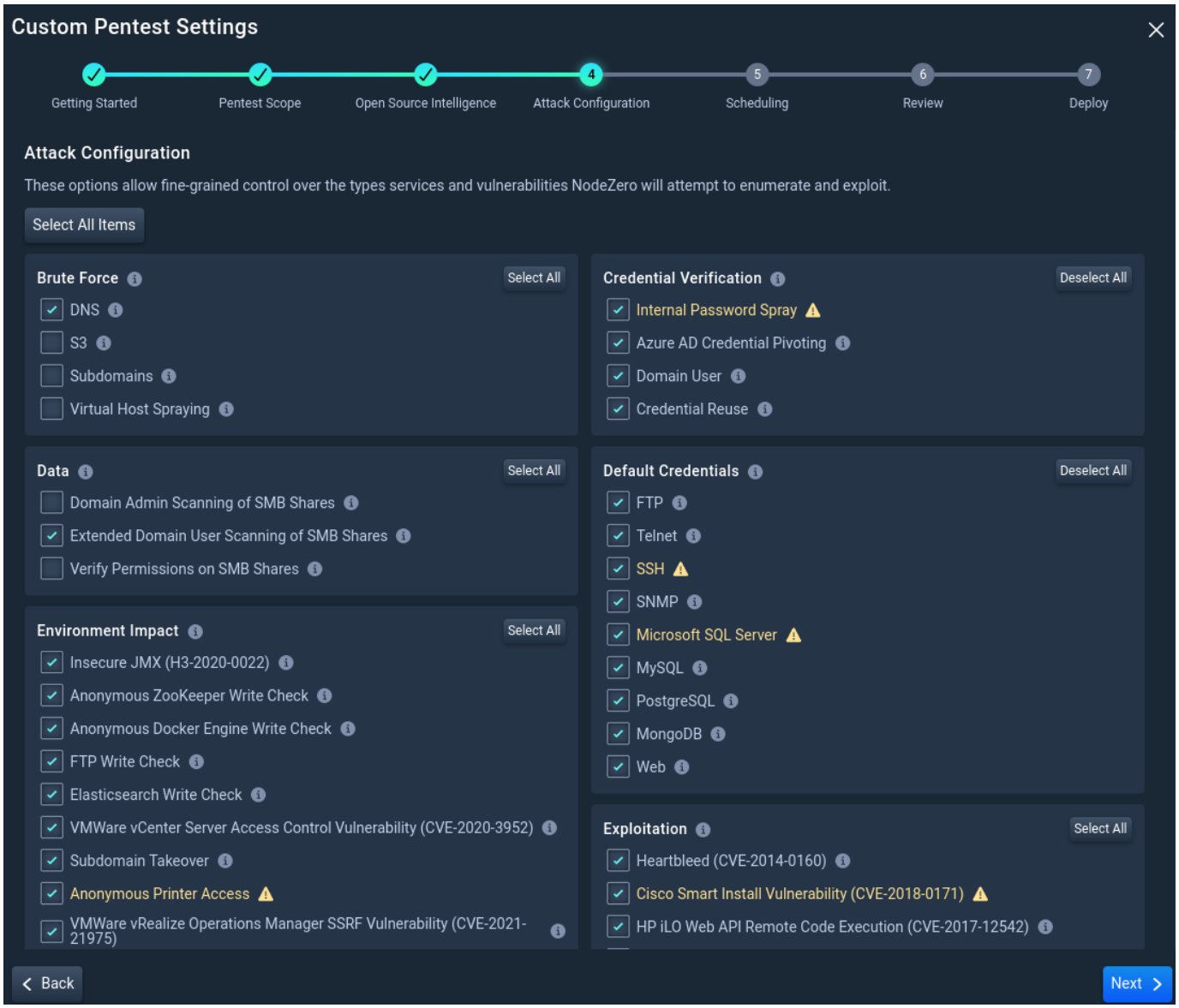
3.3 Add Open-Source Intelligence

Optionally add Domains, Company Names, Weak Password Terms, or Git and AWS Accounts



3.4 Advanced Configuration Options

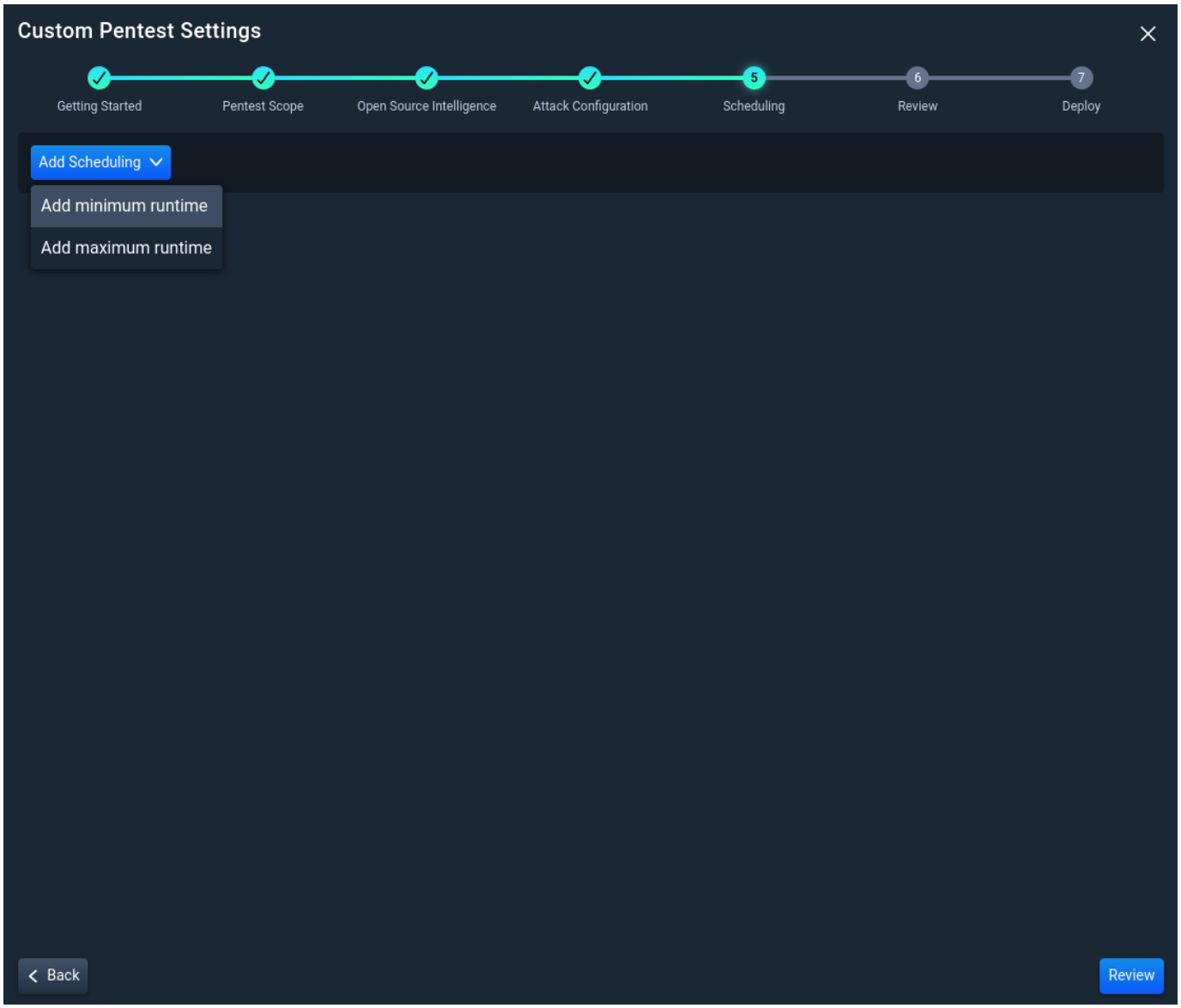
Select the types of services and vulnerabilities NodeZero will attempt to enumerate and exploit. Click Next



3.5 Additional Pentest Options

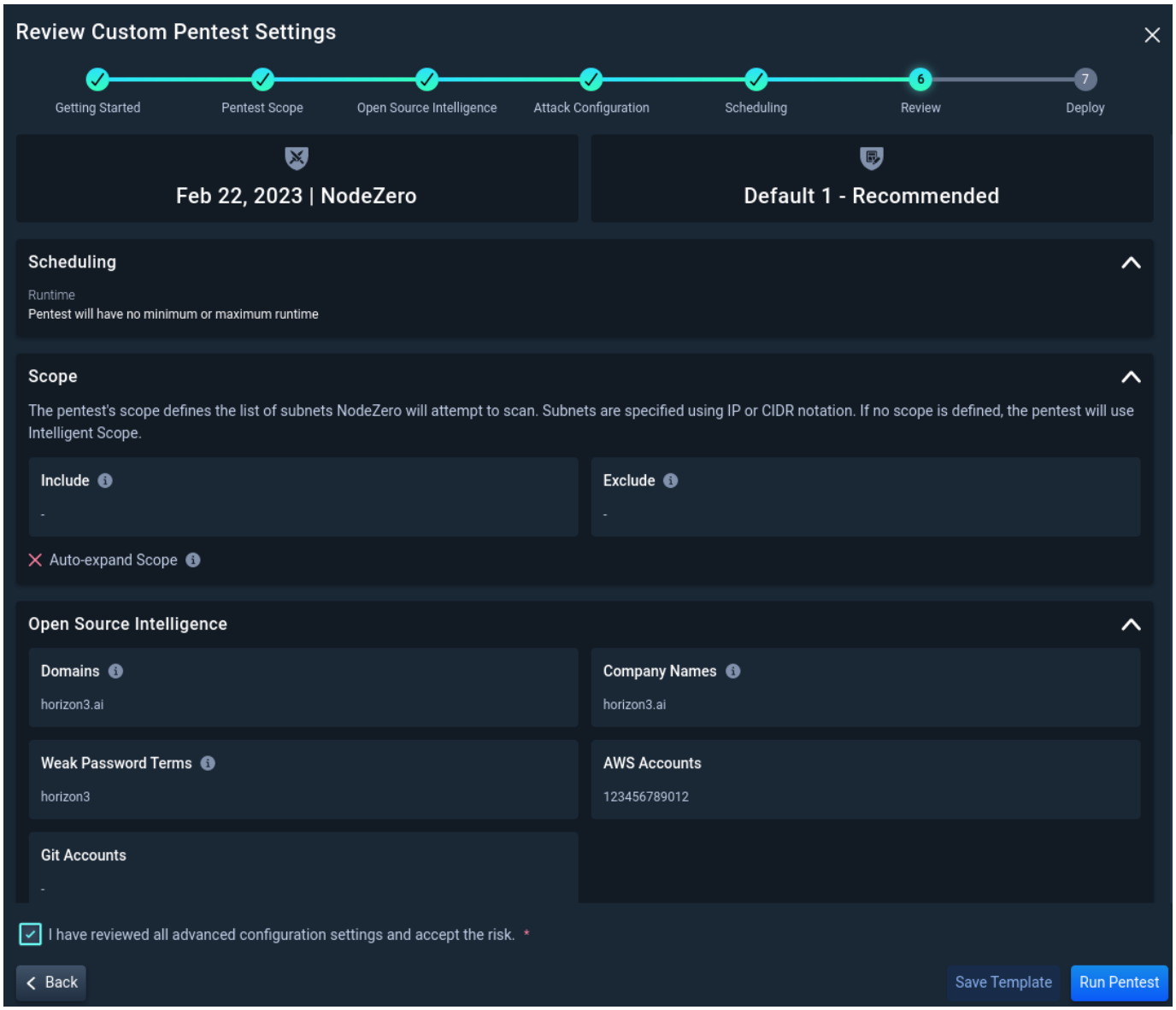
Add a minimum or maximum amount of time to allow some attacks to have more time

Then, click Review.



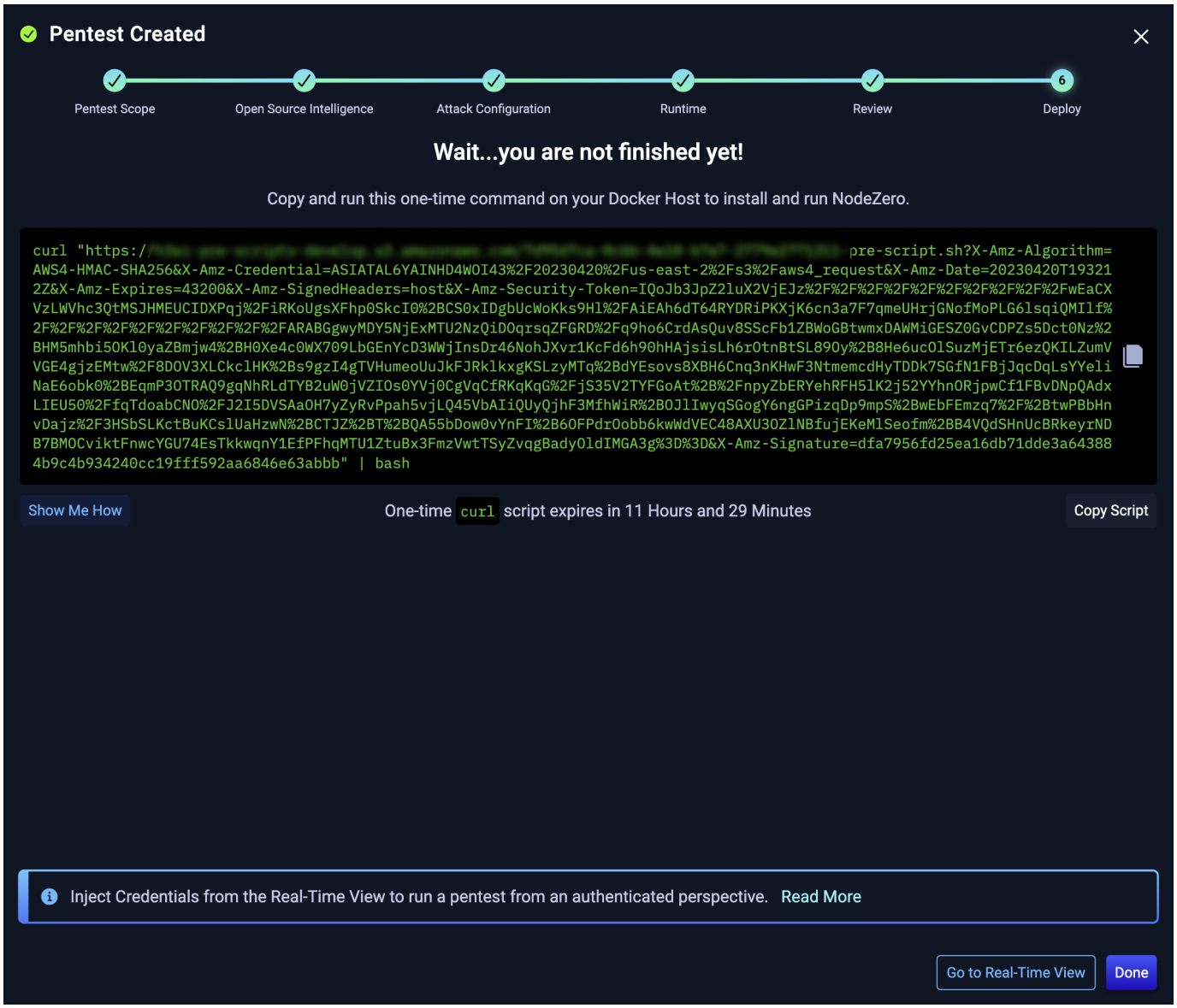
3.6 Review the Internal Pentest Configuration

Once satisfied with your pentest selections, check the box to indicate you've reviewed all advanced configuration settings. Then click Run Pentest, which launches the internal pentest.



4. DEPLOY NODEZERO

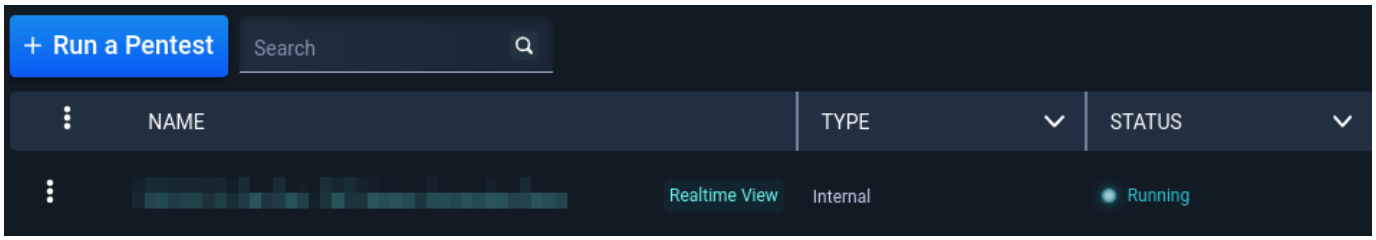
While the pentest is provisioning, its companion one-time-use software module, NodeZero, is made ready for deployment on your NodeZero Host.



Copy the Launch Script and paste it into the shell of your NodeZero Host.

This script will validate the Docker installation, download the most up-to-date NodeZero Docker image, and begin the pentest. In the Portal, you will see the status of the pentest transition from Ready to Running.

Click `Go to Real-Time View` to open the Real-Time View, from where you can Inject Credentials and monitor pentest progress, or click `Copy Script and Close` to copy the script and return to the Pentest Page.



You've started an Internal Pentest

NodeZero sends an email once the Internal Pentest completes.

Tip

NodeZero can also run pentests from an authenticated perspective. Go to the Real-Time View and Inject Credentials to see the impact an attacker would have by leveraging compromised credentials!

2.6 Run an AD Password Audit

2.6.1 AD Password Audit

NodeZero AD Password Audit is an easy way to audit the strength and similarity of user passwords in an Active Directory environment. Read below for motivations, necessary environment configurations, and instructions on running an AD Password Audit.

- Motivations for AD Password Audit
- How to Configure Your Environment
- How to Run an AD Password Audit

Motivations for AD Password Audit

REASONS TO PERFORM A PASSWORD AUDIT

Weak passwords factor into a number of commonly used attacker techniques:

MITRE Tactic	MITRE Technique / Sub-Technique
TA0001: Initial Access	T1078.002: Valid Accounts: Domain Accounts
TA0006: Credential Access	T1110.0001: Brute Force: Password Guessing
TA0006: Credential Access	T1110.0002: Brute Force: Password Cracking
TA0006: Credential Access	T1110.0003: Brute Force: Password Spraying
TA0006: Credential Access	T1110.0004: Brute Force: Credential Stuffing
TA0006: Credential Access	T1621: Multi-Factor Authentication Request Generation

Attackers take advantage of weak passwords for both initial access and lateral movement.

On the perimeter, it's common for attackers to target endpoints not enabled for MFA. For cases that MFA is enabled, having a user's password becomes the first stage to subsequent attacks such as MFA fatigue attacks.

On internal networks, attackers generally don't need to worry about MFA as there are many endpoints and protocols (e.g. SMB, DCOM/RPC, NTLM) available for abuse that don't support MFA.

REASONS TO RUN A PASSWORD AUDIT REGULARLY

It's important to audit passwords regularly because the state of the passwords and accounts in Active Directory changes over time.

- Employees join or leave an organization.
- Employees change their passwords. Many organizations have policies in place that require employees to rotate their passwords.
- New data breaches in the wild can expose passwords previously thought to be safe.

Audit passwords regularly

We recommend running an AD Password Audit at least once every three (3) months.

How to Configure Your Environment

This section reviews the prerequisites to running an AD Password Audit. Instructions are given for configuring your environment to ensure the audit can run successfully.

Already configured your environment?

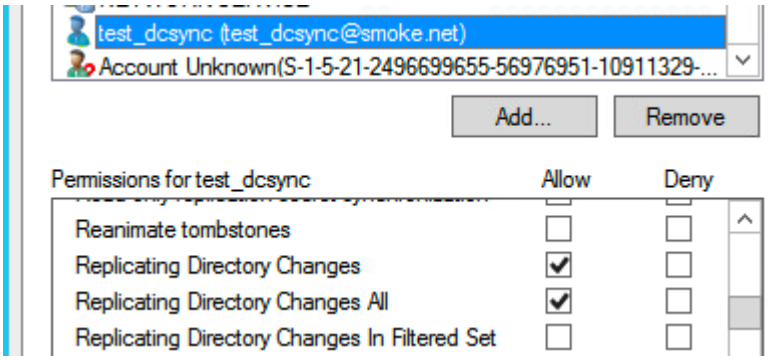
Jump to the section on How to Run an AD Password Audit.

PRIVILEGED ACCOUNT

Prior to running an AD Password Audit, you must have the credentials for a domain user account with DCSync privileges. A domain admin account can be used, but we recommend setting up a separate account with the minimum privileges required.

The minimum privileges required are:

```
Replicating Directory Changes
Replicating Directory Changes All
```



CONFIGURING EDR TO ALLOW DCSYNC

EDR software running on the domain controller may block DCSync from taking place. This is a good thing in general as it prevents attackers from using this well-known attack technique. However, for the sake of the AD Password Audit, DCSync needs to be allowed. We recommend consulting your EDR vendor's documentation for instructions on how to adjust the EDR configuration to enable DCSync. Once you finish running an AD Password Audit, the configuration should be reset to block DCSync again.

TESTING DCSYNC

You can manually verify that DCSync will work using the NodeZero Docker container. On a host where you have already run a NodeZero pentest before, start up an old NodeZero container.

First run `docker ps -a` to list any existing containers.

```
[root@nodezero [ ~ ]# docker ps -a
CONTAINER ID   IMAGE                                                                COMMAND                  CREATED        STATUS                      PORTS     NAMES
0acc5a87e919   206961115674.dkr.ecr.us-east-2.amazonaws.com/h3/n0:naveen1729.latest   "/docker-entrypoint.…"   28 hours ago   Exited (0) 27 hours ago             n0-8836
root@nodezero [ ~ ]#
```

Then run `docker start n0-xxxx` to start an existing container (substitute `n0-xxxx` for your container's name).

```
[root@nodezero [ ~ ]# docker start n0-8836
n0-8836
[root@nodezero [ ~ ]#
```

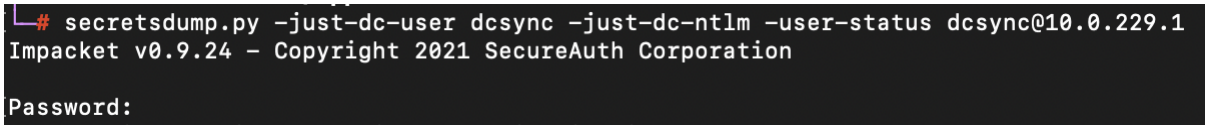
Then run `docker exec -it n0-xxxx` bash to get a shell inside the container.

```
[root@nodezero [ ~ ]# docker exec -it n0-8836 bash
┌──(root㉿nodezero)-[/app]
└─#
```

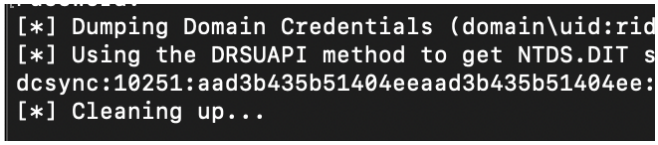
Next, using the privileged account from the previous step, run the `secretsdump` command to perform a DCSync operation (substitute in your account's username and domain controller IP).

```
secretsdump.py -just-dc-user <your_dcsync_user> -just-dc-ntlm -user-status <your_dcsync_user>@<domain_controller_ip>
```

Enter the password for the privileged account.



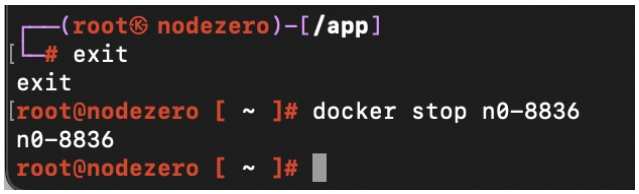
If the DCSync is successful you should see a successful retrieval of the NTLM hash for just the privileged account.



If the DCSync is not successful, you should recheck your account's permissions and EDR configuration.

Finally, after testing, exit the shell and stop the NodeZero container. Note, even if you don't shut it down, the container will automatically shut itself down after ten minutes.

```
docker stop n0-xxxx
```



How to Run an AD Password Audit

1. NAVIGATE TO PENTESTS

Navigate to Pentests to start a pentest.

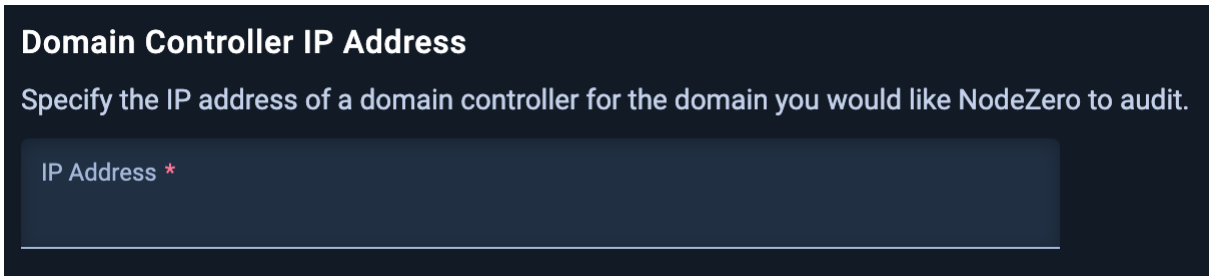
2. CLICK `+ RUN PENTEST`

Click `+ RUN PENTEST` to open the Pentest Configuration and select `AD Password Audit`

3. CONFIGURE THE AD PASSWORD AUDIT

3.1 Domain Controller IP Address

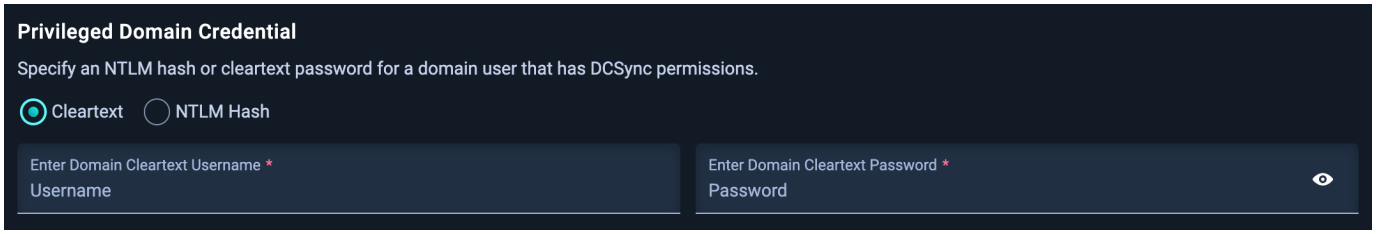
Specify the IP address of a domain controller for the domain you would like NodeZero to audit.



> **Note**
>
> Make sure the IP Address is accessible from where your docker host is deployed.

3.2 Privileged Credential

Specify an NTLM hash or cleartext password for a domain user that has DCSync permissions.



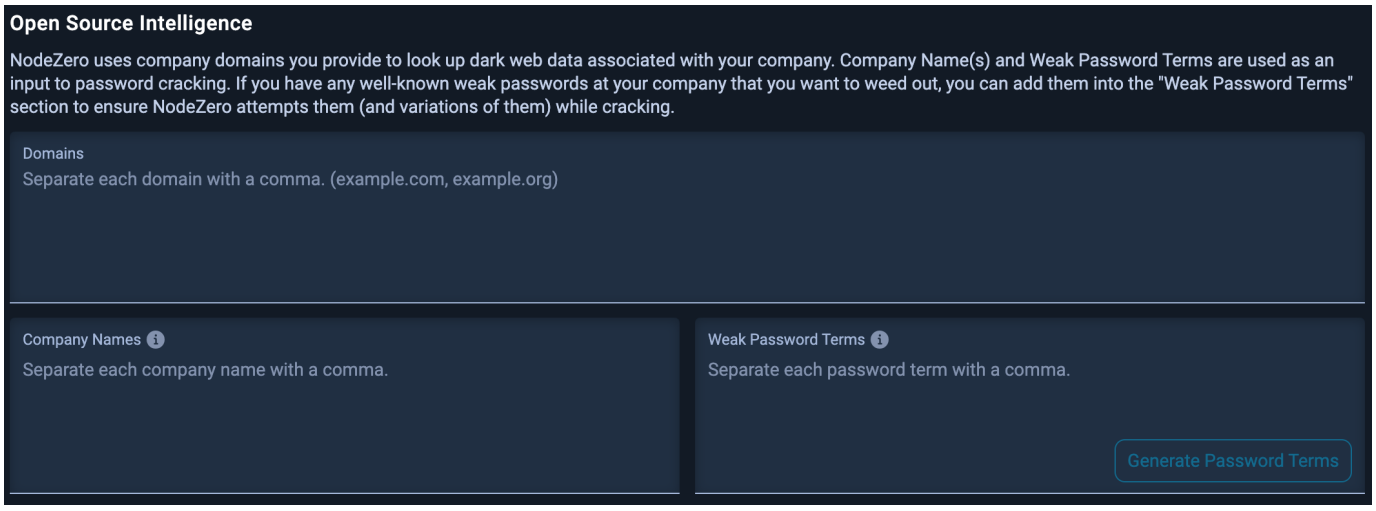
DCSync permissions

NodeZero needs DCSync permissions in order to extract the NTDS.dit file from the domain controller. The NTDS.dit file stores Active Directory information about user objects and includes the NTLM hashes for all the users. These NTLM hashes are what NodeZero will attempt to crack. For more information on setting up a credential with the needed permissions, see How to Configure Your Environment

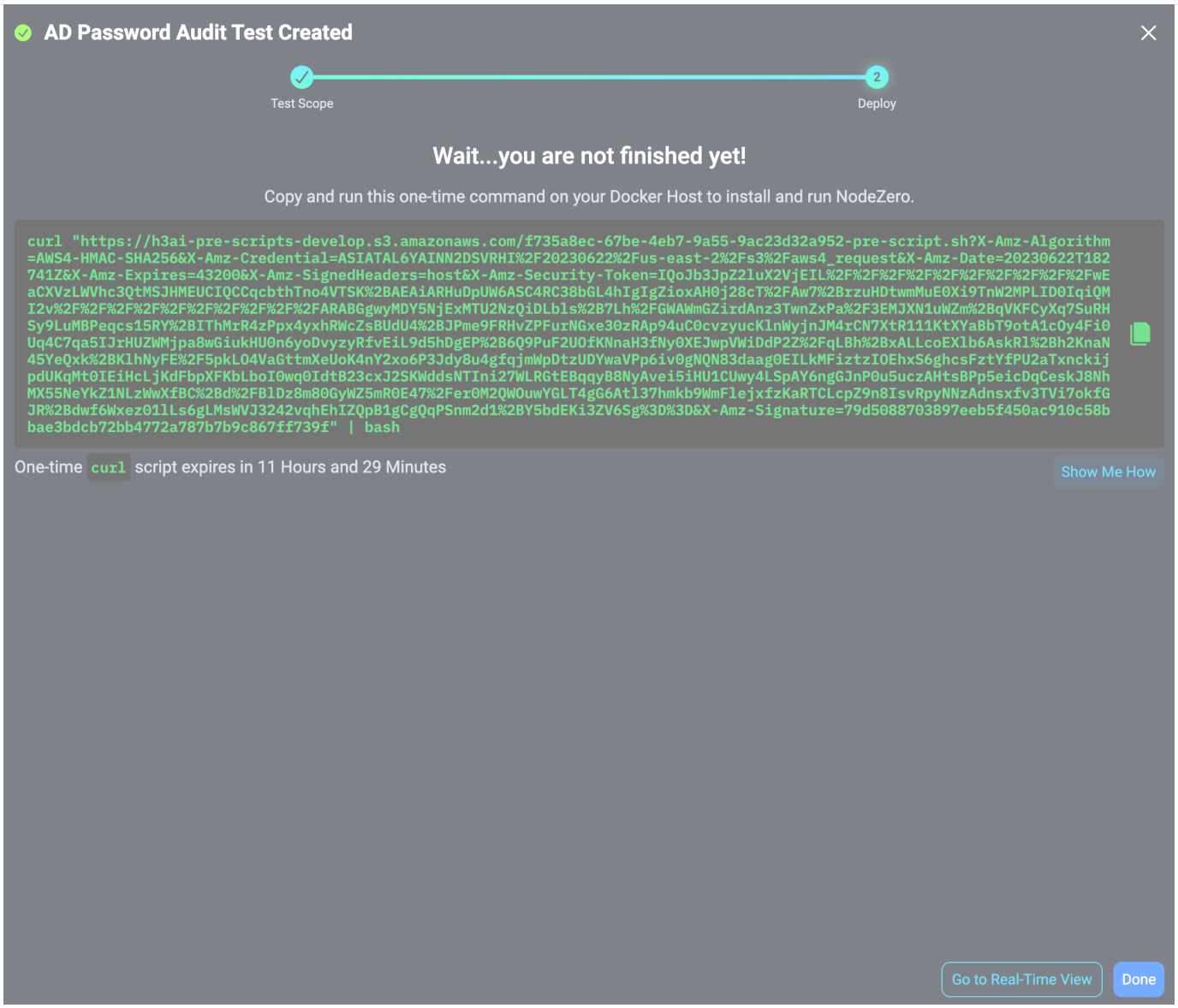
3.3 Add Open-Source Intelligence

The Open Source Intelligence step is important for providing NodeZero context about your company - context that an attacker may be able to leverage to guess weak passwords.

NodeZero uses company domains you provide to look up dark web data associated with your company. Company Name(s) and Weak Password Terms are used as an input to password cracking. If you have any well-known weak passwords at your company that you want to weed out, you can add them into the “Weak Password Terms” section to ensure NodeZero attempts them (and variations of them) while cracking.



4. DEPLOY NODEZERO



Run the curl script on your NodeZero host to kick off the AD Password Audit. Or, if you are using a NodeZero runner, the audit will kick off automatically on its own.

Don't have a NodeZero host?

See our documentation in Setup NodeZero Host.

This script will validate the Docker installation, download the most up-to-date NodeZero Docker image, and begin the pentest. In the Portal, you will see the status of the pentest transition from Ready to Running.

Click `Go to Real-Time View` to open the Real-Time View to monitor pentest progress, or click `Copy Script and Close` to copy the script and return to the Pentest Page.

Successful launch!

Upon completion of the AD Password Audit, NodeZero sends an email to the user that launched it.

No passwords found?

If the AD Password Audit fails to display any data upon its completion, verify you have properly configured the injected credential with privileged access. For instructions on configuring the credential, see How to Configure Your Environment.

Learn how NodeZero works

Keep reading to learn how NodeZero conducts an AD Password Audit.

How NodeZero Conducts an AD Password Audit

User passwords are stored as NTLM hashes within the Active Directory database file, called NTDS.dit, on domain controllers. To conduct an AD Password Audit, the NodeZero Docker container first performs a remote operation called DCSync to connect to a domain controller and pull these password hashes. NodeZero requires a privileged credential to perform this operation.

User password hashes are transmitted by the NodeZero Docker container to the Horizon3.ai cloud and stored within a secured ephemeral private VPC network for the duration of the audit. The VPC is allocated in real-time when the audit is started and destroyed after the audit completes. All sensitive data, including password hashes, are destroyed at the end of the audit.

PASSWORD CRACKING

During the audit, NodeZero attempts to crack password hashes using a variety of methods:

- ✅hecking for basic commonly used weak passwords
- ✅hecking for passwords that resemble usernames
- ✅hecking for passwords that resemble well-known contextual terms about a company such as a company's name or domain name
- ✅hecking for passwords that match or are similar to passwords in dark web data associated with the company
- ✅hecking for passwords that match or are similar to any known breached passwords
- ✅hecking for passwords that match any user-provided custom password terms

PASSWORD REUSE

NodeZero also analyzes the population of passwords for password reuse, i.e. cases where two or more users are using the exact same or similar passwords. Password reuse is important to minimize. When users use similar passwords, attackers can easily take a single account compromise and turn it into a multi-account compromise, potentially gaining more privileges and access to data.

REPORTS

After completing the AD Password Audit, the Fix Action Report can be downloaded by clicking on the download button in the upper-right corner of the summary page.

VIEWING DISABLED USERS

To see disabled users in the AD Password Audit results, go to the "AD Password Audit" page. Click on the 'kebab' menu (three vertical dots) located in the upper-left corner of the results table. From the dropdown menu, select 'View Disabled Users'.

References

1. Metrics That Matter: An Attacker's Perspective on Assessing Password Policy
2. The Undeniable Effectiveness of Password Spray

3. NIST Special Publication 800-63B

2.7 Run 1-click Verify

2.7.1 Run 1-click verify

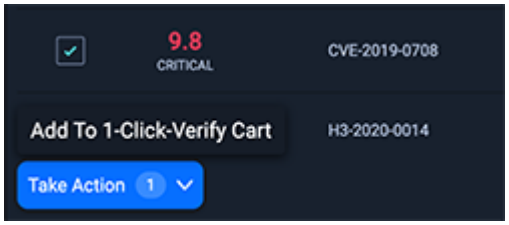
1-click verify enables you to quickly and conveniently schedule a pentest to verify that specific weaknesses have been remediated. This supports and helps to facilitate the Find-Fix-Verify loop.

How does it work?

1. SELECT THE WEAKNESSES AND ADD THEM TO THE CART

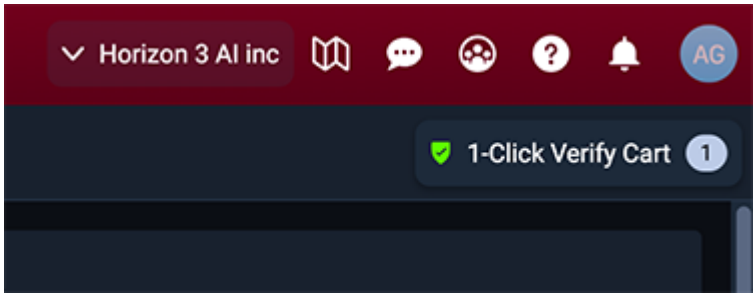
Under the Weaknesses tab, we've added a 1-click verify functionality to all weaknesses that are 1-click verifiable. Note that not all weaknesses are 1-click verifiable. 1-click verifiable weaknesses are those that can be reliably verified even when the pentest scope is limited to only a single host or set of hosts affected by the weakness. Weaknesses that require complex chains involving multiple hosts and credentials would not be considered 1-click verifiable.

If a weakness is 1-click verifiable, you will see a checkbox on the row, and when you check it, you will notice that the `Take Action` button on the bottom of the page is now clickable. When you click on the button, you will see a sub menu item called `Add To 1-Click-Verify Cart`.



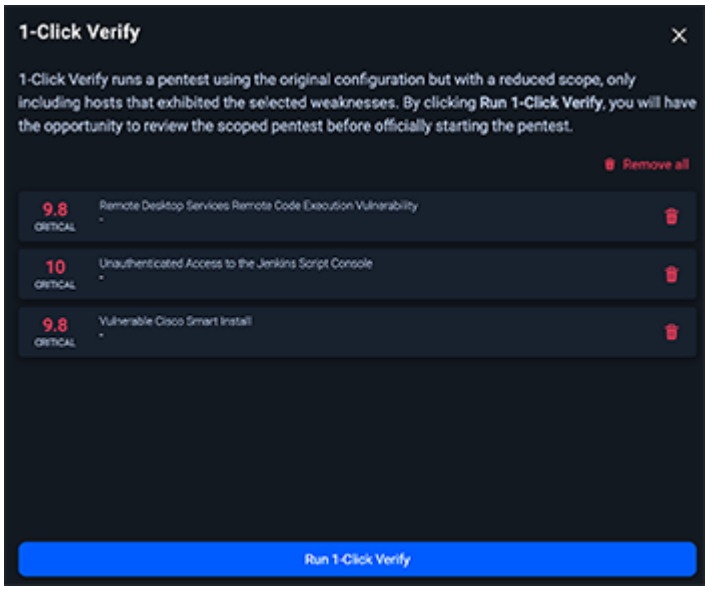
2. VERIFY THE CART CONTENTS

When you click on the sub menu to add the weakness to the cart, you will see the `1-Click Verify Cart` showing up on the top right corner of the page.



You can verify the content of the cart and remove any unnecessary weaknesses, or cleanup all and start over.

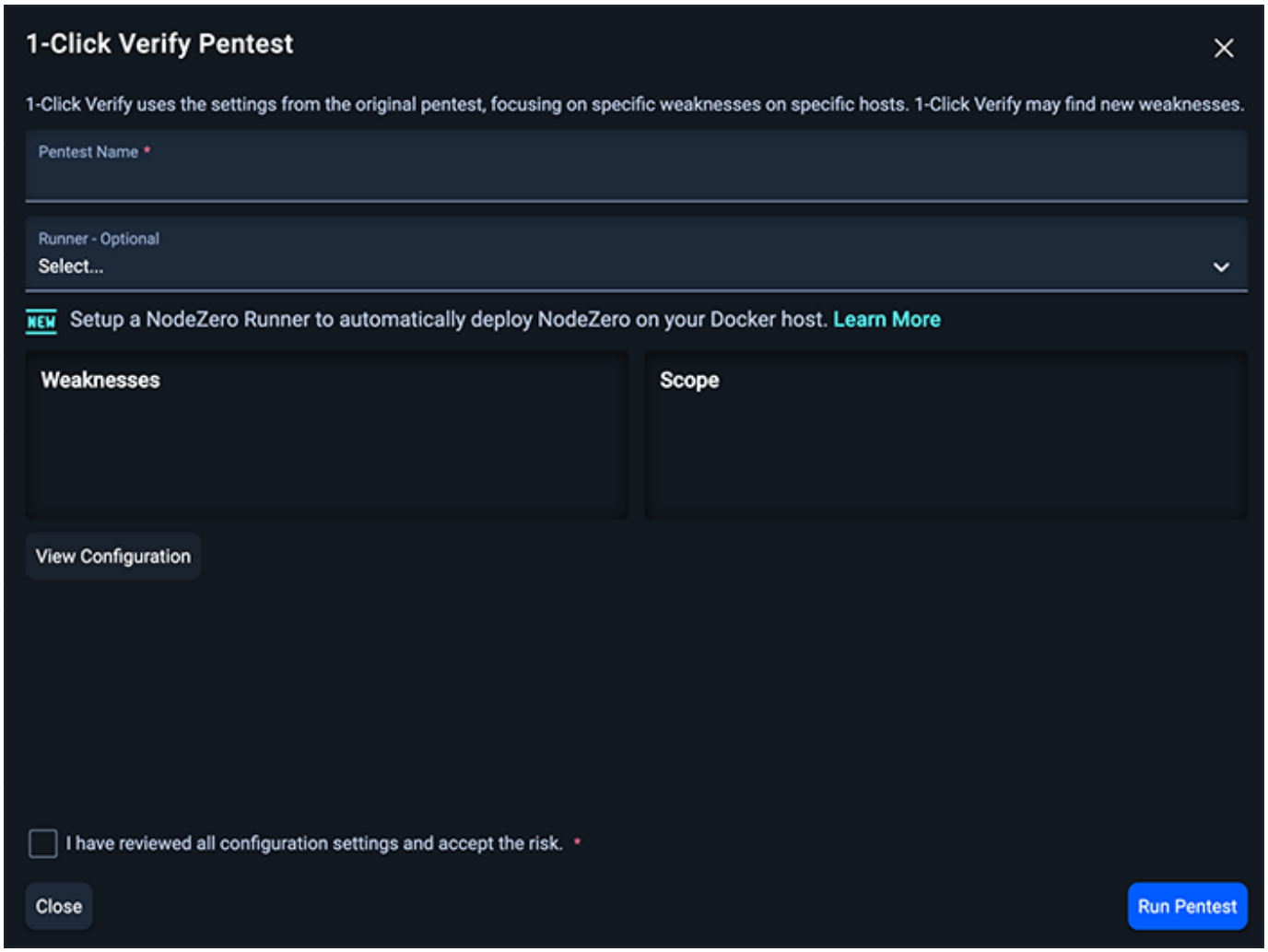
Once you are ready, click the `Run 1-Click Verify` button at the bottom of the page, and you'll be presented with the pentest-scheduling popup.



3. RUN 1-CLICK VERIFY PENTEST

The pentest scheduling popup is pre-populated with a scope that includes only the hosts affected by the weakness. All other configuration settings are copied from the original pentest containing the weakness being verified.

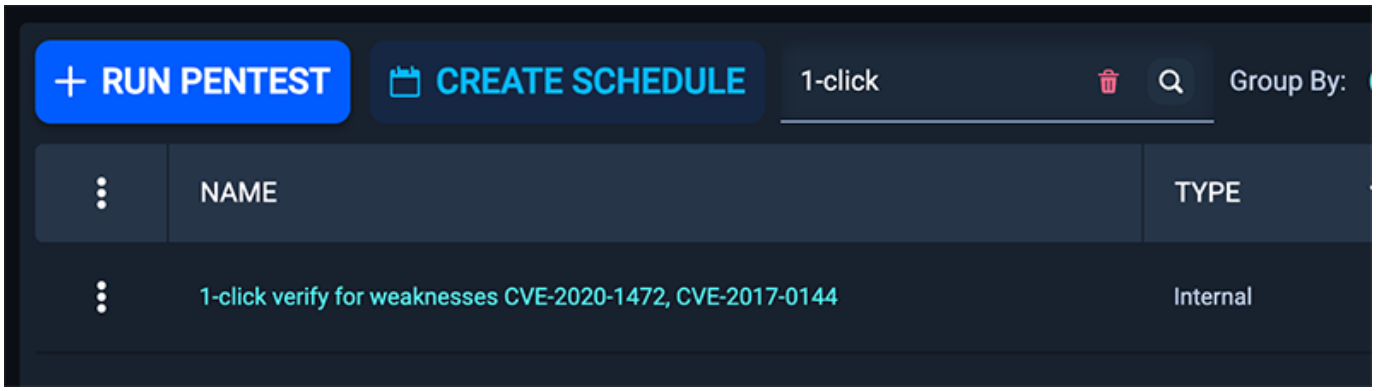
In this popup window, you have the option of scheduling the pentest as-is, or you can update the pentest name, or optionally choose a runner. You can also see the existing configuration by clicking the `View Configuration` button. From here, make sure that you check the checkbox that reads `I have reviewed all configuration settings and accept the risk.` and click on the `Run Pentest` button.



4. CHECKING THE RESULTS

The newly scheduled 1-click verify pentest appears in the pentest list with its own record. Its name defaults to `1-click verify for weakness CVE-XXXX-XXXX`, unless you changed it during the previous step.

When the pentest completes, the weakness being verified will (ideally) not appear in the output. This most likely means the user has successfully remediated. However, the only thing we can claim with certainty is that we did not find the weakness.



Why do you need it?

An ideal use case is that you first run a pentest against your entire network and find perhaps many weaknesses across dozens or even hundreds of hosts. Depending on the size of their network, the pentest might take a while to run (e.g., a

pentest against 5000+ hosts might take a week). You might remediate one or two weaknesses and wish to run a second pentest to verify. With 1-click verify, you can narrow the scope of the second pentest to the hosts affected by the weakness you wish to verify. Given the narrowed scope, the second pentest is likely to complete much more quickly, perhaps within an hour (all depends on the size of the scope). This enables a quick verification of your remediation efforts and fuels the Find-Fix-Verify loop.

What problem does it solve?

Rapid, targeted verification of your remediation efforts enables you to make incremental and verifiable progress toward the goal of eliminating weaknesses and improving the security posture of your network.

1-click verify can be a fast-path to helping you verify and document the corrections you have made during a compliance audit, such as the Payment Card Industry Data Security Standard. Once you have verified that your remediation is complete, simply download the 1-click verify report as evidence to submit to your assessor.

2.8 NodeZero Portal Access Control

2.8.1 NodeZero User Access and Control

Portal Login Types

> **Note**
>
> NodeZero Web UI will be commonly referred to as simply Portal.

The NodeZero Portal supports the following login types.

- **Username and Password** (MFA required)
- **Social Logins**:
- Google
- LinkedIn
- Microsoft
- Azure AD
- Office365 (Azure AD is the backend)
- Free Microsoft accounts
- **Private Single Sign-On (SSO)** (Beta)

User Management

- **Setting User Permissions** NodeZero allows users to be added, edited, removed and have their permissions set and changed

2.8.2 User Personas

The following access control pages provide instructions from the perspective of two user personas: `Portal Org Admin` and `Identity Team Admin`.

User Personas

Portal Org Admin - User with the Org Admin role in Portal.

Identity Team Admin - An admin at your company with access to make changes to your identity provider

Each section with access control setup instructions should start with a heading like the ones below. This will help identity the correct persona that needs to take action to enable the described functionality.

Identity Team Admin Persona

Portal Org Admin Persona

If all sections on a page are written for a single user persona, you should see a single heading at the top of the page similar to the one below instead of a heading per section.

All sections of this page should be completed by someone with the Identity Team Admin Persona

It is possible for these user personas to be the same person, but typically they are different people.

2.8.3 User Management

> All sections of this page should be completed by someone with the Portal Org Admin Persona

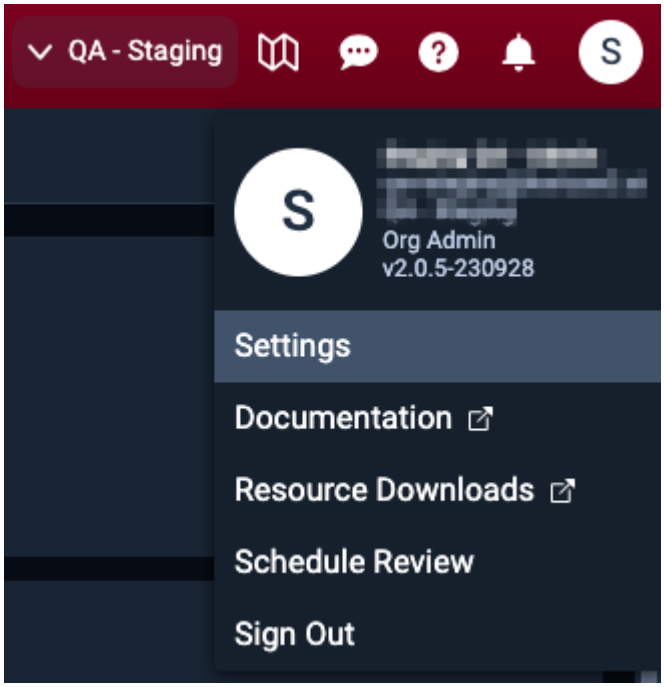
A new user must be invited to a company account. If a user accesses Portal via social login or private SSO without an invite, a personal, read-only account is created. This personal account does not have access to any company accounts by default.

A user with Portal Org Admin access can send an invite to allow the user to switch from their personal account to their company account. The company account will become the default for all future logins for that user.

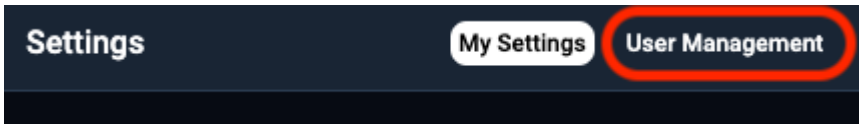
User Management Settings

Users with Portal OrgAdmin access are able to manage users and SSO provider settings via the User Management menu.

To access the User Management menu, navigate to `Settings` by clicking the user profile button in the top right of Portal.

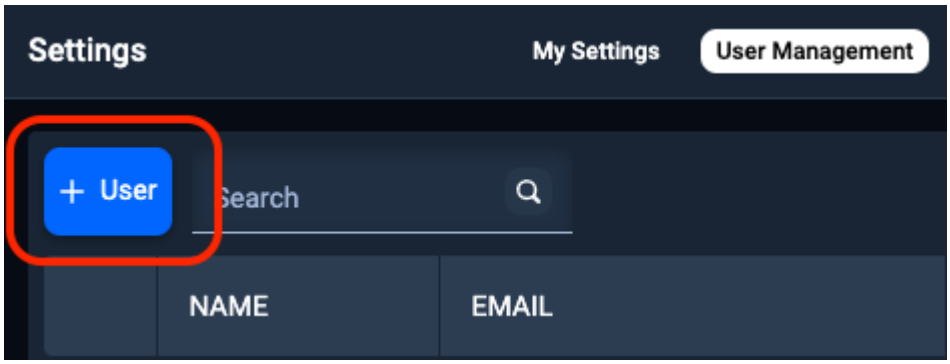


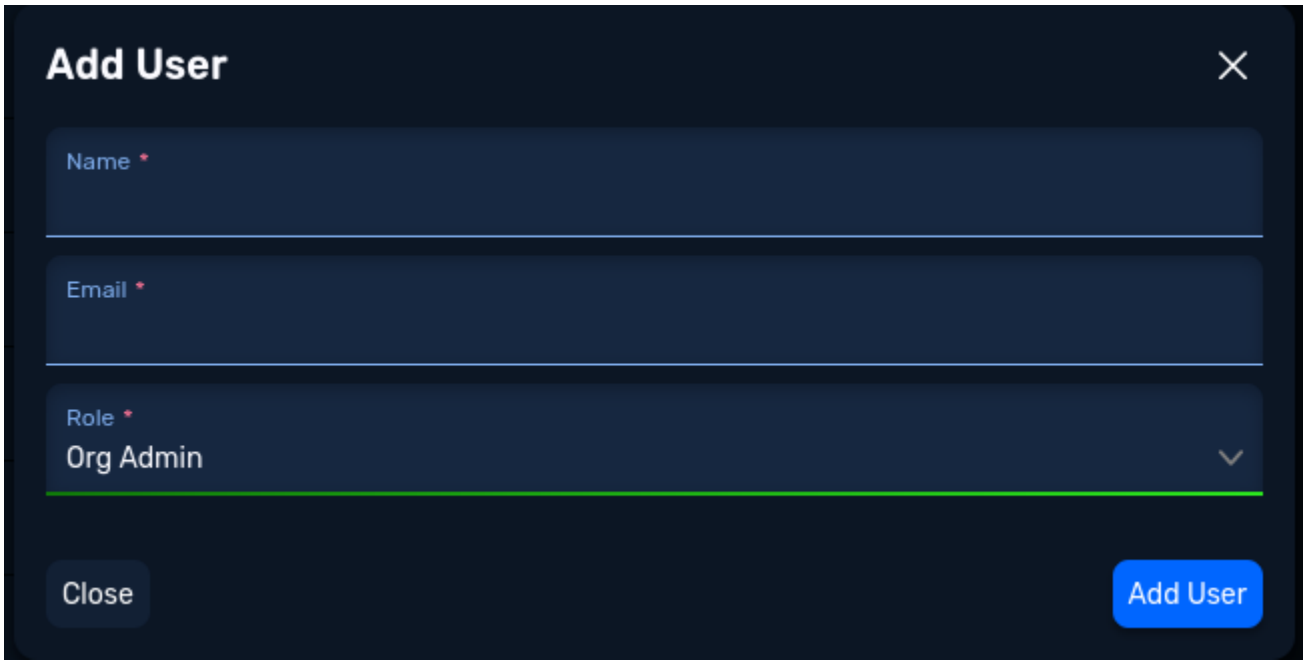
Then click `User Management`.



Add Users

Click the add user button to add users to your company account. All fields are required. Pay careful attention to the `Role` field to ensure you're selecting the appropriate level of Portal access.





Portal Access Roles

The available Portal access roles are:

Org Admin - Full access to the Company Account.

User - Access to run, schedule, and view pentests.

Readonly - Access to only view previous pentest results.

Portal Org Admins will still retain their username/password access to Portal even if SSO Only is enabled for the company account they belong to. If it is desired/required that Org Admins can only login via SSO and not username/password, please reach out to your customer success representative.

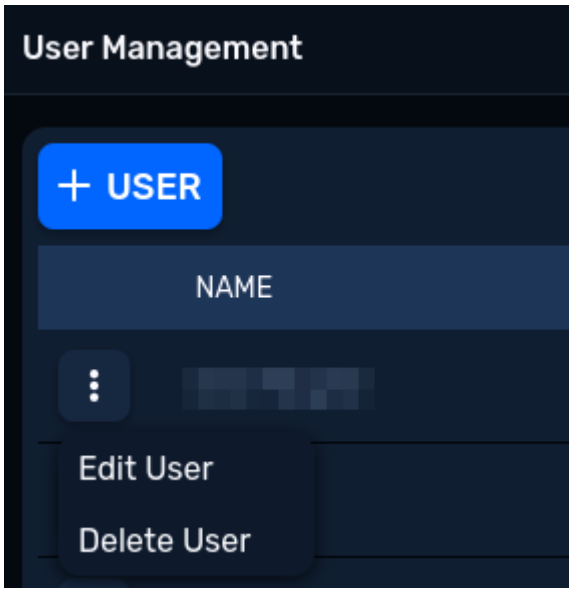
User Notifications

When a user is added without SSO, they will receive two emails to the email address provided in the Add User form: a welcome email and an email with temporary credentials to complete their account registration.

If SSO Only is enabled for a company account, the user will receive a single welcome email directing them to follow whatever login processes their company has in place to log in to Portal via their identity provider.

Edit or Delete Users

Existing users can be edited or deleted by clicking the vertical ellipsis to the left of the user's name.



Then selecting the appropriate action: `Edit User` or `Delete User`.

2.8.4 Single Sign-On (SSO)

Beta

If you encounter issues please contact your CS rep.
User experience is subject to change.

Supported Identity Providers

Portal supports OpenID Connect (OIDC) compatible identity providers (IdP). This includes but is not limited to: Okta, Keycloak, Ping, Google, and Azure AD. Any IdP that is OIDC compliant should be compatible.

Once SSO is enabled, users have two ways to login. They can login at the Portal by selecting "Continue with Private SSO" or they can can login via their IdP's application dashboard (Okta User Home, PingOne Application Portal, Microsoft My Applications, etc.).

General Requirements | US Portal Specifics | EU Portal Specifics

- OpenID Compatible Identity Provider
- Scopes: oidc (default), email, profile
- Grant Type: Authorization Code
- Authorization Type: POST

Field	Value
Sign-in redirect URIs	https://portal.horizon3ai.com https://auth.horizon3ai.com/oauth2/idpresponse

Field	Value
Sign-in redirect URIs	https://portal.horizon3ai.eu https://auth.horizon3ai.eu/oauth2/idpresponse

Enable Single Sign-On

The following sections provide the steps for setting up the SSO Provider for a Company Account. Note that some sections need to be followed by the Portal Org Admin and others by the Identity Team Admin.

BUILD IDP APP

Identity Team Admin Persona

Check out our Identity Provider Guides section for some helpful direction on how to build and configure your app.

For IdPs not covered in our guides, we'll need the below outputs at a minimum. Once the app is built, provide the Portal Org Admin who will be creating the SSO Provider in Portal with the following information:

Generic OpenID Provider **Okta** **Azure**

- Client ID
- Secret ID
- Issuer URL - Your IdP documentation should point you to the correct path. Typically `https://<yourIdPdomain>/<tenantID>/.well-known/openid-configuration`.

- Client ID
- Secret ID
- Issuer URL (`https://your_company_domain.okta.com/.well-known/openid-configuration`)

- Client ID
- Secret ID
- Issuer URL (`https://login.microsoftonline.com/<uuid>/v2.0/.well-known/openid-configuration`)

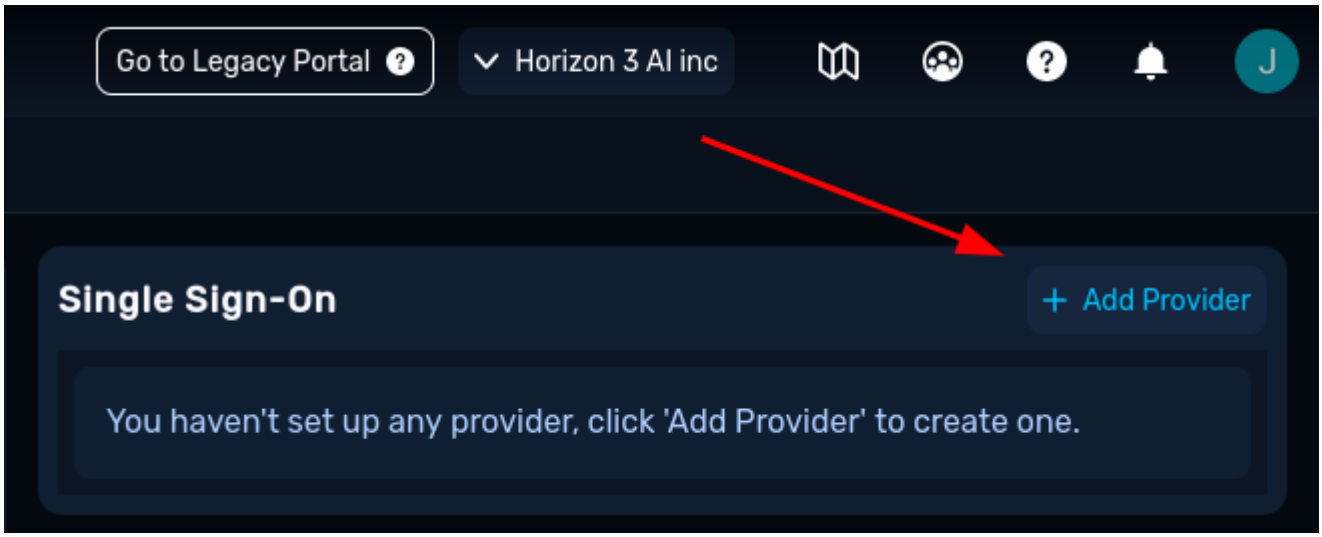
CREATE SSO PROVIDER IN PORTAL

Portal Org Admin Persona

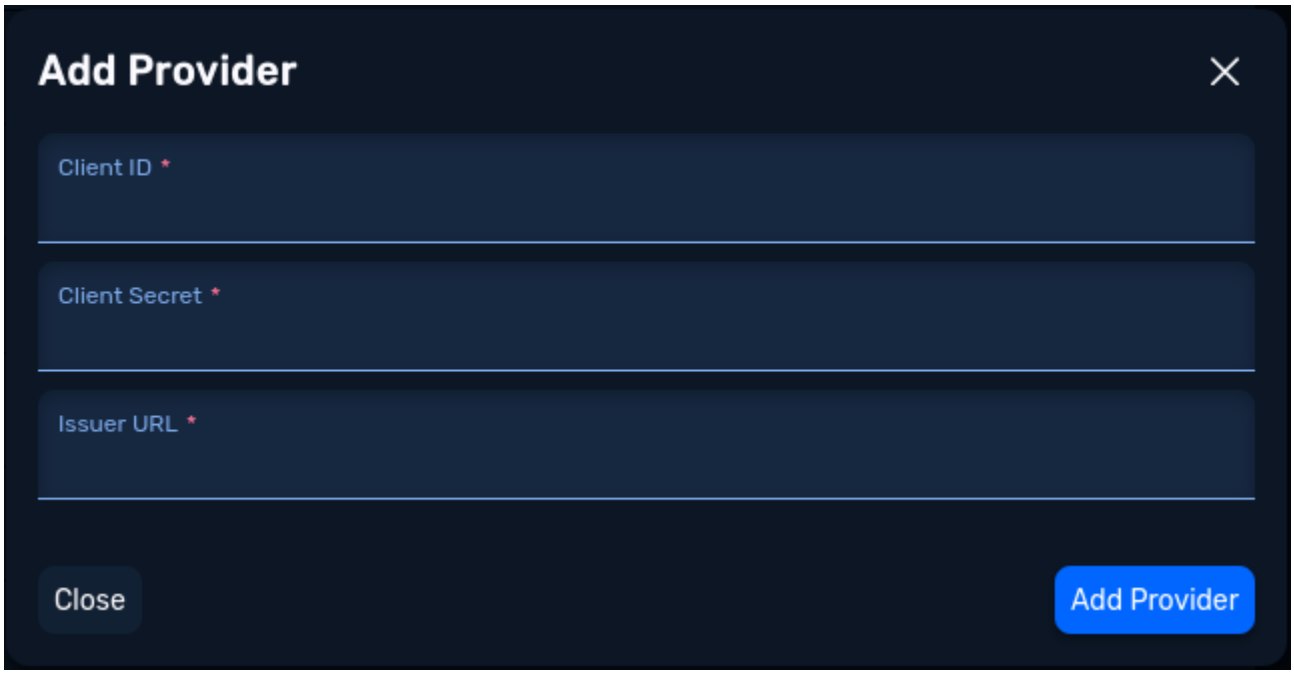
Navigate to the User Management page. To access the User Management menu, navigate to `Settings` by clicking the user profile button in the top right of Portal.

Then click `User Management`.

Click the `Add Provider` button.



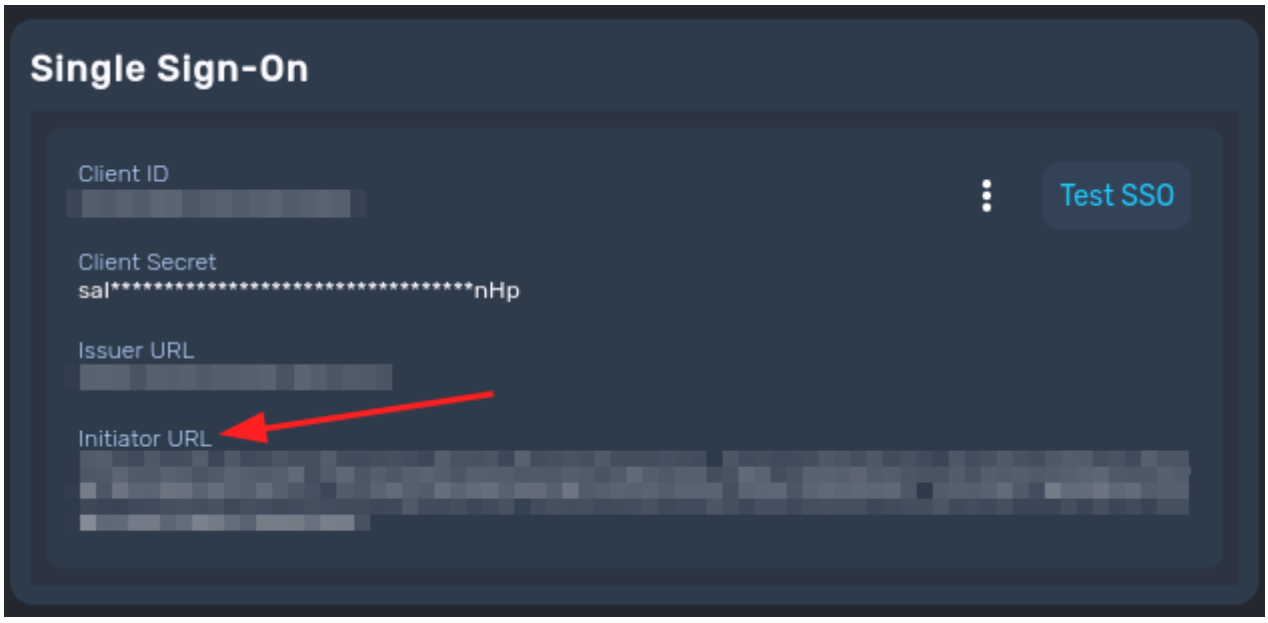
Populate the Add Provider form with the details provided by your Identity Team Admin from the Build IdP App section.



An Initiator URL value will be provided in Portal when the SSO Provider has finished updating. Test your SSO configuration and then provide this value to your Identity Team Admin.

Note

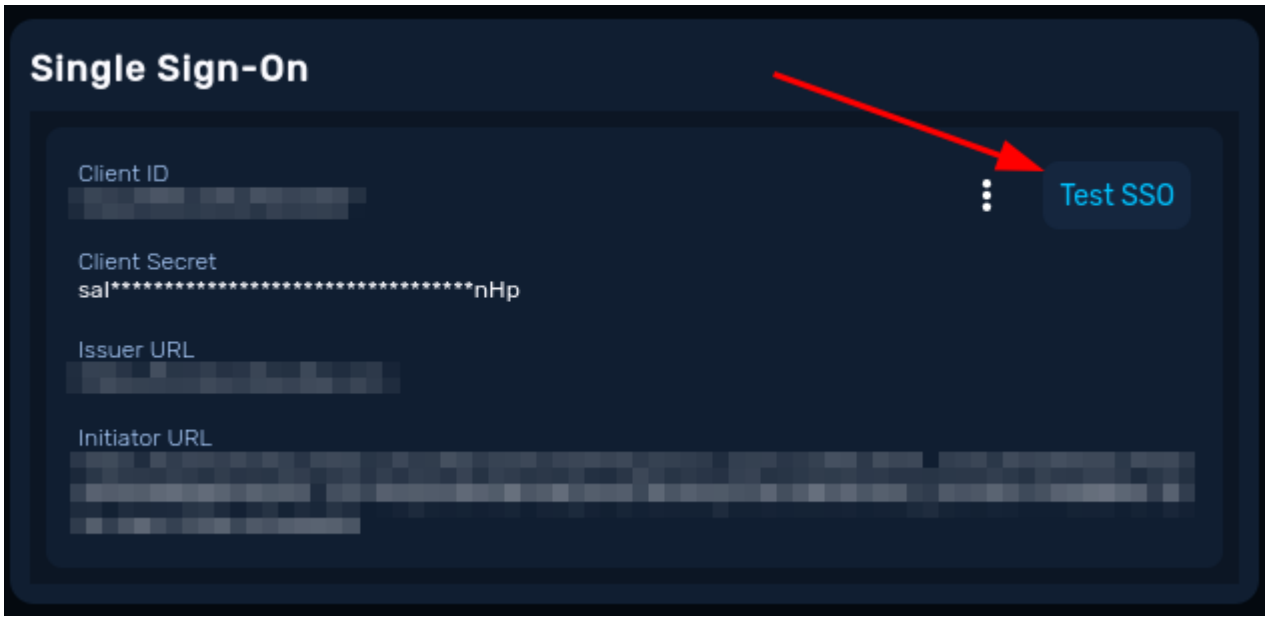
It can take 30 - 60 seconds for the SSO Provider to return the Initiator URL.



TEST SSO CONFIGURATION

Portal Org Admin Persona

The `Test SSO` button allows Org Admins to test the SSO configuration prior to sending the Initiator URL value to their Identity Team Admin.



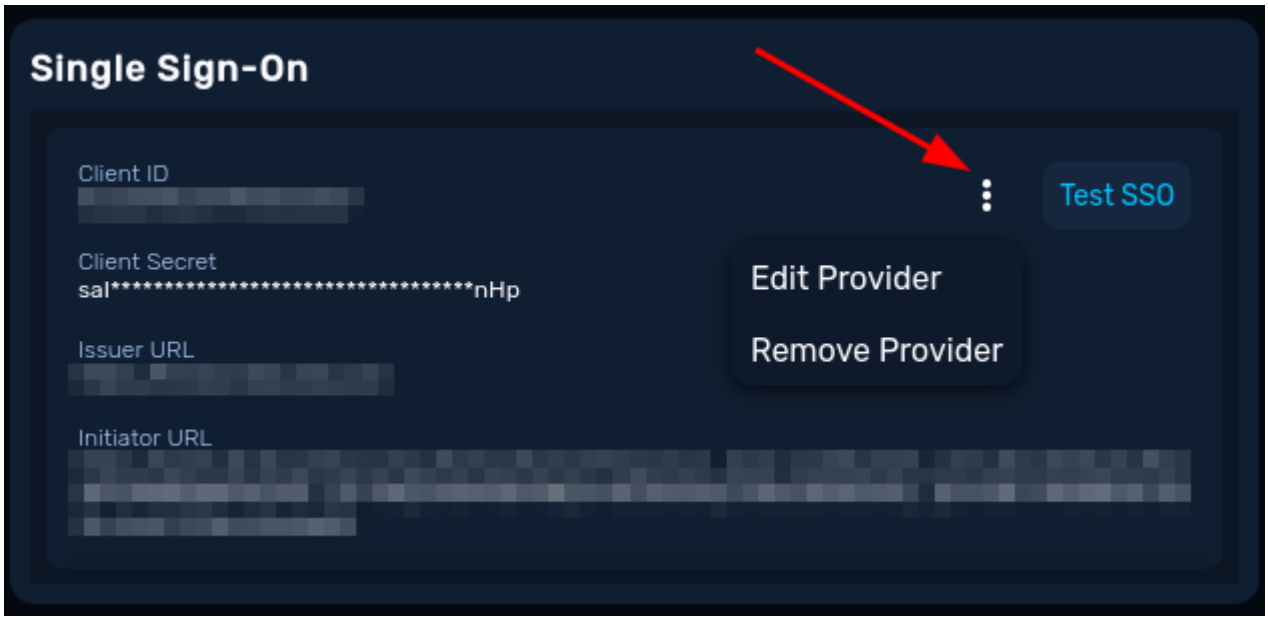
CONFIGURE PORTAL ACCESS VIA IDP APP

Identity Team Admin Persona

Add the Initiator URL value as the Initiate Login URI to your IdP app to allow your users to access Portal via your IdP application.

Edit/Delete SSO Provider

The SSO Provider can be deleted by clicking the vertical ellipsis in the Single Sign-On section



2.8.5 Identity Provider Guides

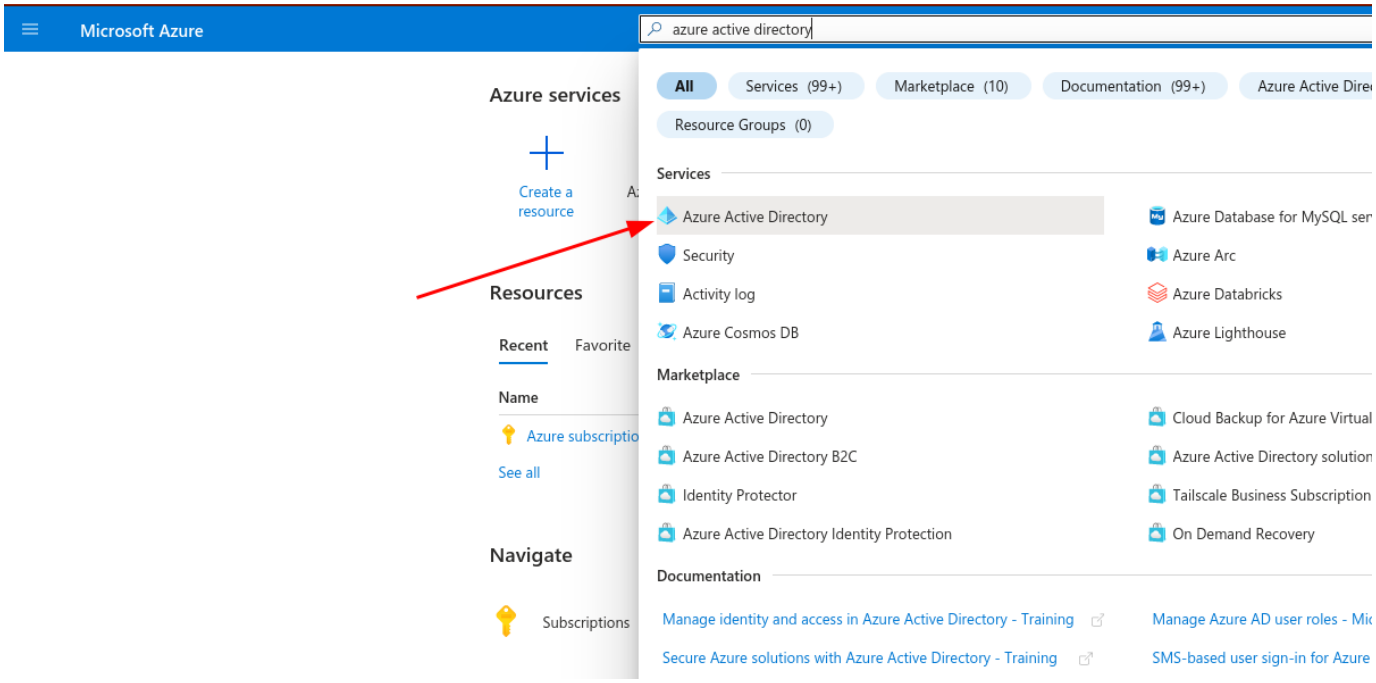
Azure

> **Warning**
>
> This guide should be used as a functional example only. Identity Admins should follow their Company's policies and best practices when implementing Single Sign-On (SSO).
>
> Similarly, because these guides are for services Horizon3 does not control, screenshots and configuration options may be different then what you see here.

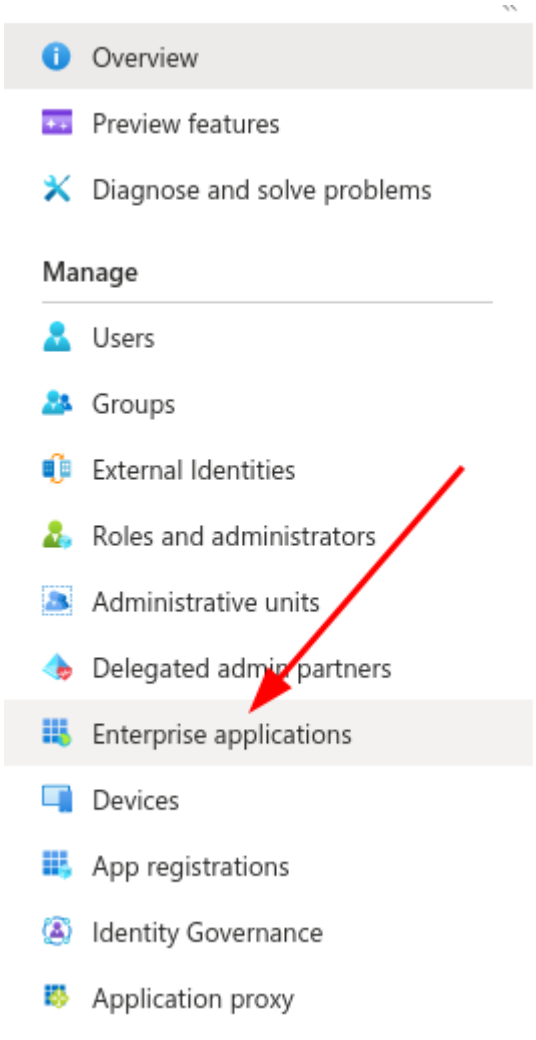
All sections of this page should be completed by someone with the Identity Team Admin Persona

CREATE AZURE ENTERPRISE APPLICATION

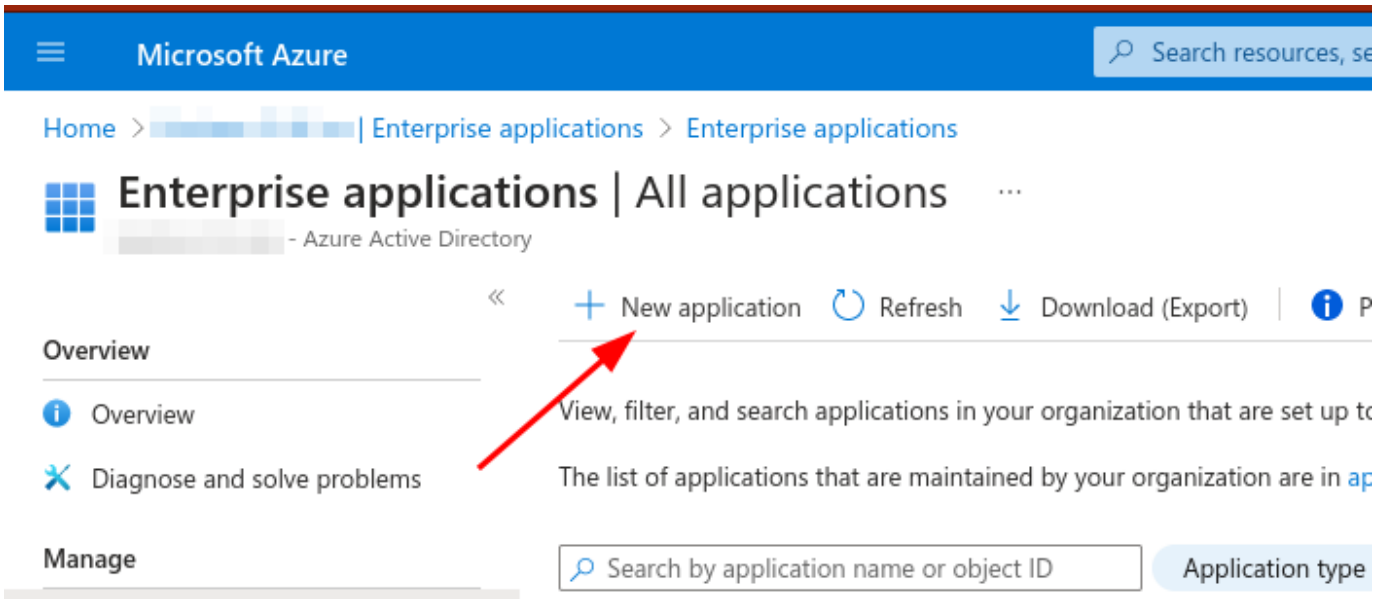
Log into Azure Portal and browse to the "Azure Active Directory" service.



In the left hand menu under the "Manage" section, click "Enterprise applications".



Then click "New Application".

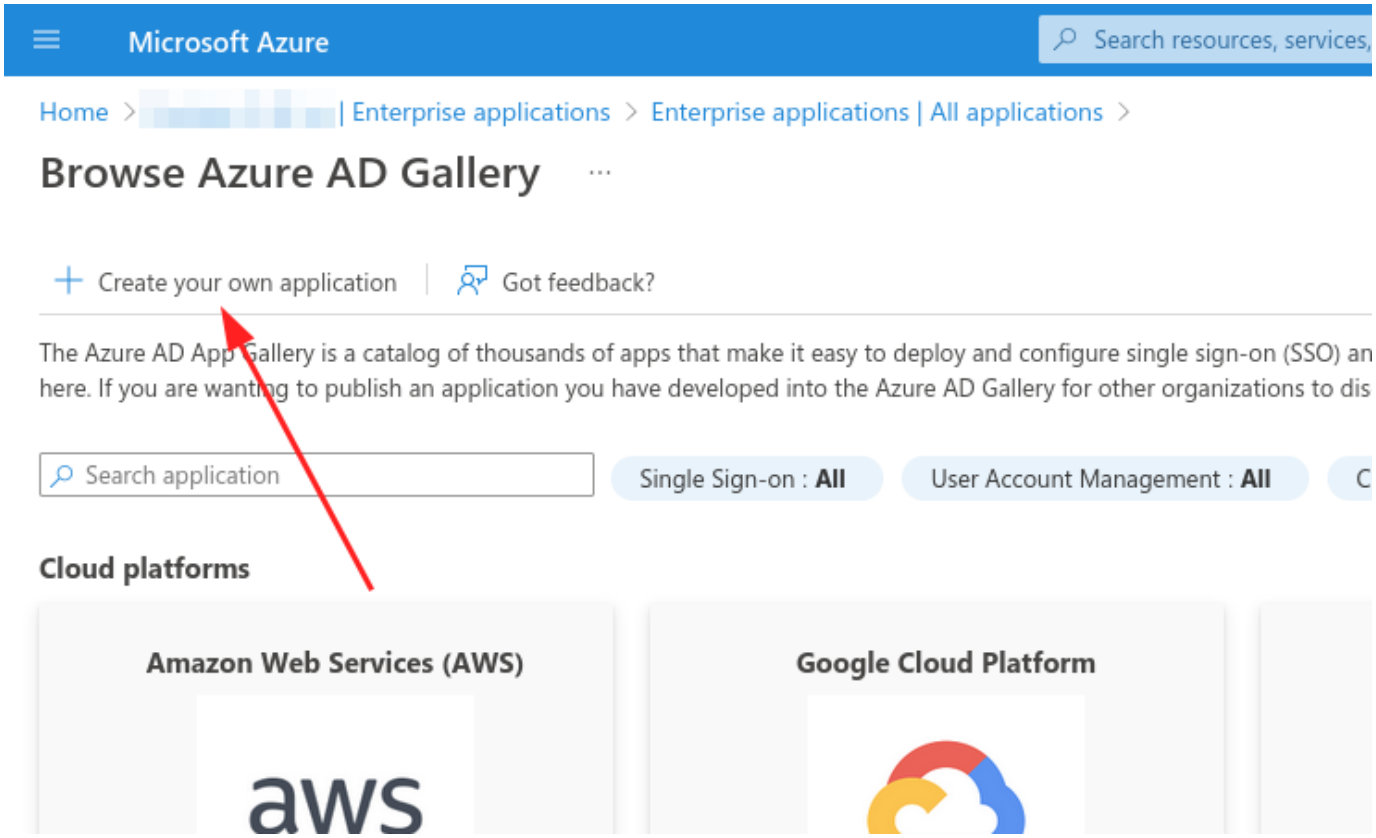


Then click "Create your own application".

Required Role

You will need to have one of the following Azure AD roles in order to create a new application

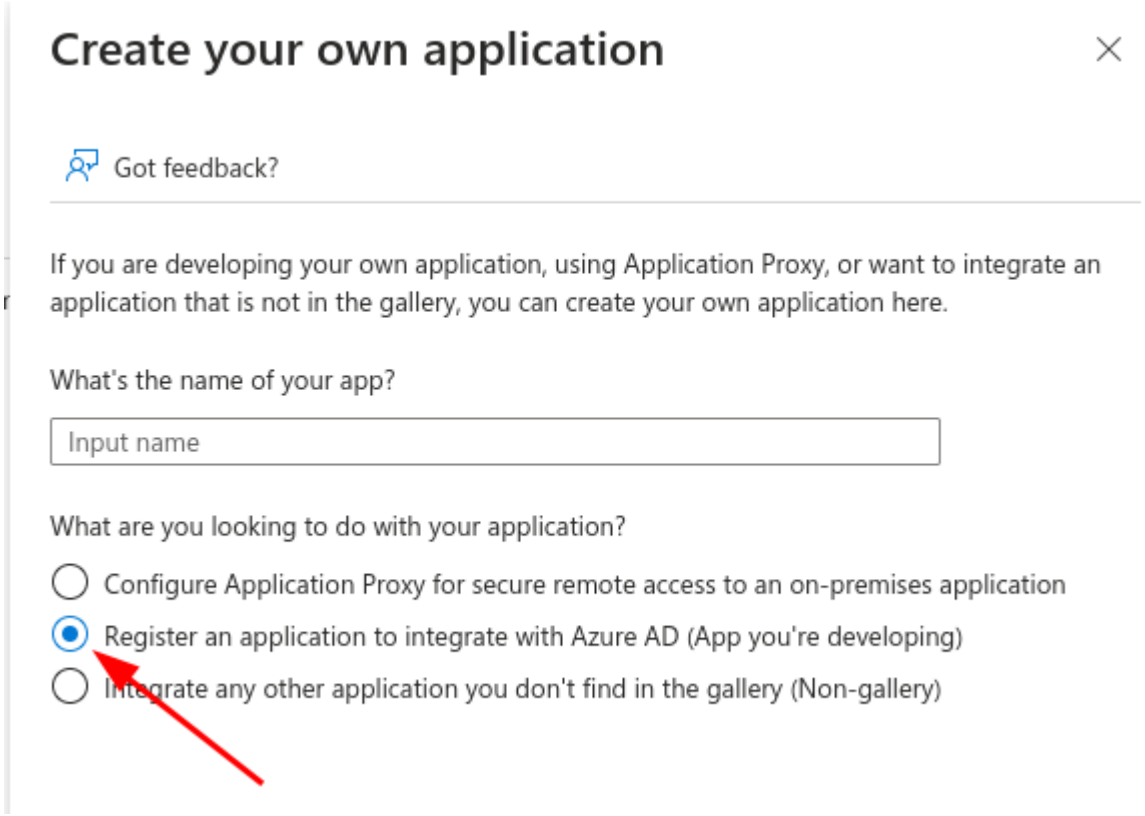
Global Administrator
Application Administrator



Name your app "NodeZero Portal".

Select the "Register an application to integrate with Azure AD (App you're developing)" option.

Click Create.

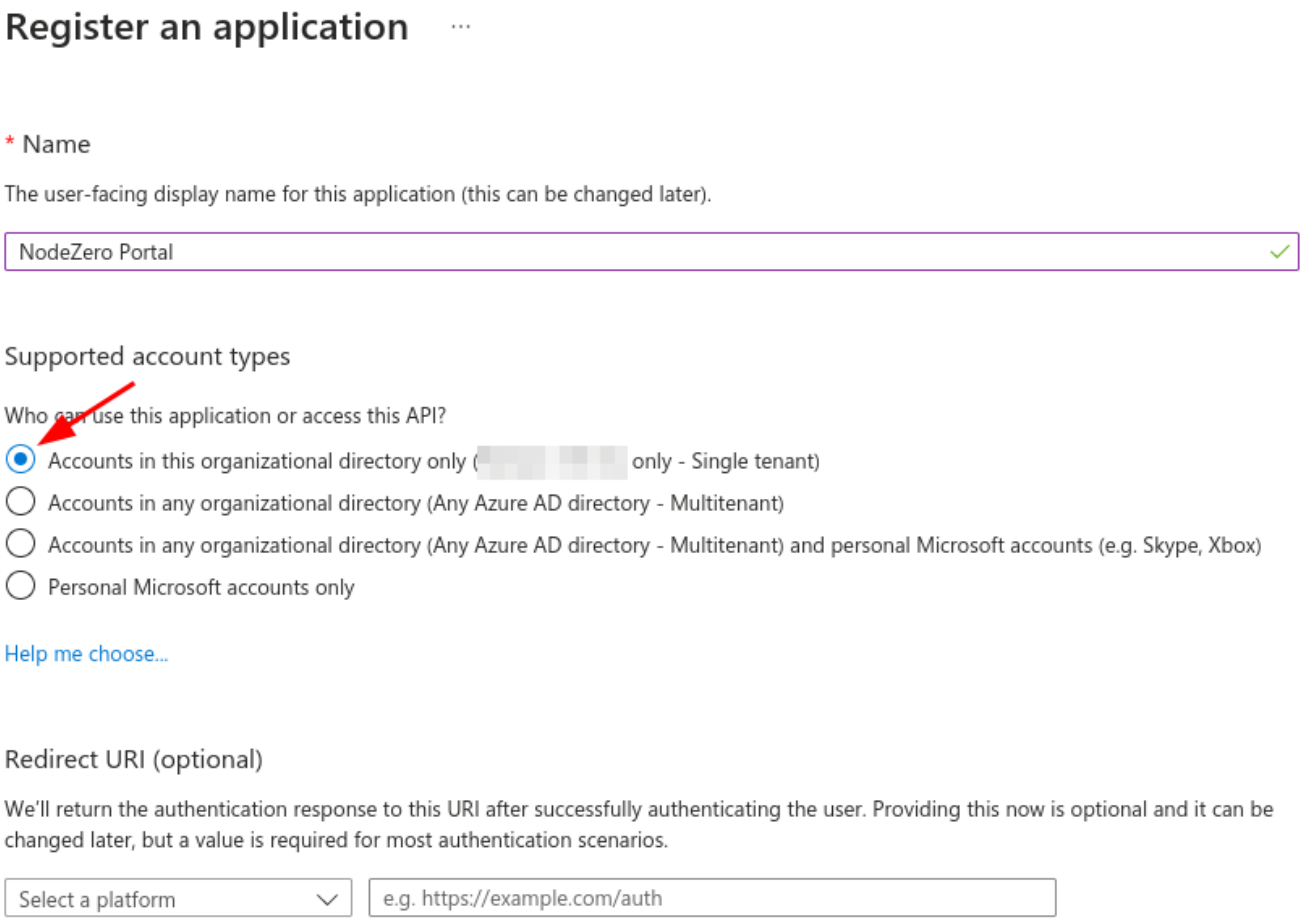


On the "Register an application" page, you can choose to set a different user-facing name for the app, if desired.

Ensure the "Supported account types" option is set to "Single tenant".

Leave the "Redirect URI" section blank for now.

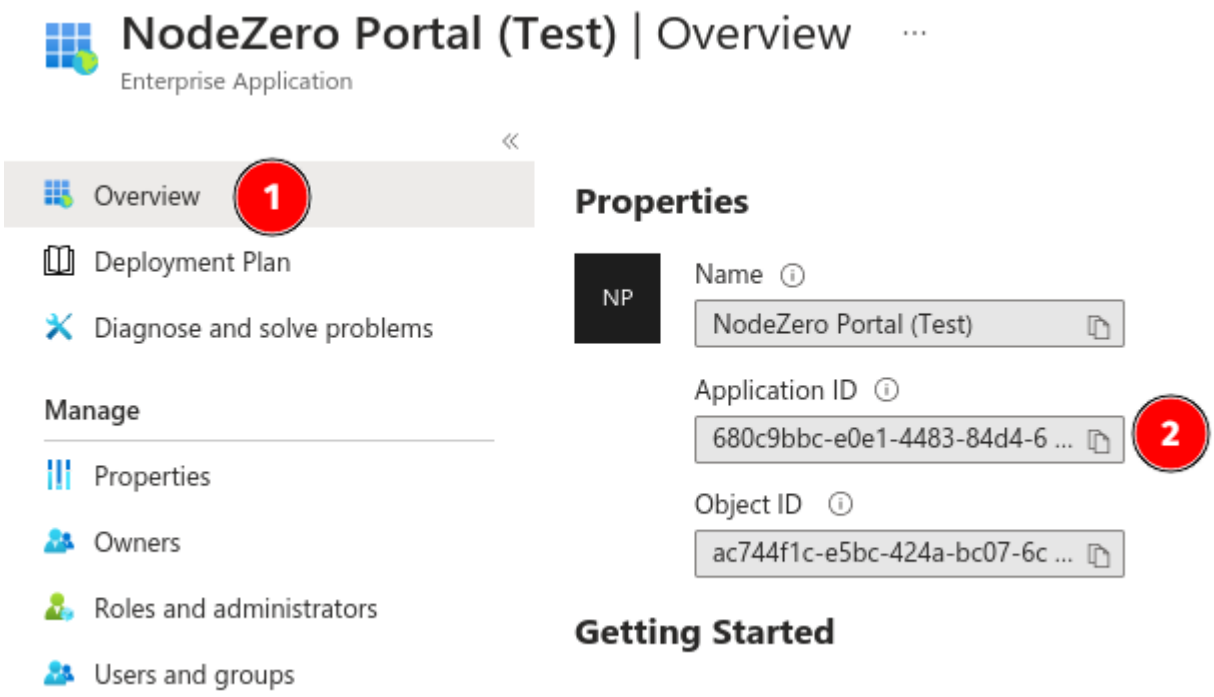
Click Register.



COPY CLIENT ID

After registering the app, you'll be taken back to the "Browse Azure AD Gallery" page. Navigate back to the Enterprise applications page, find your newly created app, and click it.

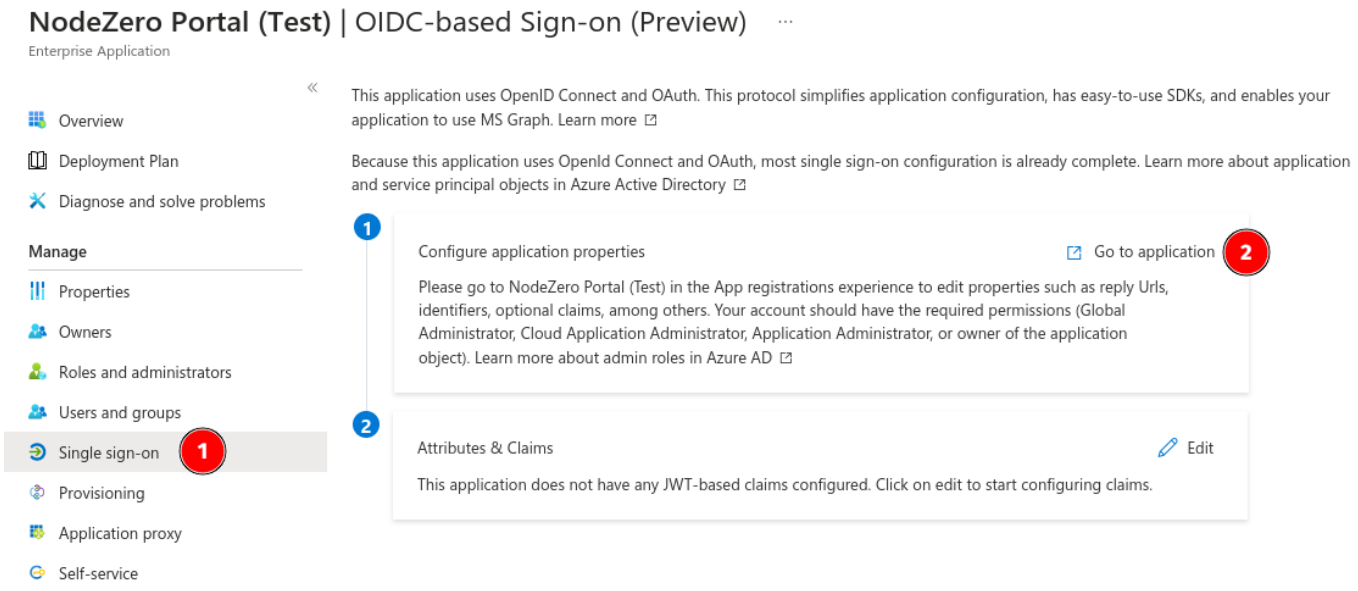
On the Overview page, save the Application ID. This is the Client ID that you will need to provide to your Portal Org Admin later.



CONFIGURE SINGLE SIGN-ON

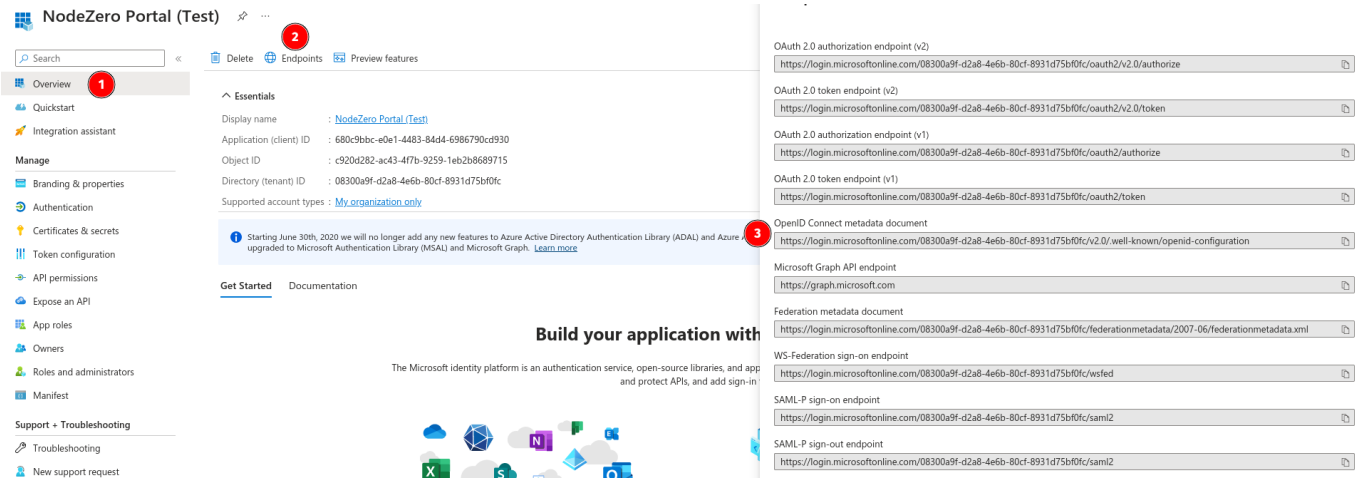
Under the Manage section of the left hand menu, click Single sign-on.

Click `Go to application`.



COPY ISSUER URL

On the new Overview page, click the Endpoints tab and copy the `OpenID Connect metadata document` value. This is this Issuer URL that you will need to provide to your Portal Org Admin later.

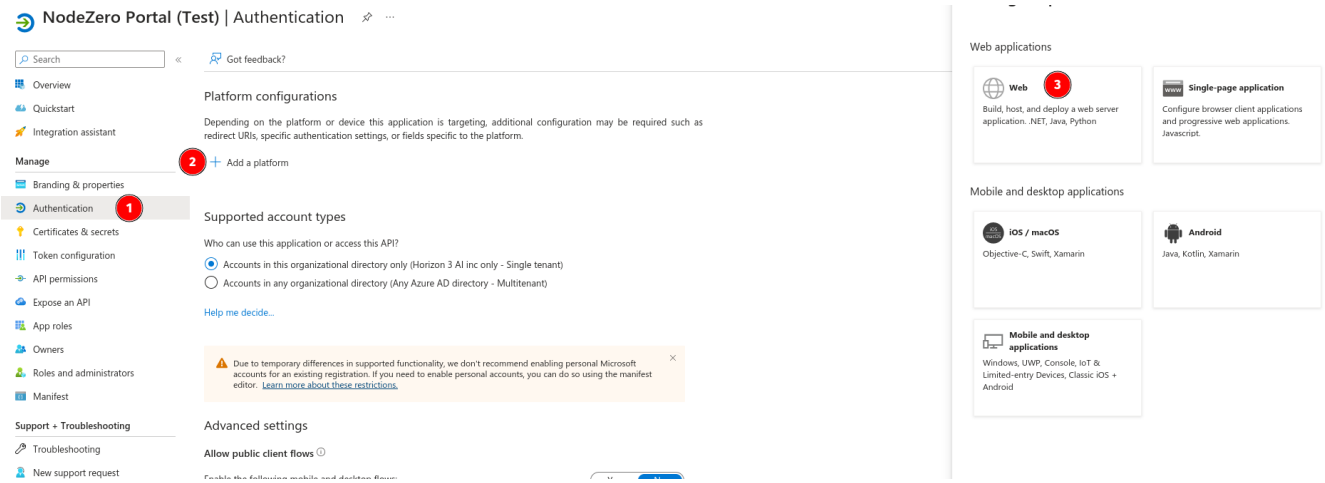


CONFIGURE AUTHENTICATION

Under the Manage section, click Authentication.

Click `Add a platform` under the "Platform configurations" section.

In the form that opens to the right, click the `Web` button under the "Web applications" section.



Use the information in the below table to fill out the "Redirect URIs" field. Be sure to select the correct tab based on which regional Portal your users access.

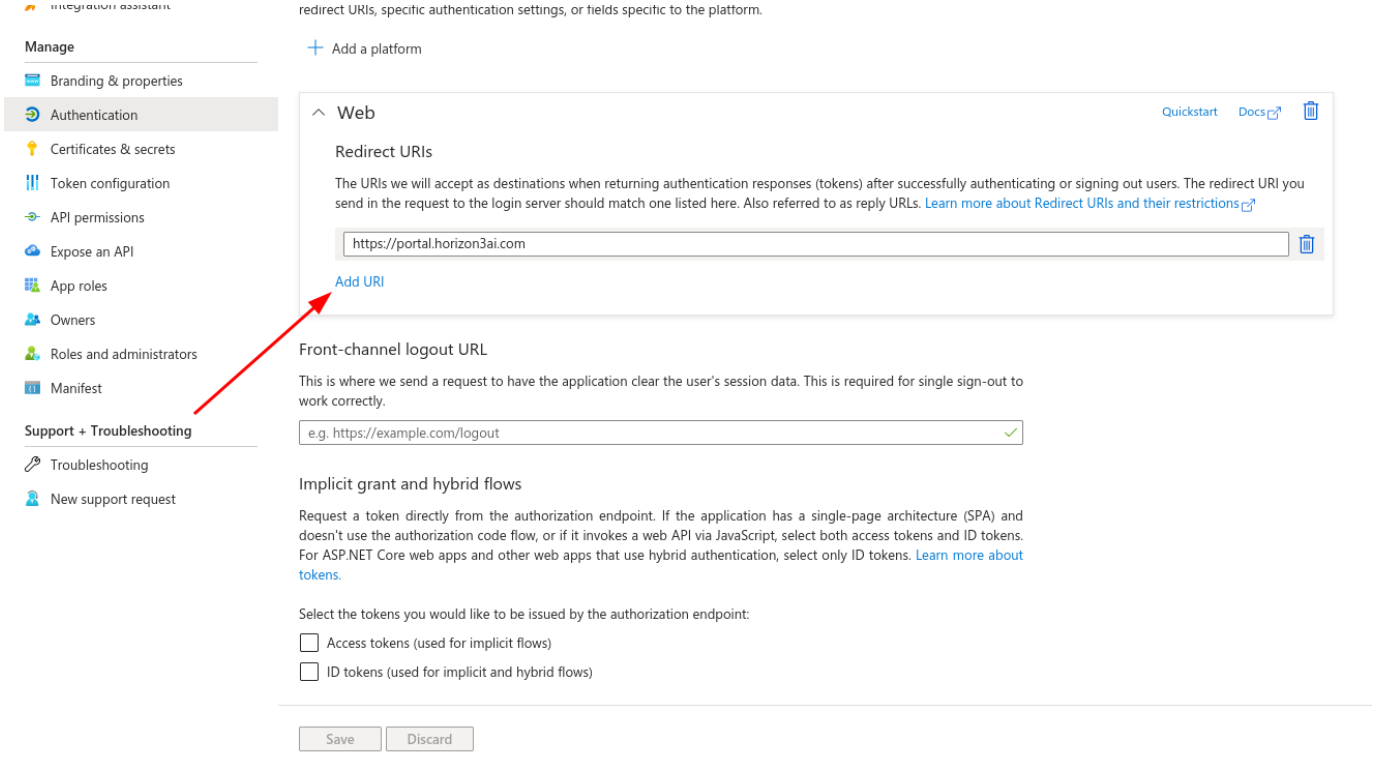
US Portal EU Portal

Field	Value
Sign-in redirect URIs	https://portal.horizon3ai.com https://auth.horizon3ai.com/oauth2/idpresponse

Field	Value
Sign-in redirect URIs	https://portal.horizon3ai.eu https://auth.horizon3ai.eu/oauth2/idpresponse

Adding multiple Sign-in redirect URIs

The initial form appears to only allow you to enter a single URI. Enter the first URI from the appropriate table below, click Configure, then click the `Add URI` link in the Web > Redirect URIs section on the main page. Enter the 2nd URI and click `Save`.



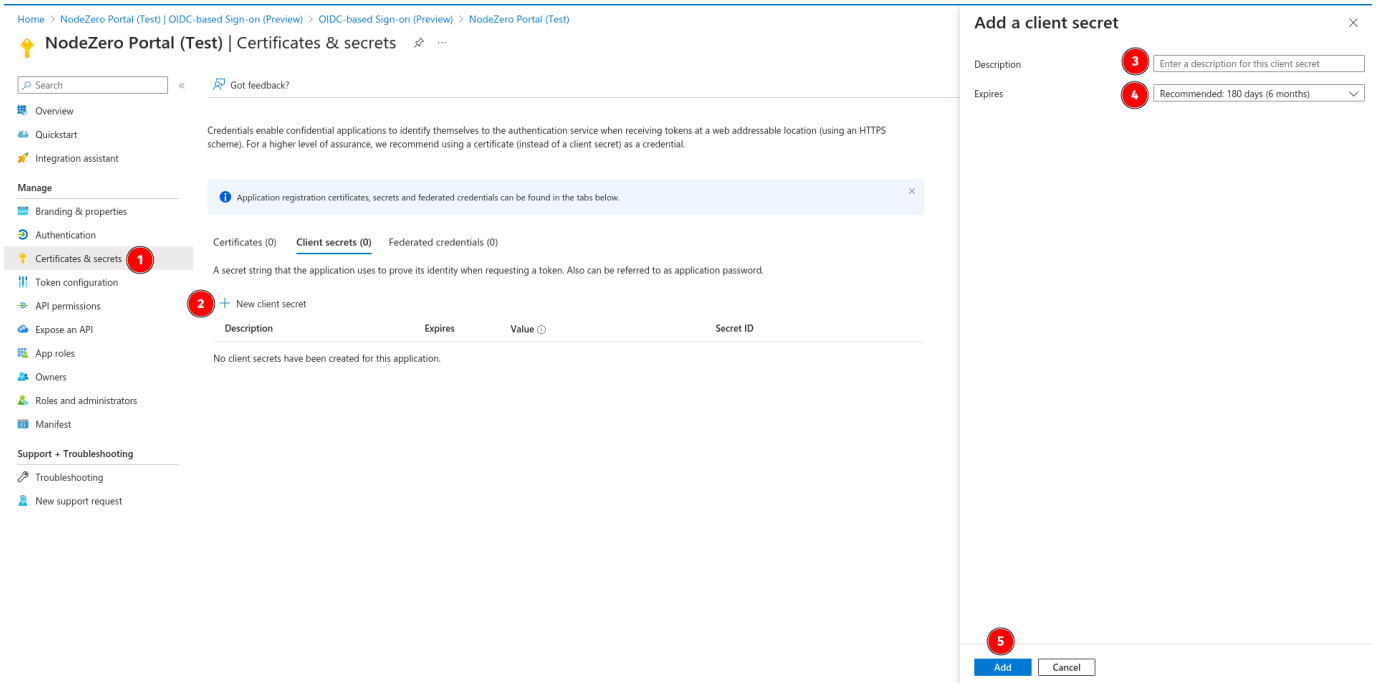
CREATE CLIENT SECRET

Under the Manage section, click Certificates & Secrets.

Click `New client secret`.

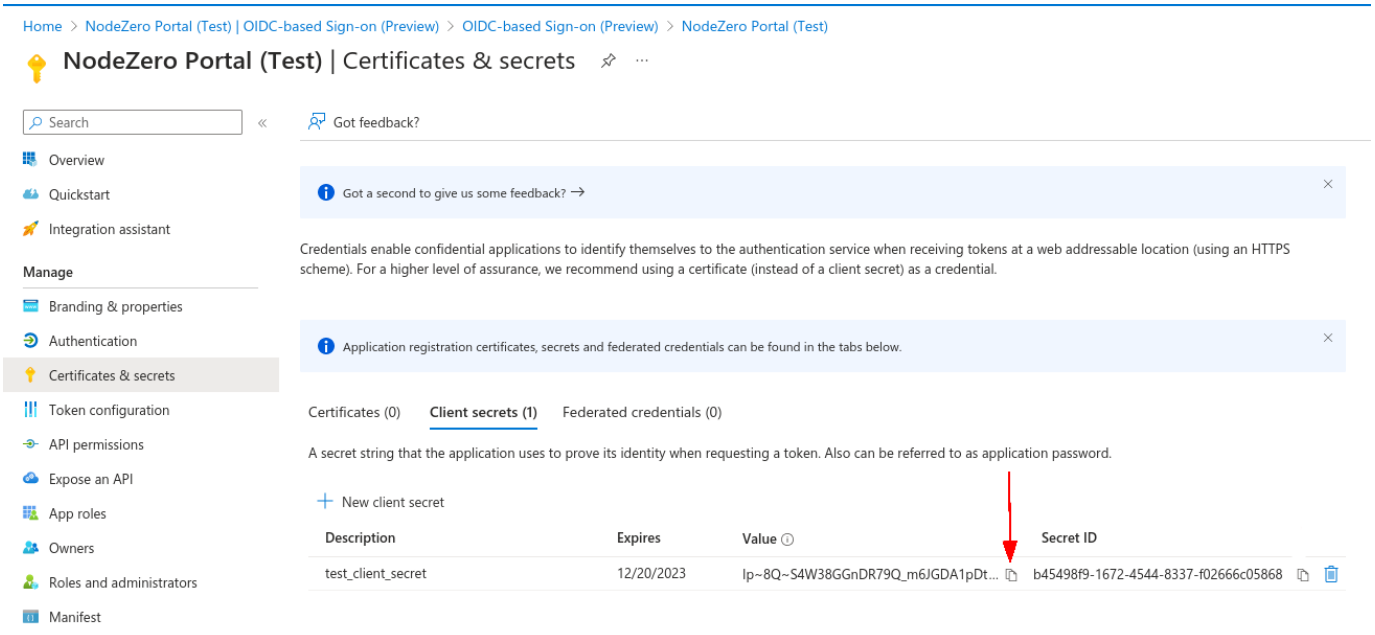
Enter a description. Set the `Expires` column to a value that aligns with your Company's policies.

Click `Add`.



Copy the Secret Value.

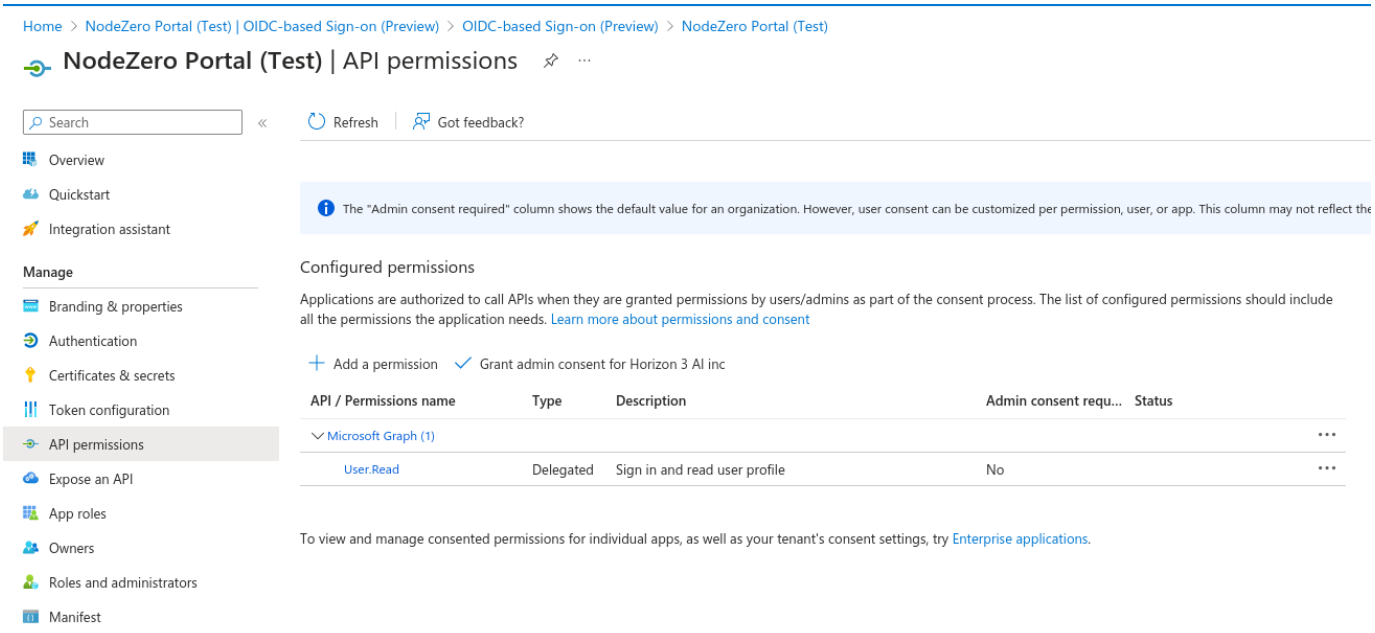
This is the Secret VALUE, that you will need to provide to your Portal Org Admin later.



CONFIGURE API PERMISSIONS

Under the Manage section, click Api permissions.

Ensure the Microsoft Graph `User.Read` permission is configured (it should be by default).



CONFIGURE APP ROLES

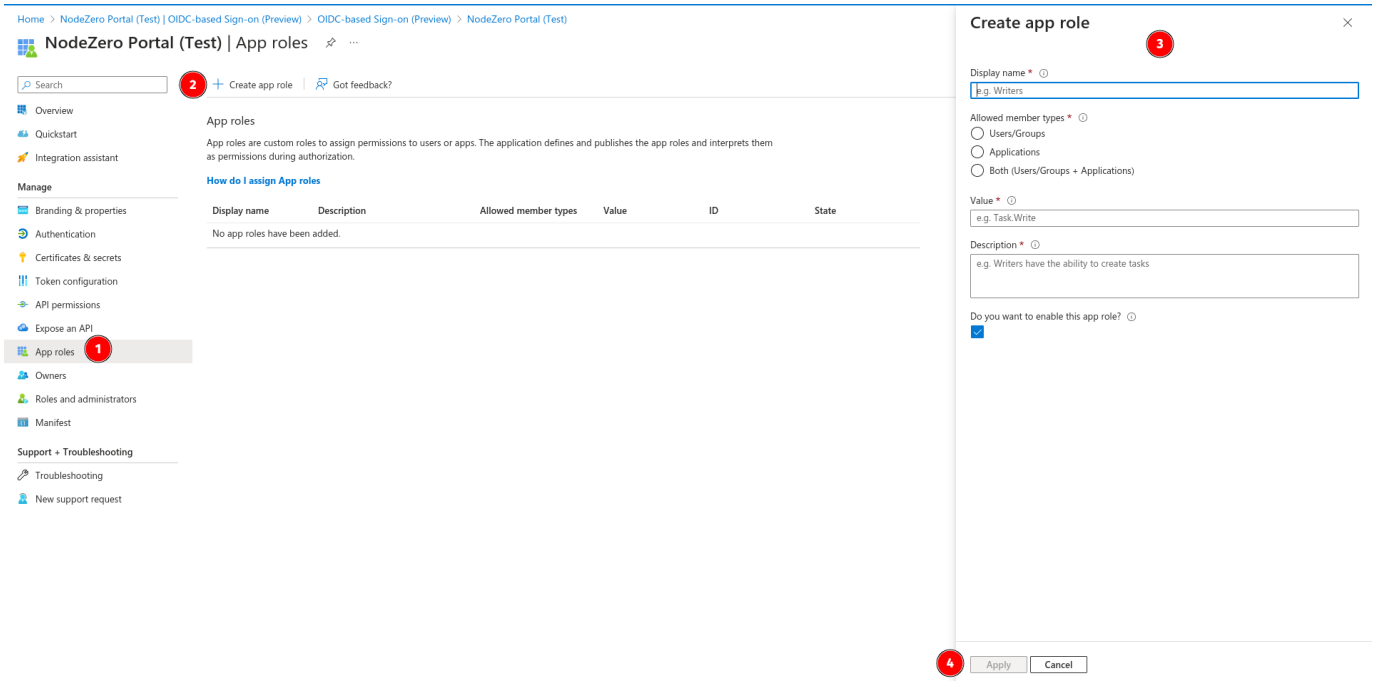
Under the Manage section, click `App roles`.

Click `Create app role`.

Fill out the form that opens on the right using the information in the table below.

Field	Value
Display name	NodeZero Portal Users
Allowed Member Types	Users/Groups
Value	Read
Description	App role granting read to NodeZero Portal app.
Do you want to enable this app role?	✓

Click `Apply`.



PROVIDE INFORMATION TO ORG ADMIN

Provide the Client ID, Client Secret, and Issuer URL you copied in previous steps to your Portal Org Admin so they can configure the SSO Provider in Portal. After the SSO Provider has been set up, your Portal Org Admin will need to provide you the Initiator URL so you can complete the app configuration.

CONFIGURE BRANDING & PROPERTIES

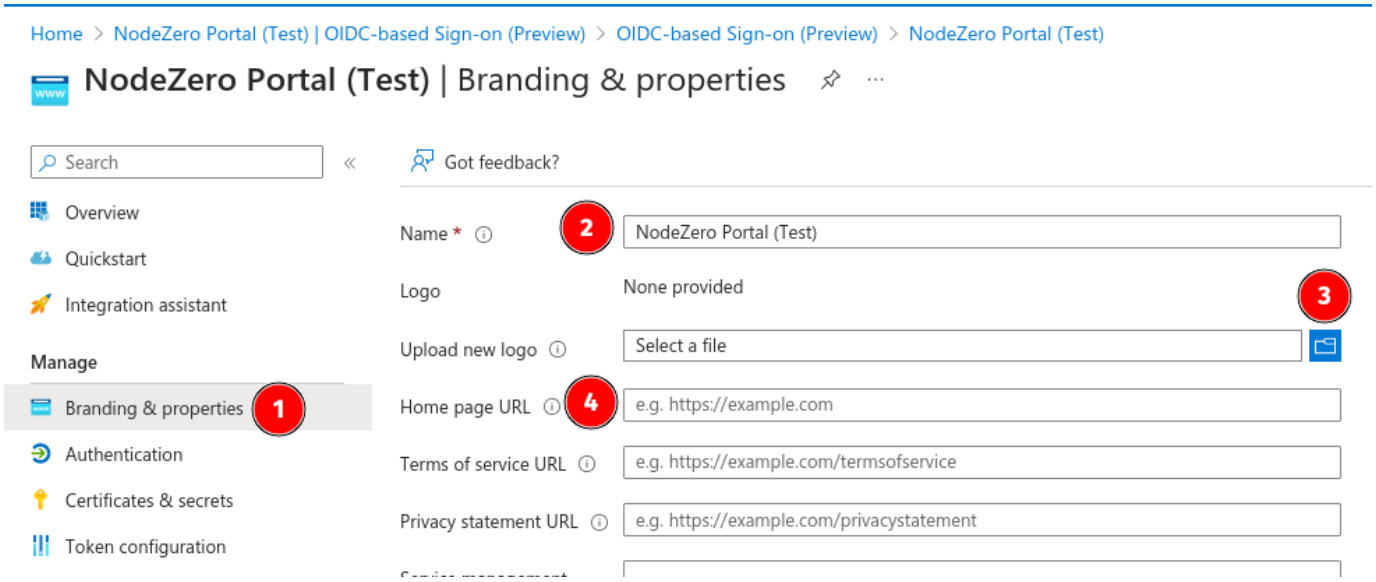
> **Initiator URL**
>
> You will need the Initiator URL from your Portal Org Admin before you can proceed with this section.

Under the Manage section, click `Branding & properties`.

Fill out the form using the information in the table below.

Field	Value
Name	NodeZero Portal
Logo	H3 Logo
Home page URL	Add Initiate login URI here

Click `Save`.

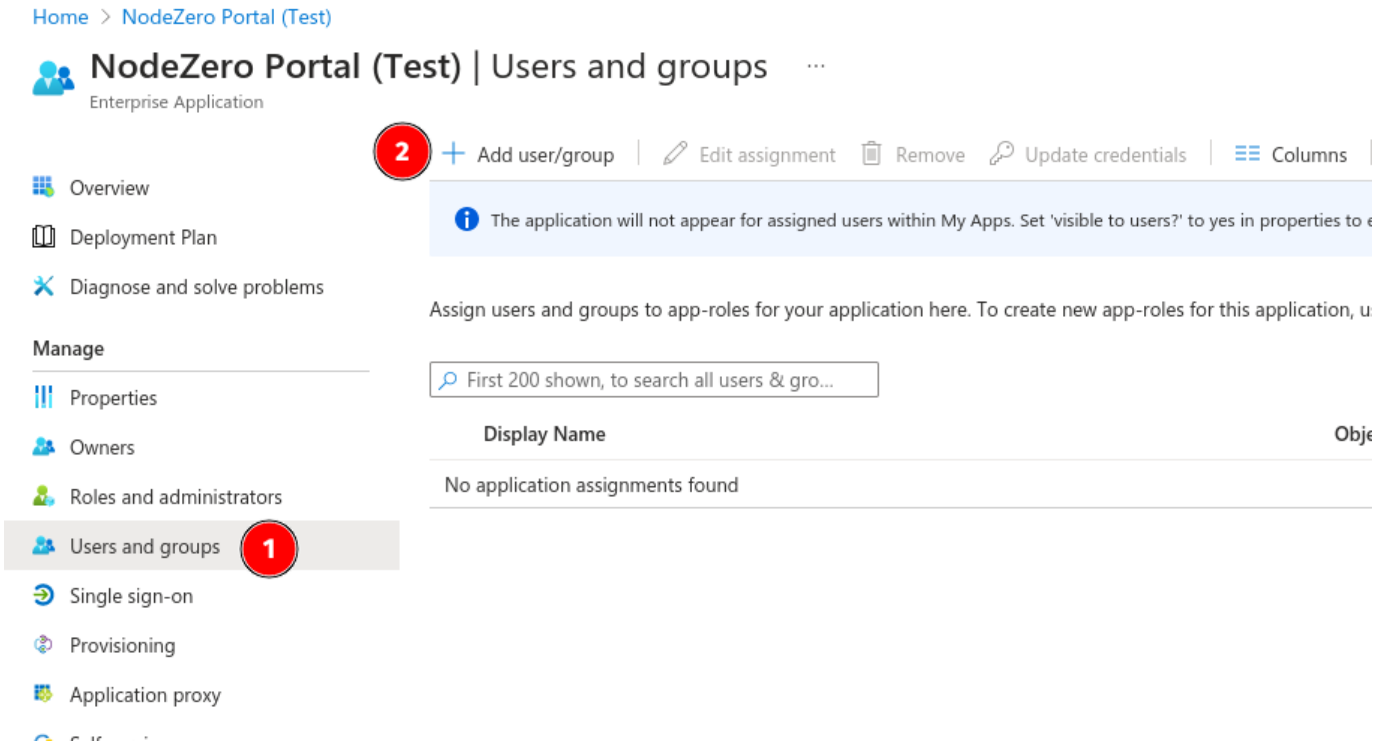


CONFIGURE USERS AND GROUPS

To grant users access to your new app, you will first need to navigate back to the Enterprise applications page we visited at the beginning of this guide.

Under the Manage section, click `Users and groups`.

Click `Add user/group`.



Select the appropriate users/groups.

Select the "NodeZero Portal Users" app role we created in a previous step.

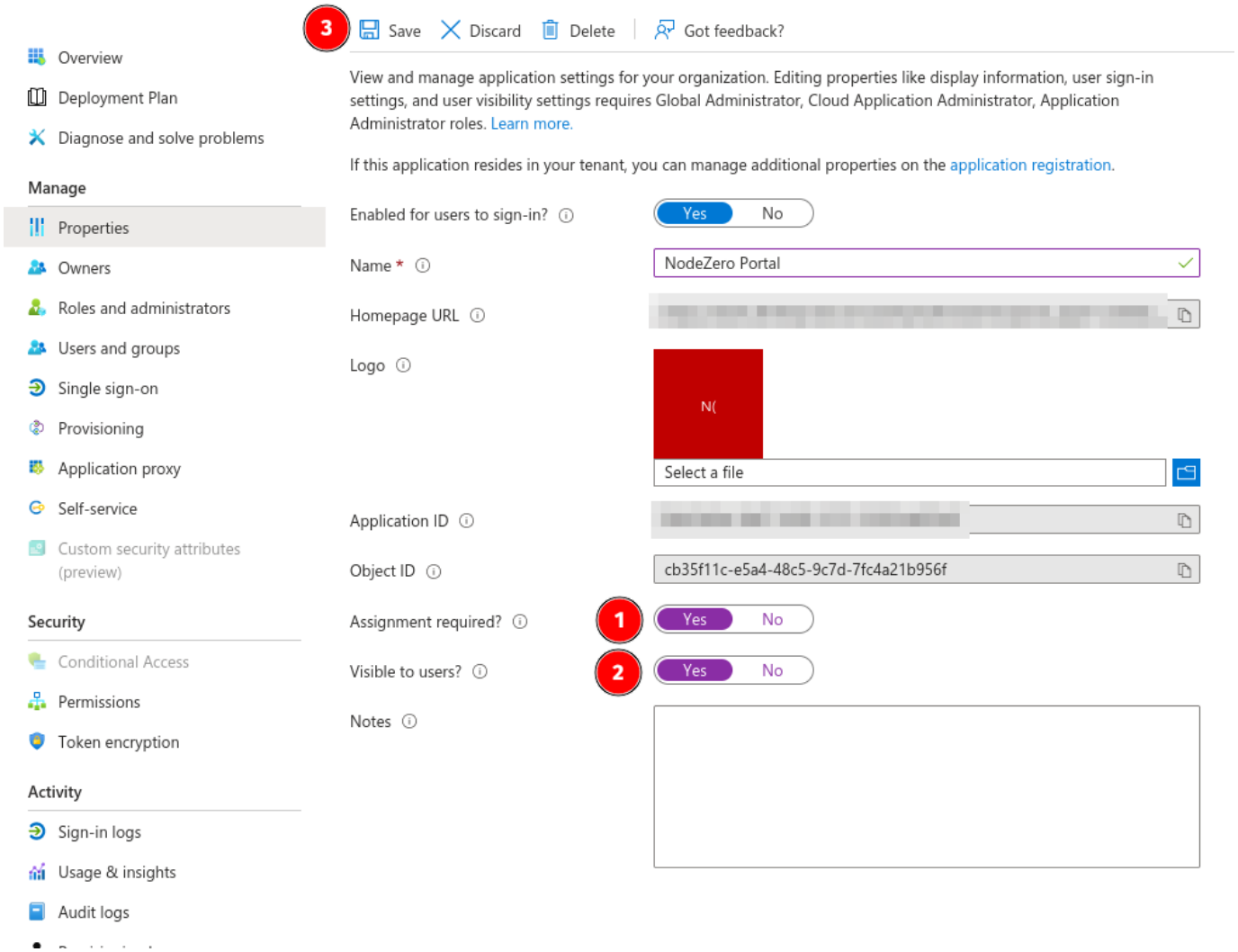
Click `Assign`.

EDIT APP PROPERTIES

By default, the app will not appear for assigned users within MyApps. You will need to edit the visibility and assignment properties of the app.

Under the Manage section, click `Properties`.

Slide the toggle to "Yes" for both `Assignment required?` and `Visible to users?`.



It can take 5 - 10 minutes for the app to appear in MyApps.

At this point, users can access by navigating to MyApps, logging in with their company credentials, and selecting the `NodeZero Portal` application tile.

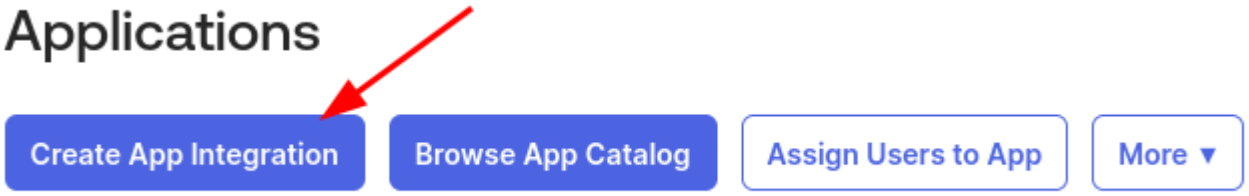
Okta

> **Warning**
>
> This guide should be used as a functional example only. Identity Admins should follow their Company's policies and best practices when implementing Single Sign-On (SSO).
>
> Similarly, because these guides are for services Horizon3 does not control, screenshots and configuration options may be different then what you see here.

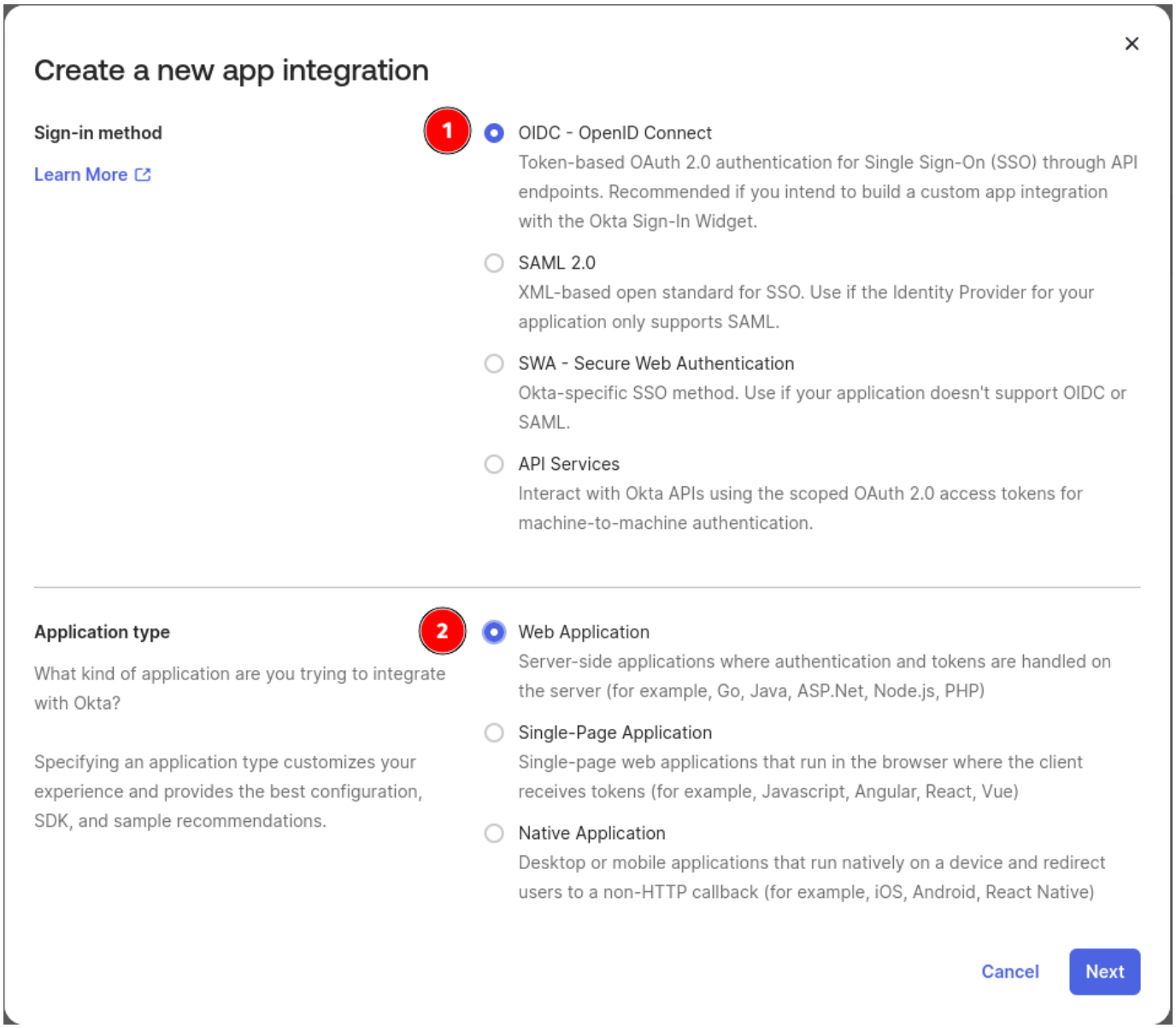
All sections of this page should be completed by someone with the Identity Team Admin Persona

CREATE OKTA APP INTEGRATION

Log into your Okta Admin Console, browse to the Applications section, and click `Create App Integration`.



In the "Create a new app integration" form, select `OIDC - OpenID Connect` as the Sign-in method and then `Web Application` as the Application Type.


Create a new app integration
Sign-in method
Learn More
1
OIDC - OpenID Connect
Token-based OAuth 2.0 authentication for Single Sign-On (SSO) through API endpoints. Recommended if you intend to build a custom app integration with the Okta Sign-In Widget.
SAML 2.0
XML-based open standard for SSO. Use if the Identity Provider for your application only supports SAML.
SWA - Secure Web Authentication
Okta-specific SSO method. Use if your application doesn't support OIDC or SAML.
API Services
Interact with Okta APIs using the scoped OAuth 2.0 access tokens for machine-to-machine authentication.
Application type
What kind of application are you trying to integrate with Okta?
Specifying an application type customizes your experience and provides the best configuration, SDK, and sample recommendations.
2
Web Application
Server-side applications where authentication and tokens are handled on the server (for example, Go, Java, ASP.Net, Node.js, PHP)
Single-Page Application
Single-page web applications that run in the browser where the client receives tokens (for example, Javascript, Angular, React, Vue)
Native Application
Desktop or mobile applications that run natively on a device and redirect users to a non-HTTP callback (for example, iOS, Android, React Native)
Cancel
Next

Click Next. Use the information in the below table to fill out the "New Web App Integration" page. Be sure to select the correct tab based on which regional Portal your users access.

US Portal EU Portal

Field	Value
App integration name	NodeZero Portal
Logo	H3 Logo
Grant Type	Authorization Code
Sign-in redirect URIs	https://portal.horizon3ai.com https://auth.horizon3ai.com/oauth2/idpresponse
Assignments	Limit access to selected groups OR Skip group assignment for now

Field	Value
App integration name	NodeZero Portal
Logo	H3 Logo
Grant Type	Authorization Code
Sign-in redirect URIs	https://portal.horizon3ai.eu https://auth.horizon3ai.eu/oauth2/idpresponse
Assignments	Limit access to selected groups OR Skip group assignment for now

Note about Groups

If you set Assignments above to "Limit access to selected groups", you will need to add the Okta groups that should be assigned the app in order to save your settings.

Portal does not pull groups or roles from the IdP so they cannot be used for providing access via Portal.

Click Save. Your page should refresh and you should be on the app console for your newly create app!

GATHER INFO

Under the `Client Credentials` section, copy the `Client ID`.

Under the `Client Secrets` section, copy the `Secret`.

Copy your Issuer URL. This is most often your okta domain. Include the `https://` prefix when submitting the Issuer URL

Provide the Client ID, Client Secret, and Issuer URL to your Portal Org Admin so they can configure the SSO Provider in Portal. After the SSO Provider has been set up, your Portal Org Admin will need to provide you the Initiator URL value so you can complete the app configuration.

EDIT LOGIN SETTINGS

Initiate login URI

You will need the Initiator URL value from your Portal Org Admin before you can proceed with this section.

This is the value of the "Initiator URL" returned when the Org Admin creates the SSO Provider in Portal.

In the Okta Admin console, browse to your recently created "NodeZero Portal" app.

Scroll down to the "General Settings" section and click the Edit button.

Under the "General Settings > Login" section set the "Sign-out redirect URIs" to your Issuer URL. This is most often your okta domain.

Under the "General Settings > Login" section click the `Login initiated by` dropdown menu and select "Either Okta or App". This should cause the `Application visibility` and `Login flow` fields to appear.

For the `Application visibility` field, check only the box for "Display application icon to users".

For the `Login flow` field, select the "Redirect to app to initiate login (OIDC Compliant)" option.

For the `Initiate login URI` field, enter the value of the "Initiator URL" provided by your Portal Org Admin.

Click Save.

You can now assign this app to users. When you do, they should see a "NodeZero Portal" tile in their Okta App Dashboard and Okta Browser Plugin. Clicking it should log the user into Portal.

3. Reference

3.1 Reference

Helpful resources pertaining to Horizon3.ai products and services:

1. API & CLI - programmatically control and review pentests
2. Automate Scheduling - automate scheduling of pentests
3. Attack Configuration - configure pentest behavior
4. Injecting Credentials - run NodeZero from an authenticated perspective
5. Phishing Impact Test - use NodeZero to measure the impact of simulated phishing campaigns
6. NodeZero Modules - review specific NodeZero functionality
7. Notifications - notifications during pentest lifecycle
8. Media - videos available in NodeZero
9. Templates - manage pentest templates
10. Release Notes - learn about the latest improvements

See navigation pane for the full list of resources.

3.2 API

3.2.1 API

Horizon3.ai provides a publicly accessible Application Programming Interface (API), powered by GraphQL, which offers a subset of features available in the Horizon3.ai Portal. To use the API, ready-to-use tools and documentation for API queries and mutations are available:

1. CLI Tool - ready-to-use CLI tool for interacting with the GraphQL API
2. API Reference - GraphQL API documentation of queries and mutations

---

References:

1. Horizon3.ai Portal
2. GraphQL,org

3.2.2 CLI Tool

CLI Tool

Horizon3.ai provides its users with a Command Line Interface (CLI) tool, called `h3-cli`. It is convenient for accessing the Horizon3.ai Application Programming Interface (API). The Horizon3.ai API provides programmatic access to a subset of functionality available through the Horizon3.ai Portal.

The `h3-cli` tool is available for download from the Horizon3.ai GitHub. See here for instructions on using `h3-cli`:

Getting Started

Additional Guides:

1. Automate NodeZero Deployment - auto-deploy NodeZero to a configured host using a NodeZero Runner
2. Scheduling from CLI - configure recurring pentests via the CLI
3. Auto-inject Credentials - automatically inject credentials into a recurring pentest
4. Monitor Pentests - monitor the status of active pentests
5. Paginate Results - paginate the results of pentests

Develop Your Own Tools

Alternatively, you may develop your own software tools to interact with the Horizon3.ai GraphQL API.

References:

1. h3-cli on GitHub
2. Horizon3.ai Portal

Getting Started

H3-CLI: CLI TOOL FOR THE HORIZON3.AI API

h3-cli is a convenient CLI (command-line interface) for accessing the Horizon3.ai API. The Horizon3.ai API provides programmatic access to a subset of functionality available through the Horizon3.ai Portal. At a high level, the API allows you to:

- schedule an autonomous pentest
- download and run NodeZero™
- monitor the status of a pentest while it is running
- retrieve a pentest report after it's complete

The API can be used for a variety of use cases such as scheduling periodic assessments of your environment or kicking off a pentest as part of a continuous integration build pipeline.

INSTALLATION AND INITIAL SETUP

The steps below will get you up and running quickly with h3-cli. These instructions were tested on Linux machines, and generally should work on any POSIX-compliant system with bash support.

If you plan to run internal pentests using h3-cli, you should install h3-cli on the same Docker Host where you launch NodeZero.

It is assumed you already have an account with Horizon3.ai. If not, sign up at https://portal.horizon3ai.com/.

1. Create an API key

An API key is required to access the H3 API. You can create one in the Portal under the User -> Account Settings menu.

When creating an API key you must assign it a role that controls its permissions. The available roles are:

- **User:** Basic read/write permissions. The API key can run pentests and read results.
- **Read-only:** The API key can read pentest results, but cannot run pentests.
- **NodeZero Runner:** A specialized, heavily restricted role designed specifically for NodeZero Runners.

We recommend the **User** role if you're testing out h3-cli and want to experiment with all its features. After that, you may want to use more restrictive permissions, based on your use case. For example, if you only want to use h3-cli to set up a NodeZero Runner, we recommend using the **NodeZero Runner** role.

You can easily manage multiple API keys within the same h3-cli install. Learn more here.

Keep your API key secure, as anyone with your API key can access your H3 account. Think of an API key as a username + password rolled into one. Anyone with the API key can access your account from anywhere. h3-cli will store your API key under the `$HOME/.h3` directory. This directory is created during installation and configured with permissions such that only you can read or write to it.

2. Install h3-cli

Install the h3-cli git repo on your machine by executing the following git command within a shell/terminal session.

```
git clone https://github.com/horizon3ai/h3-cli
```

This will create a new directory, `h3-cli`, and download the contents of the repo to it. The `h3-cli` directory will be created in the directory where you run the git command. You can install h3-cli anywhere on the filesystem.

If you don't have git, you can download the repo as a zip archive from the menu above, and unzip it anywhere on the filesystem.

3. Run the h3-cli install script

Run the following commands to install and configure h3-cli. Substitute `your-api-key-here` with your actual API key.

```
cd h3-cli
bash install.sh your-api-key-here
```

The install script will install dependencies (jq) and create your default h3-cli profile under the `$HOME/.h3` directory. Your API key is stored in your h3-cli profile. The directory and profile permissions are restricted so that no other users (besides yourself) can read or write to it.

The install script will ask you to edit your shell profile (`$HOME/.bash_profile` or `$HOME/.bash_login` or `$HOME/.profile`, depending on your operating system) to set the following environment variables:

- `H3_CLI_HOME` : this environment variable is used by h3-cli to locate itself and its supporting files.
- `PATH` : this environment variable specifies the directories to be searched to find a shell command.

After updating your shell profile, re-login or restart your shell session to pick up the profile changes, then verify you can invoke h3 by running it from the command prompt:

```
h3
```

If everything's installed correctly, you should see the h3-cli help text.

UPGRADING H3-CLI

We release new features, bug fixes, and other updates for the h3-cli every month. Upgrade your installation using one of the methods below.

Option 1: Via the `h3 upgrade` command (recommended)

As of June, 2023, you can use the `h3 upgrade` command to upgrade to the latest version of h3-cli.

If you get an `ERROR: unrecognized command: "upgrade"`, then you are on a previous version of h3-cli that does not support the upgrade command. Use one of the methods below to upgrade h3-cli.

Option 2: Via `easy_install.sh` (recommended if `h3 upgrade` is unavailable)

Run this command from h3-cli's parent directory (i.e. the directory that contains the `h3-cli/` directory):

```
curl https://raw.githubusercontent.com/horizon3ai/h3-cli/public/easy_install.sh | bash
```

Option 3: Via git

If you used `git clone` to install the repo, then simply run `git pull` to install the latest version.

Option 4: Via zip download

If you downloaded the repo as a zip file, then re-download the zip file and unzip it to the same location (in other words replace your existing h3-cli installation with the new zip).

Checking the current version

As of June, 2023, you can view your current version of h3-cli via:

```
h3 version
```

GETTING STARTED

1. Verify connectivity with the API

Run the following command to verify connectivity with the API.

```
h3 hello-world
```

You should see the response:

```
{
  "data": {
    "hello": "world!"
```

```
  }
}
```

If you are getting an error response, please contact H3 via the chat icon in the Horizon3.ai Portal.

2. Query the list of pentests in your account

The command below will return the list of pentests in your account, most recent first.

```
h3 pentests
```

To filter for pentests that match a given search term, pass the search term as a parameter:

```
h3 pentests sample
```

3. Query a specific pentest from your account

To query the most recent pentest in your account:

```
h3 pentest
```

To query any pentest in your account, pass the `op_id` of the pentest as a parameter:

```
h3 pentest your-op-id-here
```

Several h3-cli commands will use the most recent pentest as the default, unless an `op_id` is passed as a parameter.

The terms "op" and "pentest" are often used interchangeably.

4. Run a pentest

Running a pentest requires specifying an op template. An op template specifies a full pentest configuration, which includes scope, attack parameters, and other (optional) configuration.

Horizon3.ai provides new users with a default op template named `Default 1 - Recommended`. This template is always up-to-date with our latest attack parameters and recommended configuration. The default template does not define a scope, in which case NodeZero will use Intelligent Scope - NodeZero's host subnet will provide the initial scope, and it will expand organically during the pentest as more hosts and subnets are discovered. For more information on Intelligent Scope and other deployment options, visit our product documentation.

For experienced users, custom op template(s) may be created via the Horizon3.ai Portal. To create a custom op template, walk through the Run a Pentest modal until you see the option to customize the pentest configuration. The op template can be created without actually running the pentest.

To provision a pentest using the default op template and Intelligent Scope:

```
h3 run-pentest
```

The JSON response contains the details for the newly created pentest. You can verify the pentest is provisioning by checking your Horizon3.ai Portal, or by running `h3 pentest`.

There are several ways to specify additional parameters when creating pentests. For more information see additional examples here.

WAIT! YOU'RE NOT DONE!

For internal pentests (which are the default), additional steps are required before the pentest will begin running. See the next section about downloading and running NodeZero in order to complete the initiation of your pentest.

If you are running an external pentest, NodeZero is launched for you automatically in the H3 cloud as part of `h3 run-pentest`, in which case there are no additional steps on your end to initiate the pentest.

5. Run NodeZero

◌The following step applies to internal pentests only; for external pentests, NodeZero is launched for you automatically in the H3 cloud.

After creating an internal pentest, you then have to run our NodeZero container on a Docker Host inside your network. This is done by running the NodeZero Launch Script on your Docker Host.

To run the NodeZero Launch Script for your most recently created pentest:

```
h3 run-nodezero
```

YOUR PENTEST HAS BEEN LAUNCHED! Assuming all commands ran without error, then you have successfully created and launched your pentest. You should see output from the NodeZero Launch Script being logged to the console. The script will first verify your system is compatible with NodeZero before downloading and running it. When the pentest is complete, NodeZero will automatically shut itself down.

NodeZero is a Docker container. You can view it using `docker ps`. The container name will be of the form `n0-xxxx`.

6. Download pentest reports

After your pentest has finished, use the following command to download a zip file containing all PDF and CSV reports for the most recently created pentest:

```
h3 pentest-reports
```

The above command will download the zip file to `pentest-reports-{op_id}.zip` in the current directory.

USE CASES

- **Automated NodeZero deployment using a NodeZero Runner.** Learn how to use a NodeZero Runner to deploy NodeZero on your Docker Host automatically, without having to manually copy+paste the NodeZero Launch Script.
- **Automated scheduling.** Learn how to run pentests automatically on a regular schedule, for example once a week or once a month.
- **Auto-injected credentials.** Learn how to automatically inject credentials into a regularly scheduled pentest using a NodeZero Runner.
- **Monitoring pentests.** Learn how to monitor pentests using h3-cli.
- **Paginating results.** Learn how to paginate through large result sets using h3-cli.
- **JSON Parsing using `jq`.** Learn how to leverage the power of `jq` to parse JSON responses from h3-cli. `jq` can parse specific fields, print the structure of a response, and even transform a JSON response to CSV.

AUTHENTICATION & H3-CLI PROFILES

Authentication happens seamlessly and automatically when you invoke the `h3` command. There's nothing explicit you need to do to authenticate. This section documents the underlying mechanics.

h3-cli reads your `H3_API_KEY` from your h3-cli profile (under `$HOME/.h3`) to authenticate to the Horizon3.ai API and establish a (temporary) session. The session token (a JWT) is cached under `$HOME/.h3`. The session token expires after 1 hour, at which point h3-cli will automatically re-authenticate and re-establish a session.

You can explicitly authenticate using the following command:

```
h3 auth
```

The above command will output the session token (and also cache it under `$HOME/.h3`). If you already have an established (non-expired) session token, `h3 auth` will continue to use that session token rather than re-authenticate.

If you want to force h3-cli to re-authenticate, use the `force` option:

```
h3 auth force
```

h3-cli profiles

You can manage multiple h3-cli authentication profiles under the same `$HOME/.h3` directory. Each h3-cli profile has its own API key.

When you first install h3-cli it will automatically create an initial profile named `default` with the API key you provided to `install.sh`.

If you wish to create another profile with a different API key, use the following command:

```
h3 save-profile my-profile {api-key}
```

This will create a profile named `my-profile` under `$HOME/.h3` for the given `{api_key}`. To activate the profile in your current shell session, use the following command (note the leading dot `.`):

```
. h3 profile my-profile
```

You can verify the currently active profile using `h3 profile`, and view details about its API key using `h3 whoami`:

```
h3 profile
h3 whoami
```

You can save multiple API keys under different h3-cli profiles and switch between them as needed using the command above. For example, to switch back to the `default` profile:

```
. h3 profile default
```

To view the list of h3-cli profiles under your `$HOME/.h3` directory:

```
h3 profiles
```

You can delete a profile from your `$HOME/.h3` directory using:

```
h3 delete-profile {name}
```

This will remove the profile named `{name}` and its API key from your `$HOME/.h3` directory on the local machine. Note that it will NOT revoke the API key; it only deletes it from the local machine. You can revoke the API key from the Portal.

RUNNING PENTESTS WITH H3-CLI

This section contains additional examples for running pentests using h3-cli.

The simplest way to provision a pentest is to use the default op template and Intelligent Scope:

```
h3 run-pentest
```

To provision a pentest AND launch NodeZero on the local machine (for internal pentests only):

```
h3 run-pentest-and-nodezero
```

Note that this only applies to internal pentests. For external pentests, NodeZero is launched for you automatically in the H3 cloud as part of `h3 run-pentest`.

If you happen to run `h3 run-pentest-and-nodezero` for an external pentest, it will simply skip the part where it downloads and runs NodeZero, since that is handled automatically in the H3 cloud.

To run a pentest using a custom op template, specify it as a parameter to schedule_op_template.graphql:

```
h3 run-pentest '{"op_template_name":"your-op-template-here"}'
```

To run a pentest using the default op template but assign it a name of your choosing, use the optional `op_name` parameter:

```
h3 run-pentest '{"op_name":"your-op-name-here"}'
```

To run a pentest using the default op template but specify its name and scope, use the optional `schedule_op_form` parameter:

```
h3 run-pentest '{"schedule_op_form":{"op_name":"your-op-name-here", "op_param_max_scope": "192.168.0.0/24"}}'
```

Note that `h3 run-pentest` and `h3 run-pentest-and-nodezero` accept all the same optional parameters.

To run a pentest and assign it to a NodeZero Runner named `my-nodezero-runner`:

```
h3 run-pentest '{"schedule_op_form":{"op_name":"Pentest created via h3-cli and launched via runner", "runner_name":"my-nodezero-runner"}}'
```

Running external pentests

If you have an op template configured for your external pentest:

```
h3 run-pentest '{"op_template_name":"your-op-template-here"}'
```

If you do NOT have an op template, you can run an external pentest by first looking up your Asset Group's uuid via `h3 asset-groups`:

```
h3 asset-groups
```

Then use the command below to run an external pentest against that Asset Group. Substitute your Asset Group's uuid for `{your-asset-group-uuid}`:

```
h3 run-pentest '{"schedule_op_form": {"op_type": "ExternalAttack", "asset_group_uuid": "{your-asset-group-uuid}"}}'
```

POWERED BY GRAPHQL

The Horizon3.ai API is powered by GraphQL. In addition to this CLI document, relevant documentation includes:

- Horizon3.ai Docs
- GraphQL.org

Advanced: Writing your own GraphQL queries

h3-cli provides a simple mechanism to run your own GraphQL queries. First you define the GraphQL query in a file (typically with a `.graphql` extension, although that is not required). Then pass the file to `h3 gql`:

```
h3 gql {your-query-file}
```

For example, define the following in a file named `my_session.graphql`:

```
query {
    session_user_account {
        email
        name
        company_name
    }
}
```

Then run:

```
h3 gql ./my_session.graphql
```

You should see the raw JSON response from the GraphQL server. You can pretty-print the JSON response using jq:

```
h3 gql ./my_session.graphql | jq .
```

Important! You must specify the path to the graphql file (full or relative, e.g. `./my_session.graphql` instead of just `my_session.graphql`), otherwise you risk colliding with graphql files that h3-cli uses internally.

Advanced: Parameterized GraphQL queries

GraphQL queries can also define parameters, which are passed to `h3 gql` as a JSON object.

For example, define the following in a file named `my_pentest.graphql`:

```
query q($op_id: String!) {
    pentest(op_id:$op_id) {
        op_id
```

```
        name
        state
    }
}
```

In this example, `$op_id` is a parameter that must be provided in order to run the query. The parameter is passed to the query within a JSON object:

```
h3 gql ./my_pentest.graphql '{"op_id":"your-op-id-here"}' | jq .
```

Substitute `your-op-id-here` with an actual `op_id`.

Guides

AUTOMATE NODEZERO DEPLOYMENT

h3-cli: Automated NodeZero deployment using a NodeZero Runner

Part of the process of running an internal pentest involves launching the NodeZero Docker container on a Docker Host within your private network. This is typically done by manually signing in to your Docker Host and running the NodeZero Launch Script provided in the Portal.

A NodeZero Runner enables automated deployment of the NodeZero Docker container. The NodeZero Runner is a background process running on your Docker Host that listens for newly provisioned pentests and runs the NodeZero Launch Script automatically.

This allows you to provision and deploy pentests fully from the Portal, without having to manually sign in and run the NodeZero Launch Script. It also enables you to run pentests on an automated schedule.

A NodeZero Runner is used for internal pentests only. For external pentests, NodeZero is automatically deployed in the H3 cloud.

Once a NodeZero Runner is up and running, you can assign new pentests to it from the Portal (or via h3-cli). The Runner will launch NodeZero for the pentests assigned to it.

As your pentesting needs grow, you can spin up multiple NodeZero Runners across your network and assign pentests to each of them, enabling autonomous pentesting from various subnets and perspectives within your environment.

Installing and running a NodeZero Runner

There are two ways to install a Runner:

1. The easy way, using the new `easy_install.sh` script that does most of the work for you (recommended).
2. The slightly more manual way of walking thru the installation steps.

Install option #1: The easy way (recommended)

Follow the steps below to install a NodeZero Runner on your Docker Host.

1. Visit the Runners page in the Portal.
2. Click the **Install Runner** button, specify a name for the Runner, then click **Submit**.
3. Copy the provided installation command and run it on your Docker Host.

Barring any errors, **your NodeZero Runner is now up and running**. The Runner will register itself with the Portal and will be listed on the Runners page (you may need to refresh the page).

Auto-Restart: If your Docker Host runs on a Linux system that supports `systemd`, the install script will attempt to register the Runner as a system service with `systemd`. Once registered, `systemd` will automatically restart the Runner upon a system reboot. See Auto-start Runner at system startup for more info.

Behind the scenes, the above steps perform the following actions:

1. **Create an API key for the Runner.** The API key is granted **NodeZero Runner** permissions to the API. This is a specialized role created just for NodeZero Runners, with very restricted access to your account. The role **CANNOT** read existing pentest data, nor can it provision new pentests. The only thing it can do is poll the API to detect when a pentest has been assigned to it, and then run the NodeZero Launch Script.
2. **Install h3-cli on your Docker Host.** h3-cli is installed in a new `h3-cli` directory within the directory where you ran the installation command.
3. **Spin up the Runner using h3-cli.** The NodeZero Runner process is started via the `h3 start-runner` command.

Install option #2: The slightly more manual way

The steps below accomplish the same thing as install option #1, in a slightly more manual and transparent way.

1. Create an API key for the Runner

The NodeZero Runner communicates with the H3 API using h3-cli. As such, it requires an API key. API keys can be provisioned in the Portal here (or here for EU).

We recommend setting the permission level to **NodeZero Runner**. This is a specialized role created just for NodeZero Runners, with very restricted access to your account. The role **CANNOT** read existing pentest data, nor can it provision new pentests. The only thing it can do is poll the API to detect when a pentest has been assigned to it, and then run the NodeZero Launch Script.

2. Install h3-cli on your Docker Host

The NodeZero Runner process is started via the h3-cli. Therefore, the h3-cli must be installed on your Docker Host.

The quick-and-easy install steps are below. Simply copy+paste the commands into a shell on your Docker Host. For reference, full installation instructions are available here.

```
git clone https://github.com/horizon3ai/h3-cli
cd h3-cli
bash install.sh {your-api-key-here}
export H3_CLI_HOME=`pwd`
export PATH="$H3_CLI_HOME/bin:$PATH"
```

The above steps will:

1. Install h3-cli from the public git repo
2. `cd` into the h3-cli dir
3. Run the install.sh script, substituting `{your-api-key-here}` with the API key you created in step 1 above.
4. Add h3-cli to your command `$PATH`

Once installed, run the command below to verify h3-cli is working and is using your newly provisioned API key:

```
h3 whoami
```

3. Spin up the Runner using h3-cli

Use the following h3-cli command to spin up the NodeZero Runner:

```
h3 start-runner my-nodezero-runner /tmp/my-nodezero-runner.log
```

The NodeZero Runner process runs in the background and logs its output to the provided log file, in this case `/tmp/my-nodezero-runner.log`. The process is disconnected from the shell session, so it will continue to run in the background after you sign out.

Auto-Restart: If your Docker Host runs on a Linux system that supports `systemd`, you can optionally register the Runner as a system service with `systemd`. Once registered, `systemd` will automatically restart the Runner upon a system reboot. See Auto-start Runner at system startup for more info.

Naming: Each NodeZero Runner is given a name, in this case `my-nodezero-runner`. The name can be anything you want, but it should be unique if you spin up multiple Runners across your network. The name helps you identify the Runner when assigning pentests to it.

To verify the Runner has connected to the H3 API and registered itself, run the command below:

```
h3 runners
```

You should see an entry for the Runner `my-nodezero-runner` that you just started. You can now assign pentests to your NodeZero Runner from the Portal and the Runner will automatically launch NodeZero.

Additional notes about the Runner

- **Runs as:** The Runner process runs as the user that invoked `h3 start-runner`.
- **Background process:** The Runner process is disconnected from the shell session and runs in the background. It continues to run after the shell session is closed.
- **Auto-Restart:** To enable auto-restart of the Runner on system reboot, see Auto-start Runner at system startup.

- **Runner Log**: To view the Runner log, use `h3 tail-runner {name}`
- **Stop Runner:** To terminate the Runner process, use `h3 stop-runner {name}`.
- **Delete Runner:** To delete a Runner, use `h3 delete-runner {name}`.
- This only deletes its registration record with H3. If a deleted Runner is still active, it will be re-registered upon its next heartbeat.
- **Unique Runner names:** Runner names should be treated as unique identifiers. Avoid re-using the same name for different Runners in your account.
- **Rename Runner:** You can NOT rename an existing Runner; however you can stop (and optionally delete) a Runner, then start a new Runner with a different name.
- **NOTE:** if you saved the old Runner name to an op template, the template will need to be updated to use the new Runner name.
- **Auto-Injecting Credentials:** Runners can also be used to auto-inject credentials into a pentest.

Troubleshooting

The NodeZero Runner polls the H3 API every 60s. The last polling time is reported in the output of `h3 runners`, in the `last_heartbeat_at` field. If the Runner's last heartbeat is more than a minute ago, then the Runner process has either terminated or lost connectivity to the H3 API. In which case, sign-in to your Docker Host and check the health and connectivity of the Runner.

Check if the Runner process is alive:

```
h3 ps-runner
```

This will list the Runner processes on the local machine.

Look for errors in the log:

```
h3 tail-runner {runner_name}
```

View Runner command errors

Use the following to list out the last 5 commands executed by the Runner. The output includes the exit status and output from the command:

```
h3 runner-commands {runner_name}
```

Docker permission errors

Example:

```
[#] Checking Docker functionality by running the hello-world test container:
[+] PASSED: Docker version installed meets the minimum required version 20.10.
[!] FAILED: Failed to validate Docker. Verify this account has permissions to run Docker and retry.
```

If your Docker Host requires `sudo` to run `docker` commands, then you may need to start the Runner using `sudo` as well.

Alternatively, you try adding the user that invokes `h3 start-runner` to the `docker` group, for example (using `ubuntu` user):

```
sudo usermod -aG docker ubuntu
sudo systemctl restart docker
```

(Note: Make sure to log out and back in after changing groups for the actively logged on user) Retry

After resolving issues with your Runner, you can retry a NodeZero deployment by directly queuing a request to the Runner using the following command. Substitute `{op_id}` and `{runner_name}` for your specific usage.

```
h3 run-nodezero-on-runner {op_id} {runner_name}
```

You can use `h3 pentest` to get the `op_id` for the most recently created pentest.

Auto-start Runner at system startup

A common use case is to register your Runner to start automatically at system startup. This ensures that your Runner will continue working even after the system is rebooted.

In order to do this you must register the Runner with the system's boot service. Different systems have different boot services, but the most popular one used by various Linux distributions is `systemd`.

h3-cli provides built-in support for `systemd`. To register your Runner with `systemd`, use the `h3 start-runner-service` command, example below. Note the command will attempt to use `sudo` for running the necessary `systemd` / `systemctl` commands.

```
h3 start-runner-service my-nodezero-runner /tmp/my-nodezero-runner.log
```

If your system uses a different boot service than `systemd`, contact us for assistance with setting up your NodeZero Runner service.

If all goes well, your NodeZero Runner is now registered as a service with `systemd`. This means the Runner will...

- automatically start up at boot time
- automatically be restarted if it fails for any reason

The Runner service will be registered with `systemd` under the name `nodezero-runner-{runner-name}`. For example the command above will register the service as `nodezero-runner-my-nodezero-runner`.

Helpful `systemctl` commands for managing the Runner service

To view the status of the Runner service, use `systemctl status`, for example:

```
systemctl status nodezero-runner-my-nodezero-runner
```

To stop the Runner service:

```
sudo systemctl stop nodezero-runner-my-nodezero-runner
```

Note this will stop the current Runner service, but it will not de-register it from `systemd`. This means the Runner service will be started again upon the next system boot. To disable the Runner service such that it no longer starts at boot time, use `systemctl disable`, for example:

```
sudo systemctl disable nodezero-runner-my-nodezero-runner
```

To re-register the Runner service and re-enable it to start at boot time:

```
sudo systemctl enable nodezero-runner-my-nodezero-runner
```

To start the Runner service:

```
sudo systemctl start nodezero-runner-my-nodezero-runner
```

For more information related to `systemd` and `systemctl`, check out these resources:

- How to use systemctl to manage Linux services
- `systemctl` man page
- `systemd` wiki

Troubleshooting

- **209/STDOUT Error**: This typically means `systemd` could not start the service because it did not have permission to write to the log file (`/tmp/my-nodezero-runner.log` in the examples above). Try chmod'ing the log file or simply delete it and let `systemd` create a new one.

SCHEDULING FROM CLI

h3-cli: Automated scheduling using h3-cli

A common use case for h3-cli is running pentests automatically on a recurring basis, for example once a week or once a month, without any required user intervention - no need to log into the Portal, no need to copy+paste NodeZero launch scripts.

The instructions below walk through how to use h3-cli to configure a pentest to run automatically on a regular basis.

Note: You can also create pentest schedules directly in the Portal. See here for more info.

1. Enable automated NodeZero deployment

If you plan to run internal pentests (which are the default), you will need to enable h3-cli to automatically deploy NodeZero on a Docker Host inside your network.

This is done using a NodeZero Runner. The NodeZero Runner is a background process running on your Docker Host that automatically launches NodeZero whenever a new pentest is assigned to it.

To set up a NodeZero Runner, follow the instructions here.

2. Create a scheduled action to run a pentest on a recurring schedule

The command below will create a recurring schedule called `my-schedule` that will automatically run a pentest every Monday at 5pm UTC.

```
h3 create-scheduled-action \
    my-schedule \
    '0 17 * * 1' \
    run-pentest '{"schedule_op_form":{"op_name":"Pentest created via h3-cli and launched via runner", "runner_name":"my-nodezero-runner"}}'
```

The command uses a CRON expression, `0 17 * * 1`, to specify the recurring schedule for the pentest.

Breaking it down:

- `my-schedule` is the name of the schedule. A schedule may contain multiple actions. For example, you can configure timing windows for pentesting around by business hours, by scheduling a pentest to launch on Monday at 5pm, pause every day at 8am, resume at 5pm, and finally terminate on Friday if it is still running.
- `0 17 * * 1` is the CRON expression. CRON expressions specify the `{minute} {hour} {day-of-month} {month} {day-of-week}` to run a given action. Visit the [link]((https://crontab.guru/){:target="_blank"} for more information about CRON expressions.
- **NOTE:** Only hourly resolution is supported. The `{minute}` component of the CRON expression is always forced to be `0` on the backend.
- CRON expressions are in **UTC** time. So the example CRON expression above is set to **5pm UTC**.
- `run-pentest` is the action. Supported actions are:
- `run-pentest`: launches a new pentest (if one is not currently active for this schedule)
- `pause-pentest`: pauses the active pentest associated with the schedule
- `resume-pentest`: resumes the active pentest associated with the schedule
- `cancel-pentest`: cancel the active pentest associated with the schedule
- `'{"schedule_op_form":{"op_name":"Pentest created via h3-cli and launched via runner", "runner_name":"my-nodezero-runner"}}'`: additional parameters for the `run-pentest` action. These parameters are the same as those you would use if you executed `h3 run-pentest` directly from the command line.

A named schedule can have **only one active pentest at a time**. This prevents a schedule from kicking off a new pentest when its previous pentest has not yet completed.

Troubleshooting: `[403] You are not authorized`

If you receive the error `[403] You are not authorized`, you may be trying to use your **NodeZero Runner** API key to create the pentest schedule. For security reasons, NodeZero Runners have restricted permissions. All they can do (more or less)

is run NodeZero for an already created pentest. They cannot create new pentests, view pentest results, or create pentest schedules.

To create a pentest schedule, you'll need to create a separate API key with **User** permissions. You can then create a separate h3-cli profile for the new API key, and easily switch between multiple h3-cli profiles as needed.

Learn more about managing h3-cli profiles here.

For further assistance, contact H3 support via the chat icon in the Portal.

3. Verify your schedule is registered with H3

Use the following command to view your pentest schedules:

```
h3 schedules
```

If all is well, you should see an entry for your schedule `my-schedule`.

4. Test your scheduled action by triggering it now

To ensure everything is wired up as expected, you can trigger your scheduled action immediately with the following command:

```
h3 trigger-scheduled-action my-schedule run-pentest
```

This will trigger the `run-pentest` action for the `my-schedule` schedule, which will cause a pentest to be created. The NodeZero Runner on your NodeZero Docker Host will see the new pentest and automatically launch NodeZero.

You can monitor the NodeZero Runner process by tailing the log:

```
tail -f /tmp/my-nodezero-runner.log
```

In a minute or so you should see the Runner kick off the NodeZero Launch Script for the newly created pentest. The NodeZero Launch Script will download and launch the NodeZero Docker container on the local machine, just as if you had copy+pasted the `curl` command from Run Pentest wizard in the Portal.

You can view the newly created pentest via:

```
h3 pentest
```

Side note: If for whatever reason you need to kill the NodeZero Launch Script before it downloads and launches NodeZero, you can use `pkill`:

```
pkill -f h3-run-nodezero
```

Once the NodeZero Docker container is running, you can manage its lifecycle via the Docker API.

5. Troubleshooting

If you don't see NodeZero get launched in the NodeZero Runner log, use `h3 schedules` to see if any errors occurred when the action was triggered:

```
h3 schedules
```

The command output will resemble the readout below. Look at the `last_triggered_*` fields to help diagnose any problems:

```
{
  "name": "my-schedule",
  "state": "ENABLED",
  "created_at": "2023-02-06T06:52:04.660895",
  "last_updated_at": "2023-02-10T23:16:36.722392",
  "actions": [
    {
      "action": "run-pentest",
      "params": {
        "schedule_op_form": {
            "op_name": "Pentest created via h3-cli and launched via runner",
            "runner_name": "my-nodezero-runner"
        }
```

```
      },
      "cron_expression": "0 17 * * 1",
      "cron_description": "At 05:00 PM, only on Monday",
      "last_triggered_at": "2023-02-10T22:08:05.010069",
      "last_triggered_time_ago": "an hour ago",
      "last_triggered_error": null
    }
  ]
}
```


You will also receive an email notification every time a scheduled action is triggered. If the action fails, the error will be included in the email.

For further assistance, contact H3 support via the chat icon in the Portal.

6. Create a second scheduled action to cancel the pentest (optional)

Let's add a second action to our schedule for canceling the pentest. We'll schedule it to run 1hr after the pentest is launched.

```
h3 create-scheduled-action my-schedule '0 18 * * 1' cancel-pentest
```

Once again we can test the action by triggering it immediately:

```
h3 trigger-scheduled-action my-schedule cancel-pentest
```

After a moment you should see your pentest get canceled and move into the post-processing state.

7. Enabling and disabling a schedule

You can view all of your schedules via:

```
h3 schedules
```

You can disable a schedule and all its actions via `disable-schedule`:

```
h3 disable-schedule my-schedule
```

And you can re-enable a schedule via `enable-schedule`:

```
h3 enable-schedule my-schedule
```

8. Configuring pentesting windows

You can use scheduled actions to pause and resume pentests around pentesting windows, e.g. around business hours. The commands below show how to create a schedule that will:

- launch pentests on Mondays at 5pm UTC
- pause the running pentest every weekday at 8am UTC
- resume the paused pentest every weekday at 5pm UTC
- cancel the pentest if it's still running on Friday at 8am UTC

```
h3 create-scheduled-action my-schedule '0 17 * * 1' run-pentest '{"schedule_op_form":{"op_name":"Auto-scheduled weekly pentest", "runner_name":"my-nodezero-runner"}}'
h3 create-scheduled-action my-schedule '0 8 * * 2-4' pause-pentest
h3 create-scheduled-action my-schedule '0 17 * * 2-4' resume-pentest
h3 create-scheduled-action my-schedule '0 8 * * 5' cancel-pentest
```

9. Updating and deleting scheduled actions

You can update a scheduled action by simply running the `create-scheduled-action` again with the new settings. For example, if you wish to change the schedule above such that it cancels the pentest at 7am UTC instead of 8am:

```
h3 create-scheduled-action my-schedule '0 7 * * 5' cancel-pentest
```

Or if you wish to delete a scheduled action, use the `delete-scheduled-action` command:

```
h3 delete-scheduled-action my-schedule cancel-pentest
```

10. Creating multiple schedules

You can create multiple schedules by simply using a different schedule name. For example, here's a separate schedule named `my-weekend-schedule` that launches a pentest on Friday at 5pm UTC and cancels it Monday at 8am UTC:

```
h3 create-scheduled-action my-weekend-schedule '0 17 * * 5' run-pentest '{"schedule_op_form":{"op_name":"Weekend Pentest", "runner_name":"my-nodezero-runner"}}'
h3 create-scheduled-action my-weekend-schedule '0 8 * * 1' cancel-pentest
```

MONITOR PENTESTS

h3-cli: Monitoring pentests

You can use h3-cli to monitor the status of a pentest. This is done by periodically polling the API to check on the pentest `state`.

Monitoring enables you to trigger downstream actions or alerts when a pentest completes. A pentest is fully complete when its `state` hits `done` or `ended`.

Here's an example shell script that periodically polls the `state` until the pentest is complete.

```
#!/bin/bash

# loop forever until the pentest reaches 'done' or 'ended' state.
while [ 1 ]; do
    res=`h3 pentest`    # returns the most recent pentest
    pentest_state=`cat <<<$res | jq -r .state`
    pentest_name=`cat <<<$res | jq -r .name`
    if [ "$pentest_state" = "done" -o "$pentest_state" = "ended" ]; then
        echo "Pentest \"$pentest_name\" is complete; state=$pentest_state"
        break           # exit the loop
    fi
    echo "Pentest \"$pentest_name\" is still active; state=$pentest_state ..."
    sleep 30            # sleep 30 seconds then loop around and retry
done
```

PAGINATE RESULTS

h3-cli: Paginating results

Queries that return potentially a lot of results can be paginated by using the optional `page_input` on the GraphQL request.

For example action_logs.graphql is parameterized to accept `page_num` and `page_size` as parameters. These parameters are passed to `page_input` within the query file.

```
h3 gql action_logs '{"op_id":"your-op-id-here", "page_num":1, "page_size":100}' | jq .
```

Here's an example shell script that paginates through the full result set. It exits the loop when the query returns no further results.

```
#!/bin/bash

#
# Helper function for building the JSON parameters
# for the GraphQL request.
#
function build_json_params {
    op_id=$1
    page_num=$2
    page_size=$3
    cat <<HERE
{"op_id":"$op_id", "page_num":$page_num, "page_size":$page_size}
HERE
}

#
# 1. take the op_id as a param to the shell script.
#
op_id=$1
page_num=1
page_size=100

#
# 2. read page by page until the request returns no further results.
#
while [ 1 ]; do
    json_params=`build_json_params $op_id $page_num $page_size`
    res=`h3 gql action_logs "$json_params"`
    len=`cat <<<$res | jq '.data.action_logs_page.action_logs | length'`
    echo "Read $len records on page $page_num"
    if [ -z "$len" -o $len -eq 0 ]; then
        break
    fi
    (( page_num++ ))
done
```

3.2.3 API Reference

API Reference

The Horizon3.ai API enables programmatic access to a subset of functionality available in the Horizon3.ai Portal. You must be an existing user to gain access to the Horizon3.ai API. Existing users may get started below.

1. Authentication - learn how to authenticate with the API
2. Getting Started - basic examples to get started with using the API
3. GraphQL API - documentation of schema types, queries, and mutations

> **Command Line Interface**
>
> Be sure to check out the Horizon3.ai CLI for a ready-to-use API client.

References:

1. Horizon3.ai Portal
2. GraphQL.org

API Authentication

Before sending requests to the GraphQL API, an authentication process must be performed. API authentication is achieved by obtaining a JWT from our `/auth` endpoint by using an API key.

> **Need an API key?**
>
> Users can obtain an API key from the Horizon3.ai Portal.

AUTHENTICATE

Examples for obtaining an authenticated JWT are given below. Each example sends an HTTP request to the Horizon3.ai auth endpoint, which gets referenced via the environment variable `H3_AUTH_URL`:

```
export H3_AUTH_URL=https://api.horizon3ai.com/v1/auth
```

Before proceeding, store your API key in environment variable `H3_API_KEY`:

```
export H3_API_KEY=<your-api-key>
```

curl Python Result

```
curl -X POST \
     -H "Content-Type: application/json" \
     -d '{"key": "'"$H3_API_KEY"'"}' \
     $H3_AUTH_URL
```

```
import os
import requests

url = os.environ["H3_AUTH_URL"]
response = requests.post(url, headers=headers, json={"key": os.environ["H3_API_KEY"]})
result = response.json() if response.status_code == 200 else None
```

```
{"token": "<your-api-token>"}
```

The resulting token can be used to send authenticated requests to the GraphQL API. Get started here:

1. GraphQL Getting Started - recommended if you are new to GraphQL
2. GraphQL API Reference - recommended if you are familiar with GraphQL

> **JWT expiration**
>
> Each JWT expires after **1 hour**. If a given JWT has expired, the `/graphql` endpoint will respond with the token expiration message shown below. If this occurs, obtain a new JWT by re-authenticating with the `/auth` endpoint.
>
> ```
> {"message":"The incoming token has expired"}
> ```

Status codes

HTTP status codes returned by the `/auth` endpoint are summarized below.

Code	Title	Description
`200`	Success	Response contains the JWT.
`400`	Bad Request	Malformed request, review the error message.
`401`	Unauthorized	Not authenticated due to invalid API key.
`5xx`	Internal Server Error	An unexpected error was encountered.

Getting Started with GraphQL

The motivation for using GraphQL is explained, and examples of common scenarios are provided to help get started with using the Horizon3.ai API. This will help develop an understanding of how to use the GraphQL API to interact with your data.

Authentication

Before getting start with the examples here, be sure to review the section on API Authentication.

Leverage the full power

Once you are comfortable interacting with the GraphQL API, head over to our API Reference for a complete list of queries and mutations available.

Questions or comments

To provide feedback, please use the form at the bottom of this page.

WHY GRAPHQL?

The ITOps, SecOps, and automations engineering worlds are filled with REST APIs. So why did we choose a GraphQL API?

Data is power. We have a good idea how you might be able to use it, but it would be wrong of us to believe that we know better than you. GraphQL gives our developers far more control and flexibility through the ability to shape API queries to meet their needs. We wanted to provide the same power to our API users.

GraphQL.org

If you are new to GraphQL, we highly recommended reviewing the learning material at GraphQL.org.

HTTP CLIENT

To send requests to the GraphQL API, an HTTP client is needed. This is mostly a personal choice and/or dependent on the systems you may be integrating with. For example, if you are trying to build your own HTML interface, you will likely need a client that runs in JavaScript to be compatible with internet browsers.

Here are a few examples of useful HTTP clients:

- curl - useful for command line applications
- Python requests - useful for building Python applications
- Altair GraphQL Client - useful for testing and building API requests
- Postman API Client - useful for testing and building API requests

Need a specific programming language?

HTTP clients exist for every major programming language. A quick internet search should reveal the most well-known HTTP client(s) in any programming language.

Command Line Interface

Also be sure to check out the Horizon3.ai CLI for a ready-to-use API client.

Status codes

HTTP status codes returned by the `/graphql` endpoint are summarized below.

Code	Title	Description
`200`	Success	Response contains the requested data.
`400`	Bad Request	Malformed request, see `errors` field in the response data.
`401`	Unauthorized	Not authenticated due to an invalid or expired JWT.
`403`	Forbidden	Not authorized to access the requested resource.
`5xx`	Internal Server Error	An unexpected error was encountered.

EXAMPLES

The examples here send HTTP requests to the Horizon3.ai GraphQL API endpoint, which gets referenced in the commands below via environment variable `H3_API_URL`:

```
export H3_API_URL=https://api.horizon3ai.com/v1/graphql
```

Before proceeding, first perform API authentication and store the JWT in environment variable `H3_API_JWT`:

```
export H3_API_JWT=<token-returned-from-auth>
```

Hello world

Let's start with the simple hello (world) query.

For documentation of this query, see API Reference for Queries > hello.

curl Python Result

```
curl \
  -X POST $H3_API_URL \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $H3_API_JWT" \
  -d '{"query": "query HelloWorld { hello }"}'
```

```
import os
import requests

query = '''
    query HelloWorld {
      hello
    }
'''
url = os.environ["H3_API_URL"]
headers = {"Authorization": f"Bearer {os.environ['H3_API_JWT']}"}
response = requests.post(url, headers=headers, json={"query": query})
result = response.json() if response.status_code == 200 else None
```

```
{
  "data": {
    "hello": "world!"
  }
}
```

Pentest data

Let's review how to fetch data associated with a given pentest, such as start/stop times, number of weaknesses found, credentials discovered, and hosts scanned. A lot more data can be queried than what is shown in this example, see the API Reference for more info.

For documentation of this query, see API Reference for Queries > pentest.

curl **Python** **Result**

```
curl \
  -X POST $H3_API_URL \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $H3_API_JWT" \
  -d @- <<HERE
{
  "query": "
    query GetPentest(\$op_id: String!) {
      pentest(op_id: \$op_id) {
        op_id
        op_type
        name
        state
        launched_at
        completed_at
        min_scope
        max_scope
        exclude_scope
        weaknesses_count
        credentials_count
        cred_access_count
      }
    }",
  "variables": {
    "op_id": "12341234-1234-1234-1234-123412341234"
  }
}
HERE
```

```
import os
import requests

query = '''
    query GetPentest($op_id: String!) {
      pentest(op_id: $op_id) {
        op_id
        op_type
        name
        state
        launched_at
        completed_at
        min_scope
        max_scope
        exclude_scope
        weaknesses_count
        credentials_count
        cred_access_count
      }
    }
'''
variables = {
    "op_id": "12341234-1234-1234-1234-123412341234"
}
url = os.environ["H3_API_URL"]
headers = {"Authorization": f"Bearer {os.environ['H3_API_JWT']}"}
response = requests.post(url, headers=headers, json={"query": query, "variables": variables})
result = response.json() if response.status_code == 200 else None
```


```
{
  "data": {
    "pentest": {
      "op_id": "12341234-1234-1234-1234-123412341234",
      "op_type": "NodeZero",
      "name": "Sample Pentest",
      "state": "done",
      "launched_at": "2022-04-25T14:41:18",
      "completed_at": "2022-04-25T15:23:21",
      "min_scope": null,
      "max_scope": null,
      "exclude_scope": [],
      "weaknesses_count": 53,
      "credentials_count": 33,
      "cred_access_count": 142
    }
  }
}
```

Need more or less data?

One of the benefits of GraphQL is the ability to only query the data you need. In this example, add or remove fields of the Pentest type to suite your needs. This avoids the issue of over-fetching and eliminates the difficulty of parsing the returned data.

Pentest reports

Let's review how to download a zip containing CSV data files and PDF reports associated with a given pentest. This approach to fetching data from the API is in contrast to the granular approach demonstrated in the previous section on querying pentest data.

For documentation of this query, see API Reference for Queries > pentest_reports_zip_url.

curl **Python** **Result**

```
curl \
  -X POST $H3_API_URL \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $H3_API_JWT" \
  -d @- <<HERE
{
  "query": "
    query GetPentestReports(\$input: OpInput!) {
      pentest_reports_zip_url(input: \$input)
    }",
  "variables": {
    "input": {
      "op_id": "12341234-1234-1234-1234-123412341234"
    }
  }
}
HERE
```

```
import os
import requests

query = '''
    query GetPentestReports($input: OpInput!) {
      pentest_reports_zip_url(input: $input)
    }
'''
variables = {
    "input": {
        "op_id": "12341234-1234-1234-1234-123412341234"
    }
}
url = os.environ["H3_API_URL"]
headers = {"Authorization": f"Bearer {os.environ['H3_API_JWT']}"}
response = requests.post(url, headers=headers, json={"query": query, "variables": variables})
result = response.json() if response.status_code == 200 else None
```

```
{
  "data": {
    "pentest_reports_zip_url": "<temporary-url-to-download-zip>"
  }
}
```

Paginate action logs

Some queries fetch lists of data, e.g. users, pentests, actions logs, etc., but the amount of data can be substantial. Pagination allows us to limit the quantity of data retrieved. Let's take a look at how this works by querying the two (2) most recent action logs for a given pentest.

For documentation of this query, see API Reference for Queries > action_logs_page.

curl Python Result

```
curl \
  -X POST $H3_API_URL \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $H3_API_JWT" \
  -d @- <<HERE
{
  "query": "
    query GetActionLogs(\$input: OpInput!, \$page_input: PageInput) {
      action_logs_page(input: \$input, page_input: \$page_input) {
        page_info {
          ...PageInfoFragment
        }
        action_logs {
          ...ActionLogFragment
        }
      }
    }

    fragment PageInfoFragment on PageInfo {
      page_num
      page_size
    }

    fragment ActionLogFragment on ActionLog {
      uuid
      endpoint_ip
      start_time
      end_time
      cmd
      module_id
      module_name
      module_description
    }
  ",
  "variables": {
    "input": {
      "op_id": "12341234-1234-1234-1234-123412341234"
    },
    "page_input": {
      "page_num": 1,
      "page_size": 2,
      "order_by": "start_time",
      "sort_order": "DESC"
    }
  }
}
HERE
```

```
import os
import requests

query = '''
    query GetActionLogs($input: OpInput!, $page_input: PageInput) {
      action_logs_page(input: $input, page_input: $page_input) {
        page_info {
          ...PageInfoFragment
        }
        action_logs {
          ...ActionLogFragment
        }
      }
    }

    fragment PageInfoFragment on PageInfo {
      page_num
      page_size
    }

    fragment ActionLogFragment on ActionLog {
      uuid
      endpoint_ip
      start_time
      end_time
      cmd
      module_id
      module_name
      module_description
    }
'''
variables = {
    "input": {
        "op_id": "12341234-1234-1234-1234-123412341234"
    },
    "page_input": {
        "page_num": 1,
        "page_size": 2,
        "order_by": "start_time",
        "sort_order": "DESC"
    }
}
url = os.environ["H3_API_URL"]
headers = {"Authorization": f"Bearer {os.environ['H3_API_JWT']}"}
response = requests.post(url, headers=headers, json={"query": query, "variables": variables})
result = response.json() if response.status_code == 200 else None
```

```
{
  "data": {
    "action_logs_page": {
      "page_info": {
        "page_num": 1,
        "page_size": 2
      },
```

GraphQL fragments

You may have noticed this example used fragments to succinctly define fields to query on a given type. To learn more, see GraphQL fragments from graphql.org.

Create internal pentest

Let's consider how we can create a new internal pentest via the API. This will require us to send a **mutation** to the GraphQL API, instead of a **query**. Mutations are used when creating, updating, or deleting resources, whereas queries are intended only for fetching existing resources.

Creating vs launching pentest

Creating a pentest is the process of configuring and preparing a pentest to be launched. The process of launching a pentest is handled separately, after creating the pentest, by deploying NodeZero in your environment. This is explained more in a later section.

For documentation of this query, see API Reference for Mutations > `schedule_op_template`.

curl **Python** **Result**

```
curl \
  -X POST $H3_API_URL \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $H3_API_JWT" \
  -d @- <<HERE
{
  "query": "
    mutation CreatePentest(
      \$op_template_name: String!
      \$op_name: String
    ) {
      schedule_op_template(
        op_template_name: \$op_template_name
        op_name: \$op_name
      ) {
        op {
          ...OpFragment
        }
      }
    }

    fragment OpFragment on Op {
      op_id
      op_name
      op_state
      op_type
      scheduled_timestamp_iso
      launched_timestamp_iso
      nodezero_script_url
    }
  ",
  "variables": {
    "op_template_name": "Default 1 - Recommended",
    "op_name": "Pentest created via API"
  }
}
HERE
```

```
import os
import requests

query = '''
    mutation CreatePentest(
      $op_template_name: String!
      $op_name: String
    ) {
      schedule_op_template(
        op_template_name: $op_template_name
        op_name: $op_name
      ) {
        op {
          ...OpFragment
        }
      }
    }

    fragment OpFragment on Op {
      op_id
      op_name
      op_state
      op_type
      scheduled_timestamp_iso
      launched_timestamp_iso
      nodezero_script_url
    }
'''
variables = {
    "op_template_name": "Default 1 - Recommended",
    "op_name": "Pentest created via API"
}
url = os.environ["H3_API_URL"]
headers = {"Authorization": f"Bearer {os.environ['H3_API_JWT']}"}
response = requests.post(url, headers=headers, json={"query": query, "variables": variables})
result = response.json() if response.status_code == 200 else None
```

```
{
  "data": {
    "schedule_op_template": {
      "op": {
        "op_id": "df087532-b6ca-48b1-bcd3-d8b1968a197f",
        "op_name": "Pentest created via API",
        "op_state": "scheduled",
        "op_type": "NodeZero",
        "scheduled_timestamp_iso": "2023-03-04T16:19:42",
        "launched_timestamp_iso": null,
        "nodezero_script_url": <url-to-download-launch-script>
      }
    }
  }
}
```

In example above, the **default** template, Default 1 - Recommended, was used to define the pentest configuration. To create your own op template, it is recommended to do so from the Horizon.ai Portal, but you may also use the API with Mutations > `save_op_template`.

What is an op template?

An "op template" refers to a predefined pentest configuration, including settings for hosts to scan, passwords to spray, and other attack config. You may create op templates for various use cases and environment(s). Each op template gets stored in your client account and can be used when creating pentests.

Ready for launch!

Continue reading to see how our newly created pentest can be **launched**. It is evident that the pentest has yet be launched based on the `null` value for `launched_timestamp_iso` in the result above.

Launch NodeZero

Once an internal pentest has been created, it is ready for NodeZero to deploy in the intended environment. To do so, simply copy the value returned in `nodezero_script_url` from the mutation above, then pass it to curl and pipe the downloaded NodeZero script to bash:

NodeZero host only

The launch script must be run from the NodeZero host inside your intended environment. For more information on configuring this host, see Setup NodeZero Host.

```
curl <nodezero_script_url> | bash
```

Once the command above successfully runs, the autonomous pentest has officially deployed. To monitor its progress and notable events, go to the pentest Real-Time View in the Horizon3.ai Portal.

Scheduling pentests

Despite the mutation name used above, i.e. `schedule_op_template`, it does **not** allow a pentest to be scheduled at a specific time. It only creates the pentest, preparing NodeZero for deployment in your environment.

CLI Tool

Our ready-to-use CLI Tool provides the ability to schedule recurring pentests and much more.

×

Close menu

Introduction

- Welcome

Operations

- Queries
- action_logs_count
- action_logs_csv_presigned_url
- action_logs_page
- activedir_passwords_csv_url
- agent
- agents
- agents_count
- asset_group
- asset_groups_count
- asset_groups_page
- client_accounts_count
- client_accounts_page
- external_domain_xop
- external_domain_xops_count
- hello
- host_tab_xop
- host_tab_xops_count
- hosts_csv
- hosts_csv_url
- op
- op_tabs_page
- op_template
- op_templates
- pentest
- pentest_reports_zip_url
- pentests_count
- pentests_page
- sample_op_tabs
- session_user_account
- weakness
- weaknesses_count
- weaknesses_csv
- weaknesses_csv_url

- Mutations
- add_domains_to_asset_group
- authorize_domains
- authorize_ips
- bulk_authorize_domains
- bulk_authorize_ips
- bulk_deauthorize_domains
- bulk_deauthorize_ips
- cancel_op
- create_asset_group
- create_client_account
- create_user_account
- deauthorize_domains
- deauthorize_ips
- delete_client_account
- delete_op_template
- delete_user_account
- pause_op
- remove_domains_from_asset_group
- resume_op
- save_op_template
- schedule_op_template
- update_asset_group_template
- update_client_account
- update_user_account
-

Types

- AWSAccountId
- AccessLevel
- ActionLog
- ActionLogsPage
- Agent
- AgentCommand
- AssetGroup
- AssetGroupOutput
- AssetGroupsPage
- AuthzRole
- BlockedPentestableEntity
- Boolean
- BrandColor
- BrandColorInput
- BrandColorType
- ClientAccount
- ClientAccountInput
- ClientAccountOutput
- ClientAccountUpdateInput
- ClientAccountsPage
- CreateUserAccountInput
- Date
- Datetime
- DeleteOpTemplateOutput
- Deleted
- EmailAddress
- ExcludedDomain
- ExcludedIP
- ExternalDomainXop
- FeatureFlag
- FeatureFlagInput
- FeatureFlagRiskType
- FilterBy
- FilterByInput
- Float
- GitAccount
- GitAccountInput
- GitAccountSource
- HexColor
- HostCSV
- HostTabXop
- ImpactType
- Int

- Long
- MitreMapping
- MitreSubtechnique
- MitreTactic
- MitreTechnique
- ModuleMeta
- Op
- OpInput
- OpTab
- OpTabsPage
- OpTemplate
- OpType
- PageInfo
- PageInput
- Pentest
- PentestableEntitiesBulkOutput
- PentestableEntitiesOutput
- PentestableEntity
- PentestableRules
- PentestsPage
- PortalOpState
- SaveOpTemplateOutput
- ScheduleOpForm
- ScheduleOpFormInput
- ScheduleOpOutput
- Severity
- SignInType
- SortInput
- SortOrder
- String
- StringNoWhitespace
- StringNotEmpty
- UpdateUserAccountInput
- UserAccount
- VulnCategory
- Weakness
- WeaknessCSV

All topics

GraphQL API Reference

Welcome to the Horizon3.ai GraphQL API reference! This reference includes the complete set of GraphQL types, queries, mutations, and their parameters available to users of Horizon3.ai products.

First, learn how to authenticate with the API. If you are new to GraphQL, see the Getting Started guide.

API Endpoints

```
https://api.horizon3ai.com/v1/graphql
```

Headers

```
Authorization: Bearer <YOUR_TOKEN_HERE>
```

Queries

`ACTION_LOGS_COUNT`

Description

Number of action log records for a given pentest.

The action log is the set of all commands executed by NodeZero across all hosts during the pentest.

Response

Returns an `Int!`

Arguments

Name	Description
`input` - `OpInput!`	Pentest to get action logs for.
`page_input` - `PageInput`	Pagination of action logs.

Example
Query

```
query action_logs_count
  $input  OpInput!
  $page_input  PageInput

  action_logs_count
           $input
                $page_input
```

Variables

```
{
  "input": OpInput,
  "page_input": PageInput
}
```

Response

```
{"data": {"action_logs_count": 987}}
```

Queries

`ACTION_LOGS_CSV_PRESIGNED_URL`

Description

Generates a temporary AWS presigned URL for downloading the action log as a CSV.

The action log is the set of all commands executed by NodeZero across all hosts during the pentest.

Response

Returns a `String`

Arguments

Name	Description
`input - OpInput!`	

Example

Query

```
query action_logs_csv_presigned_url $input  OpInput!
  action_logs_csv_presigned_url        $input
```

Variables

```
{"input": OpInput}
```

Response

```
{
  "data": {
    "action_logs_csv_presigned_url": "abc123"
  }
}
```

Queries

ACTION_LOGS_PAGE

Description

Paginated list of action log records for a given pentest.

The action log is the set of all commands executed by NodeZero across all hosts during the pentest.

Response

Returns an `ActionLogsPage!`

Arguments

Name	Description
`input` - `OpInput!`	Pentest to get action logs for.
`page_input` - `PageInput`	Pagination of action logs.

Example

Query

```
query action_logs_page
  $input  OpInput!
  $page_input  PageInput

  action_logs_page
           $input
                $page_input

    page_info
      ...PageInfoFragment

    action_logs
      ...ActionLogFragment
```

Variables

```
{
  "input": OpInput,
  "page_input": PageInput
}
```

Response

```
{
  "data": {
    "action_logs_page": {
      "page_info": PageInfo,
      "action_logs": [ActionLog]
    }
  }
}
```

Queries

`ACTIVEDIR_PASSWORDS_CSV_URL`

Description

List of Active Direction passwords found during an AD Password Audit. Returns a presigned URL to the CSV file. The presigned URL expires after a short time. The CSV format is documented under ActiveDirPasswordCSV.

Response

Returns a `String`

Arguments

Name	Description
`input` - `OpInput!`	

Example
Query

```
query activedir_passwords_csv_url $input  OpInput!
  activedir_passwords_csv_url        $input
```

Variables

```
{"input": OpInput}
```

Response

```
{
  "data": {
    "activedir_passwords_csv_url": "abc123"
  }
}
```

Queries

AGENT

Description

Fetch the agent aka NodeZero Runner with the given `uuid` or `name` in the current user's ClientAccount

Response

Returns an `Agent`

Arguments

Name	Description
`uuid` - `String`	
`name` - `String`	

Example

Query

```
query agent
  $uuid  String,
  $name  String

  agent
          $uuid
          $name

    uuid
    name
    uname
    log_file
    last_heartbeat_at
    last_heartbeat_time_ago
    last_command
      ...AgentCommandFragment

    commands
      ...AgentCommandFragment

    created_at
```

Variables

```
{
  "uuid": "abc123",
  "name": "abc123"
}
```

Response

```
{
  "data": {
    "agent": {
      "uuid": "12341234-1234-1234-1234-123412341234",
      "name": "abc123",
      "uname": "xyz789",
      "log_file": "abc123",
      "last_heartbeat_at": "2021-07-22T05:02:40.294996",
      "last_heartbeat_time_ago": "abc123",
      "last_command": AgentCommand,
      "commands": [AgentCommand],
      "created_at": "2021-07-22T05:02:40.294996"
    }
  }
}
```

Queries

`AGENTS`

Description

The list of h3-cli agents aka NodeZero Runners in the current user's ClientAccount

Response

Returns `[Agent]`

Example
Query

```
query agents
  agents
    uuid
    name
    uname
    log_file
    last_heartbeat_at
    last_heartbeat_time_ago
    last_command
      ...AgentCommandFragment

    commands
      ...AgentCommandFragment

    created_at
```

Response

```
{
  "data": {
    "agents": [
      {
        "uuid": "12341234-1234-1234-1234-123412341234",
        "name": "xyz789",
        "uname": "abc123",
        "log_file": "abc123",
        "last_heartbeat_at": "2021-07-22T05:02:40.294996",
        "last_heartbeat_time_ago": "abc123",
        "last_command": AgentCommand,
        "commands": [AgentCommand],
        "created_at": "2021-07-22T05:02:40.294996"
      }
    ]
  }
}
```

Queries

AGENTS_COUNT

Description

The count of h3-cli agents aka NodeZero Runners in the current user's ClientAccount

Response

Returns an `Int!`

Arguments

Name	Description
`page_input` - `PageInput`	

Example

Query

```
query agents_count $page_input  PageInput)
                          $page_input
}
```

Variables

```
{"page_input": PageInput}
```

Response

```
{"data": {"agents_count": 987}}
```

Queries

`ASSET_GROUP`

Description

Fetch a specific asset group from this account.

Response

Returns an `AssetGroup`

Arguments

Name	Description
`uuid` - `String!`	

Example

Query

```
query asset_group $uuid  String!
  asset_group       $uuid
    uuid
    name
    op_template_uuid
    op_template
      ...OpTemplateFragment

    user_account_uuid
    user_account_name
    client_account_uuid
    client_account_company_name
    last_ead_etl_completed_at
    created_at
    updated_at
    op_series_uuid
    assets_count
    authorized_assets_count
    external_domain_xops_count
    authorized_external_domain_xops_count
    in_scope_host_tab_xops_count
    authorized_host_tab_xops_count
```

Variables

```
{"uuid": "xyz789"}
```

Response

```
{
  "data": {
    "asset_group": {
      "uuid": "12341234-1234-1234-1234-123412341234",
      "name": "xyz789",
      "op_template_uuid": "12341234-1234-1234-1234-123412341234",
      "op_template": OpTemplate,
      "user_account_uuid": "12341234-1234-1234-1234-123412341234",
      "user_account_name": "abc123",
      "client_account_uuid": "12341234-1234-1234-1234-123412341234",
      "client_account_company_name": "xyz789",
      "last_ead_etl_completed_at": "2021-07-22T05:02:40.294996",
      "created_at": "2021-07-22T05:02:40.294996",
      "updated_at": "2021-07-22T05:02:40.294996",
      "op_series_uuid": "12341234-1234-1234-1234-123412341234",
      "assets_count": 987,
      "authorized_assets_count": 987,
      "external_domain_xops_count": 123,
      "authorized_external_domain_xops_count": 123,
      "in_scope_host_tab_xops_count": 987,
      "authorized_host_tab_xops_count": 987
    }
  }
}
```

Queries

ASSET_GROUPS_COUNT

Description

The number of asset groups in this account.

Response

Returns an `Int!`

Arguments

Name	Description
`page_input` - `PageInput`	

Example

Query

```
query asset_groups_count $page_input  PageInput)
                                $page_input
}
```

Variables

```
{"page_input": PageInput}
```

Response

```
{"data": {"asset_groups_count": 987}}
```

Queries

ASSET_GROUPS_PAGE

Description

Paginated list of asset groups in this account.

Response

Returns an `AssetGroupsPage!`

Arguments

Name	Description
`page_input` - `PageInput`	

Example

Query

```
query asset_groups_page $page_input  PageInput)
                                $page_input
    page_info
      ...PageInfoFragment

    asset_groups
      ...AssetGroupFragment


}
```

Variables

```
{"page_input": PageInput}
```

Response

```
{
  "data": {
    "asset_groups_page": {
      "page_info": PageInfo,
      "asset_groups": [AssetGroup]
    }
  }
}
```

Queries

`CLIENT_ACCOUNTS_COUNT`

Description

Number of client accounts accessible by the current user.

Response

Returns an `Int!`

Arguments

Name	Description
`page_input` - `PageInput`	

Example

Query

```
query client_accounts_count $page_input  PageInput)
                                $page_input
}
```

Variables

```
{"page_input": PageInput}
```

Response

```
{"data": {"client_accounts_count": 123}}
```

Queries

`CLIENT_ACCOUNTS_PAGE`

Description

Client accounts accessible by the current user.

Response

Returns a `ClientAccountsPage!`

Arguments

Name	Description
`page_input` - `PageInput`	

Example

Query

```
query client_accounts_page $page_input  PageInput)
                                  $page_input
    page_info
      ...PageInfoFragment

    client_accounts
      ...ClientAccountFragment


}
```

Variables

```
{"page_input": PageInput}
```

Response

```
{
  "data": {
    "client_accounts_page": {
      "page_info": PageInfo,
      "client_accounts": [ClientAccount]
    }
  }
}
```

Queries

`EXTERNAL_DOMAIN_XOP`

Description

Returns the ExternalDomainXop with the given `uuid`.

Response

Returns an `ExternalDomainXop`

Arguments

Name	Description
`uuid` - `String!`	

Example

Query

```
query external_domain_xop $uuid  String!
  external_domain_xop       $uuid
    uuid
    op_series_uuid
    xop_id
    last_op_id
    current_op_id
    is_authorized
    pentestable_rules
      ...PentestableRulesFragment

    is_dynamic_ip
    excluded_domain_from_last_pentest
      ...ExcludedDomainFragment

    third_party_aliases
    third_party_certificate_subject_cns
```

Variables

```
{"uuid": "xyz789"}
```

Response

```
{
  "data": {
    "external_domain_xop": {
      "uuid": "12341234-1234-1234-1234-123412341234",
      "op_series_uuid": "12341234-1234-1234-1234-123412341234",
      "xop_id": "abc123",
      "last_op_id": "abc123",
      "current_op_id": "abc123",
      "is_authorized": true,
      "pentestable_rules": PentestableRules,
      "is_dynamic_ip": "123.45.67.89",
      "excluded_domain_from_last_pentest": ExcludedDomain,
      "third_party_aliases": ["xyz789"],
      "third_party_certificate_subject_cns": [
        "xyz789"
      ]
    }
  }
}
```

Queries

EXTERNAL_DOMAIN_XOPS_COUNT

Description

Count of domains in the given `AssetGroup.op_series_uuid`

Response

Returns an `Int!`

Arguments

Name	Description
`op_series_uuid` - `String!`	
`page_input` - `PageInput`	

Example

Query

```
query external_domain_xops_count
  $op_series_uuid  String!
  $page_input  PageInput

  external_domain_xops_count
                   $op_series_uuid
               $page_input
```

Variables

```
{
  "op_series_uuid": "abc123",
  "page_input": PageInput
}
```

Response

```
{"data": {"external_domain_xops_count": 987}}
```

Queries

HELLO

Description

Hello world example.

Response

Returns a `String!`

Example
Query

```
query hello
  hello
```

Response

```
{"data": {"hello": "xyz789"}}
```

Queries

HOST_TAB_XOP

Description

Returns the HostTabXop with the given `uuid`.

Response

Returns a `HostTabXop`

Arguments

Name	Description
`uuid` - `String!`	

Example

Query

```
query host_tab_xop $uuid  String!
  host_tab_xop       $uuid
    uuid
    op_series_uuid
    xop_id
    ip
    last_op_id
    current_op_id
    is_authorized
    excluded_ip_from_last_pentest
      ...ExcludedIPFragment

    pentestable_rules
      ...PentestableRulesFragment

    third_party_aliases
    third_party_certificate_subject_cns
```

Variables

```
{"uuid": "xyz789"}
```

Response

```
{
  "data": {
    "host_tab_xop": {
      "uuid": "12341234-1234-1234-1234-123412341234",
      "op_series_uuid": "12341234-1234-1234-1234-123412341234",
      "xop_id": "xyz789",
      "ip": "123.45.67.89",
      "last_op_id": "abc123",
      "current_op_id": "abc123",
      "is_authorized": true,
      "excluded_ip_from_last_pentest": ExcludedIP,
      "pentestable_rules": PentestableRules,
      "third_party_aliases": ["abc123"],
      "third_party_certificate_subject_cns": [
        "xyz789"
      ]
    }
  }
}
```

Queries

HOST_TAB_XOPS_COUNT

Description

Count of IPs in the given `AssetGroup.op_series_uuid`

Response

Returns an `Int!`

Arguments

Name	Description
`op_series_uuid` - `String!`	
`page_input` - `PageInput`	

Example

Query

```
query host_tab_xops_count
  $op_series_uuid  String!
  $page_input  PageInput

  host_tab_xops_count
                  $op_series_uuid
               $page_input
```

Variables

```
{
  "op_series_uuid": "xyz789",
  "page_input": PageInput
}
```

Response

```
{"data": {"host_tab_xops_count": 987}}
```

Queries

`HOSTS_CSV`

use `Query.hosts_csv_url`
Description

List of hosts found during a pentest. Returned as an array of comma-separated values, with each element in the array representing a row. Each element in array represents a single host, with the first element containing the column names.

Response

Returns `[HostCSV]`

Arguments

Name	Description
`input` - `OpInput!`	Pentest to get hosts data for.

Example
Query

```
query hosts_csv $input  OpInput!
  hosts_csv        $input
```

Variables

```
{"input": OpInput}
```

Response

```
{"data": {"hosts_csv": [HostCSV]}}
```

Queries

`HOSTS_CSV_URL`

Description

List of hosts found during a pentest. Returns a presigned URL to the CSV file. The presigned URL expires after a short time. The CSV format is documented under HostCSV.

Response

Returns a `String`

Arguments

Name	Description
`input` - `OpInput!`	

Example
Query

```
query hosts_csv_url $input  OpInput!
  hosts_csv_url        $input
```

Variables

```
{"input": OpInput}
```

Response

```
{"data": {"hosts_csv_url": "abc123"}}
```

Queries

OP

Description

Get pentest data.

Response

Returns an `Op`

Arguments

Name	Description
`op_id` - `String!`	ID of pentest.

Example

Query

```
query op $op_id  String!
  op        $op_id
    op_id
    op_type
    op_state
    op_name
    scheduled_timestamp_iso
    scheduled_at
    scheduled_at_date
    completed_timestamp_iso
    launched_timestamp_iso
    confirmed_credentials_count
    weaknesses_count
    in_scope_hosts_count
    feature_flags
      ...FeatureFlagFragment

    nodezero_script_url
    duration_hms
    duration_humanize
    op_template_name
    impact_paths_count
    runner_name
    runner
      ...AgentFragment

    schedule_name
```

Variables

```
{"op_id": "xyz789"}
```

Response

```
{
  "data": {
    "op": {
      "op_id": "12341234-1234-1234-1234-123412341234",
      "op_type": "NodeZero",
      "op_state": "running",
      "op_name": "your op name",
      "scheduled_timestamp_iso": "2021-07-22T05:02:40.294996",
      "scheduled_at": "2021-07-22T05:02:40.294996",
      "scheduled_at_date": "2023-11-01T18:52:27.602Z",
      "completed_timestamp_iso": "2021-07-22T05:02:40.294996",
      "launched_timestamp_iso": "2021-07-22T05:02:40.294996",
      "confirmed_credentials_count": 123,
      "weaknesses_count": 123,
      "in_scope_hosts_count": 987,
      "feature_flags": [FeatureFlag],
      "nodezero_script_url": "https://example.com/example",
      "duration_hms": "22:05:21",
      "duration_humanize": "2 hours, 23 minutes",
      "op_template_name": "abc123",
      "impact_paths_count": 987,
      "runner_name": "abc123",
      "runner": Agent,
      "schedule_name": "abc123"
    }
  }
}
```

Queries

`OP_TABS_PAGE`

Description

Get a list of pentests for client accounts accessible by the current user.

Response

Returns an `OpTabsPage!`

Arguments

Name	Description
`page_input` - `PageInput`	Pagination of pentests.
`exclude_sample_ops` - `Boolean`	Exclude sample pentests from the result.

Example

Query

```
query op_tabs_page
  $page_input  PageInput,
  $exclude_sample_ops  Boolean

  op_tabs_page
                $page_input
                        $exclude_sample_ops

    page_info
      ...PageInfoFragment

    op_tabs
      ...OpTabFragment
```

Variables

```
{"page_input": PageInput, "exclude_sample_ops": true}
```

Response

```
{
  "data": {
    "op_tabs_page": {
      "page_info": PageInfo,
      "op_tabs": [OpTab]
    }
  }
}
```

Queries

`OP_TEMPLATE`

Description

Get an op template.

Response

Returns an `OpTemplate`

Arguments

Name	Description
`op_template_uuid` - `String!`	The uuid of the op template

Example

Query

```
query op_template $op_template_uuid  String!
  op_template                   $op_template_uuid
    uuid
    user_account_uuid
    client_account_uuid
    op_template_name
    op_type
    schedule_op_form
      ...ScheduleOpFormFragment

    row_created_at
    row_updated_at
```

Variables

```
{"op_template_uuid": "xyz789"}
```

Response

```
{
  "data": {
    "op_template": {
      "uuid": "12341234-1234-1234-1234-123412341234",
      "user_account_uuid": "12341234-1234-1234-1234-123412341234",
      "client_account_uuid": "12341234-1234-1234-1234-123412341234",
      "op_template_name": "xyz789",
      "op_type": "NodeZero",
      "schedule_op_form": ScheduleOpForm,
      "row_created_at": "2021-07-22T05:02:40.294996",
      "row_updated_at": "2021-07-22T05:02:40.294996"
    }
  }
}
```

Queries

OP_TEMPLATES

Description

List of op templates, of a specified op type, for the current client account. Includes default templates provided by Horizon3.ai.

Response

Returns `[OpTemplate]`

Arguments

Name	Description
`op_type` - `String`	Optional. Must be either "NodeZero" (for internal pentests) or "ExternalAttack" (for external pentests). Defaults to "NodeZero".

Example
Query

```
query op_templates $op_type  String)
                      $op_type
    uuid
    user_account_uuid
    client_account_uuid
    op_template_name
    op_type
    schedule_op_form
      ...ScheduleOpFormFragment

    row_created_at
    row_updated_at

}
```

Variables

```
{"op_type": "xyz789"}
```

Response

```
{
  "data": {
    "op_templates": [
      {
        "uuid": "12341234-1234-1234-1234-123412341234",
        "user_account_uuid": "12341234-1234-1234-1234-123412341234",
        "client_account_uuid": "12341234-1234-1234-1234-123412341234",
        "op_template_name": "xyz789",
        "op_type": "NodeZero",
        "schedule_op_form": ScheduleOpForm,
        "row_created_at": "2021-07-22T05:02:40.294996",
        "row_updated_at": "2021-07-22T05:02:40.294996"
      }
    ]
  }
}
```

Queries

PENTEST

Description

Get pentest data.

Response

Returns a `Pentest`

Arguments

Name	Description
`op_id` - `String!`	ID of pentest.

Example

Query

```
query pentest $op_id  String!
  pentest        $op_id
    op_id
    op_type
    name
    state
    user_name
    client_name
    min_scope
    max_scope
    exclude_scope
    git_accounts
      ...GitAccountFragment

    aws_account_ids
    feature_flags
      ...FeatureFlagFragment

    scheduled_at
    launched_at
    completed_at
    canceled_at
    etl_completed_at
    duration_s
    impacts_count
    impact_paths_count
    weakness_types_count
    weaknesses_count
    hosts_count
    out_of_scope_hosts_count
    external_domains_count
    services_count
    credentials_count
    users_count
    cred_access_count
    data_stores_count
    websites_count
    data_resources_count
    nodezero_script_url
    nodezero_ip
    runner
      ...AgentFragment
```

Variables

```
{"op_id": "abc123"}
```

Response

```
{
  "data": {
    "pentest": {
      "op_id": "12341234-1234-1234-1234-123412341234",
      "op_type": "NodeZero",
      "name": "abc123",
      "state": "done",
      "user_name": "xyz789",
      "client_name": "xyz789",
      "min_scope": ["xyz789"],
      "max_scope": ["xyz789"],
      "exclude_scope": ["abc123"],
```

```
      "git_accounts": [GitAccount],
      "aws_account_ids": [AWSAccountId],
      "feature_flags": [FeatureFlag],
      "scheduled_at": "2021-07-22T05:02:40.294996",
      "launched_at": "2021-07-22T05:02:40.294996",
      "completed_at": "2021-07-22T05:02:40.294996",
      "canceled_at": "2021-07-22T05:02:40.294996",
      "etl_completed_at": "2021-07-22T05:02:40.294996",
      "duration_s": 123,
      "impacts_count": 123,
      "impact_paths_count": 123,
      "weakness_types_count": 987,
      "weaknesses_count": 123,
      "hosts_count": 123,
      "out_of_scope_hosts_count": 123,
      "external_domains_count": 987,
      "services_count": 123,
      "credentials_count": 987,
      "users_count": 123,
      "cred_access_count": 987,
      "data_stores_count": 987,
      "websites_count": 987,
      "data_resources_count": Long,
      "nodezero_script_url": "https://example.com/example",
      "nodezero_ip": "123.45.67.89",
      "runner": Agent
    }
  }
}
```

Queries

`PENTEST_REPORTS_ZIP_URL`

Description

Returns a URL to a zip file containing all CSVs and PDFs for the given `op_id`.

The CSVs along with their schema types are listed below:

- certificates.csv: CertificateCSV
- credentials.csv: CredentialCSV
- data_stores.csv: ShareCSV
- database_repos.csv: DatabaseRepoCSV
- docker_registries.csv: DockerRegistryCSV
- external_domains.csv: ExternalDomainCSV
- file_shares.csv: FileShareCSV
- git_repos.csv: GitRepoCSV
- hosts.csv: HostCSV
- impacts.csv: ImpactCSV
- s3_buckets.csv: S3BucketCSV
- services.csv: ServiceCSV
- users.csv: UserCSV
- weaknesses.csv: WeaknessCSV
- websites.csv: WebShareCSV

Note: for pentests prior to Sep 2022, the zip file is generated "lazily"; ie. on demand, when the user first request it. The first request will receive a response saying the zip file is being generated. An email will be sent to the requesting user with a link to the zip file when it is ready. Subsequent requests will download the already-built zip file.

Response

Returns a `String`

Arguments

Name	Description
`input` - `OpInput!`	

Example
Query

```
query pentest_reports_zip_url $input  OpInput!
  pentest_reports_zip_url        $input
```

Variables

```
{"input": OpInput}
```

Response

```
{
  "data": {
    "pentest_reports_zip_url": "xyz789"
  }
}
```

Queries

`PENTESTS_COUNT`

Description

Counts the number of pentests in the current client account, after applying any filters in `page_input`. By default archived pentests are excluded.

Response

Returns an `Int!`

Arguments

Name	Description
`page_input` - `PageInput`	Pagination of pentests.

Example
Query

```
query pentests_count $page_input  PageInput)
                            $page_input
}
```

Variables

```
{"page_input": PageInput}
```

Response

```
{"data": {"pentests_count": 123}}
```

Queries

PENTESTS_PAGE

Description

Paginated list of pentests in the current client account. By default archived pentests are excluded.

Response

Returns a `PentestsPage!`

Arguments

Name	Description
`page_input` - `PageInput`	Pagination of pentests.

Example

Query

```
query pentests_page $page_input  PageInput)
                            $page_input
    page_info
      ...PageInfoFragment

    pentests
      ...PentestFragment


}
```

Variables

```
{"page_input": PageInput}
```

Response

```
{
  "data": {
    "pentests_page": {
      "page_info": PageInfo,
      "pentests": [Pentest]
    }
  }
}
```

Queries

SAMPLE_OP_TABS

Description

Get a list of sample pentests for the current client account.

Response

Returns `[OpTab!]`

Example
Query

```
query sample_op_tabs
  sample_op_tabs
    uuid
    op_id
    op_state
    op_name
    scheduled_timestamp
    scheduled_at
    scheduled_at_date
    scheduled_timestamp_iso
    create_timestamp
    create_timestamp_iso
    launched_timestamp
    launched_timestamp_iso
    launched_at
    completed_at
    completed_timestamp
    completed_timestamp_iso
    canceled_at
    canceled_timestamp
    canceled_timestamp_iso
    duration_hms
    duration_humanize
    op_type
    weakness_types_count
    weaknesses_count
    host_tabs_count
    domain_controllers_count
    credentials_count
    proven_credentials_count
    confirmed_credentials_count
    unproven_credentials_count
    activedir_passwords_count
    enabled_activedir_passwords_count
    disabled_activedir_passwords_count
    feature_flags
      ...FeatureFlagFragment

    impacts_headline_count
    impact_paths_count
    nodezero_script_url
    nodezero_ip
    etl_completed_at
    start_paused
    minimum_run_time
    maximum_run_time
    paused_at
    paused_by_user_account_uuid
    paused_by_user_account
      ...UserAccountFragment

    op_template_name
    excluded_ips
      ...ExcludedIPFragment

    excluded_domains
      ...ExcludedDomainFragment

    runner_name
    runner
      ...AgentFragment

    run_nodezero_command
      ...AgentCommandFragment

    schedule_name
    auto_injected_credential_uuids
```

Response


```
{
  "data": {
    "sample_op_tabs": [
      {
        "uuid": "12341234-1234-1234-1234-123412341234",
        "op_id": "12341234-1234-1234-1234-123412341234",
        "op_state": "running",
        "op_name": "your op name",
        "scheduled_timestamp": "1600793100.0",
        "scheduled_at": "2021-07-22T05:02:40.294996",
        "scheduled_at_date": "2024-05-01T18:52:27.602Z",
        "scheduled_timestamp_iso": "2021-07-22T05:02:40.294996",
        "create_timestamp": "1600793100.0",
        "create_timestamp_iso": "2021-07-22T05:02:40.294996",
        "launched_timestamp": "1600793100.0",
        "launched_timestamp_iso": "2021-07-22T05:02:40.294996",
        "launched_at": "2021-07-22T05:02:40.294996",
        "completed_at": "2021-07-22T05:02:40.294996",
        "completed_timestamp": "1600793100.0",
        "completed_timestamp_iso": "2021-07-22T05:02:40.294996",
        "canceled_at": "2021-07-22T05:02:40.294996",
        "canceled_timestamp": "1600793100.0",
        "canceled_timestamp_iso": "2021-07-22T05:02:40.294996",
        "duration_hms": "22:35:09",
        "duration_humanize": "2 hours, 23 minutes",
        "op_type": "NodeZero",
        "weakness_types_count": 123,
        "weaknesses_count": 987,
        "host_tabs_count": 987,
        "domain_controllers_count": 123,
        "credentials_count": 123,
        "proven_credentials_count": 987,
        "confirmed_credentials_count": 123,
        "unproven_credentials_count": 987,
        "activedir_passwords_count": 123,
        "enabled_activedir_passwords_count": 987,
        "disabled_activedir_passwords_count": 123,
        "feature_flags": [FeatureFlag],
        "impacts_headline_count": 123,
        "impact_paths_count": 123,
        "nodezero_script_url": "https://example.com/example",
        "nodezero_ip": "123.45.67.89",
        "etl_completed_at": "2021-07-22T05:02:40.294996",
        "start_paused": true,
        "minimum_run_time": 987,
        "maximum_run_time": 123,
        "paused_at": "2021-07-22T05:02:40.294996",
        "paused_by_user_account_uuid": "12341234-1234-1234-1234-123412341234",
        "paused_by_user_account": UserAccount,
        "op_template_name": "xyz789",
        "excluded_ips": [ExcludedIP],
        "excluded_domains": [ExcludedDomain],
        "runner_name": "abc123",
        "runner": Agent,
        "run_nodezero_command": AgentCommand,
        "schedule_name": "xyz789",
        "auto_injected_credential_uuids": [
          "abc123"
        ]
      }
    ]
  }
}
```

Queries

`SESSION_USER_ACCOUNT`

Description

Current user account.

Response

Returns a `UserAccount`

Example
Query

```
query session_user_account
  session_user_account
    uuid
    email
    name
    user_role_id
    sign_in_type
    last_signed_in_at
```

Response

```
{
  "data": {
    "session_user_account": {
      "uuid": "12341234-1234-1234-1234-123412341234",
      "email": "john.smith@example.com",
      "name": "John Smith",
      "user_role_id": "USER",
      "sign_in_type": "BASIC",
      "last_signed_in_at": "2021-07-22T05:02:40.294996"
    }
  }
}
```

Queries

`WEAKNESS`

Description

The Weakness with the given `uuid`.

Response

Returns a `Weakness`

Arguments

Name	Description
`uuid` - `String!`	

Example
Query

```
query weakness $uuid  String!
  weakness       $uuid
    uuid
    created_at
    vuln_id
    vuln_aliases
    vuln_category
    vuln_name
    vuln_short_name
    vuln_cisa_kev
    vuln_known_ransomware_campaign_use
    op_id
    ip
    has_proof
    proof_failure_code
    proof_failure_reason
    score
    severity
    base_score
    base_severity
    context_score
    context_severity
    context_score_description_md
    context_score_description
    time_to_finding_hms
    time_to_finding_s
    affected_asset_text
    affected_asset_short_text
    downstream_impact_types
    downstream_impact_types_and_counts
    impact_paths_count
```

Variables

```
{"uuid": "abc123"}
```

Response

```
{
  "data": {
    "weakness": {
      "uuid": "12341234-1234-1234-1234-123412341234",
      "created_at": "2021-07-22T05:02:40.294996",
      "vuln_id": "abc123",
      "vuln_aliases": ["abc123"],
      "vuln_category": "SECURITY_MISCONFIGURATION",
      "vuln_name": "abc123",
      "vuln_short_name": "xyz789",
      "vuln_cisa_kev": true,
      "vuln_known_ransomware_campaign_use": false,
      "op_id": "12341234-1234-1234-1234-123412341234",
      "ip": "123.45.67.89",
      "has_proof": false,
      "proof_failure_code": "xyz789",
      "proof_failure_reason": "xyz789",
      "score": 987.65,
      "severity": "INFO",
      "base_score": 123.45,
      "base_severity": "INFO",
      "context_score": 123.45,
      "context_severity": "INFO",
      "context_score_description_md": "xyz789",
```

```
      "context_score_description": "xyz789",
      "time_to_finding_hms": "22:05:31",
      "time_to_finding_s": 123,
      "affected_asset_text": "abc123",
      "affected_asset_short_text": "xyz789",
      "downstream_impact_types": ["AWSUserRoleCompromise"],
      "downstream_impact_types_and_counts": [
        "xyz789"
      ],
      "impact_paths_count": 123
    }
  }
}
```

Queries

`WEAKNESSES_COUNT`

Description

The count of Weaknesses observed during the given op.

A Weakness record represents an observed vulnerability on an affected asset; ie. there is one Weakness record per unique vuln_id + affected asset combination.

Response

Returns an `Int!`

Arguments

Name	Description
`input` - `OpInput!`	
`page_input` - `PageInput`	

Example

Query

```
query weaknesses_count
  $input  OpInput!
  $page_input  PageInput

  weaknesses_count
           $input
                $page_input
```

Variables

```
{
  "input": OpInput,
  "page_input": PageInput
}
```

Response

```
{"data": {"weaknesses_count": 123}}
```

Queries

`WEAKNESSES_CSV`

Use Query.weaknesses_csv_url
Description

The set of Weaknesses observed during the given op, as a set of CSV records.

A Weakness record represents an observed vulnerability on an affected asset; ie. there is one Weakness record per unique vuln_id + affected asset combination.

Returned as an array of strings, with each string being a single record of comma-separated values. Eah string is equivalent to a line in a CSV file.

The CSV format is documented under WeaknessCSV.

The first string contains the header line, with the list of column names.

DEPRECATED: this API will eventually be removed, in favor of `Query.weaknesses_csv_url`.

Response

Returns `[WeaknessCSV]`

Arguments

Name	Description
`input` - `OpInput!`	Pentest to get weakness data for.

Example
Query

```
query weaknesses_csv $input  OpInput!
  weaknesses_csv        $input
```

Variables

```
{"input": OpInput}
```

Response

```
{"data": {"weaknesses_csv": [WeaknessCSV]}}
```

Queries

`WEAKNESSES_CSV_URL`

Description

The set of Weaknesses observed during the given op, as a set of CSV records.

A Weakness record represents an observed vulnerability on an affected asset; ie. there is one Weakness record per unique vuln_id + affected asset combination.

Returned as an array of strings, with each string being a single record of comma-separated values. Eah string is equivalent to a line in a CSV file.

The CSV format is documented under WeaknessCSV.

The first string contains the header line, with the list of column names.

Response

Returns a `String`

Arguments

Name	Description
`input` - `OpInput!`	

Example
Query

```
query weaknesses_csv_url $input  OpInput!
  weaknesses_csv_url        $input
```

Variables

```
{"input": OpInput}
```

Response

```
{"data": {"weaknesses_csv_url": "xyz789"}}
```

Mutations

`ADD_DOMAINS_TO_ASSET_GROUP`

Description

Update an asset group with additional domains.

Response

Returns an `AssetGroupOutput!`

Arguments

Name	Description
`asset_group_uuid` - `String!`	ID of asset group to update.
`domains` - `[StringNotEmpty]!`	List of domains to add to the asset group's scope.

Example
Query

```
mutation add_domains_to_asset_group
  $asset_group_uuid  String!
  $domains  [StringNotEmpty]

  add_domains_to_asset_group
                    $asset_group_uuid
```

```
            $domains

    asset_group
      ...AssetGroupFragment
```

Variables

```
{
  "asset_group_uuid": "abc123",
  "domains": [StringNotEmpty]
}
```

Response

```
{
  "data": {
    "add_domains_to_asset_group": {
      "asset_group": AssetGroup
    }
  }
}
```

Mutations

AUTHORIZE_DOMAINS

Description

Authorize the given AssetGroup domains for external pentesting.

Response

Returns a `PentestableEntitiesOutput!`

Arguments

Name	Description
`external_domain_xop_uuids - [String!]!`	

Example

Query

```
mutation authorize_domains $external_domain_xop_uuids  [String!
  authorize_domains                              $external_domain_xop_uuids
    pentestable_entities
      ...PentestableEntityFragment

    blocked_pentestable_entities
      ...BlockedPentestableEntityFragment
```

Variables

```
{"external_domain_xop_uuids": ["abc123"]}
```

Response

```
{
  "data": {
    "authorize_domains": {
      "pentestable_entities": [PentestableEntity],
      "blocked_pentestable_entities": [
        BlockedPentestableEntity
      ]
    }
  }
}
```

Mutations

AUTHORIZE_IPS

Description

Authorize the given AssetGroup IPs for external pentesting.

Response

Returns a `PentestableEntitiesOutput!`

Arguments

Name	Description
`host_tab_xop_uuids` - `[String!]!`	

Example

Query

```
mutation authorize_ips $host_tab_xop_uuids  [String!
  authorize_ips                     $host_tab_xop_uuids
    pentestable_entities
      ...PentestableEntityFragment

    blocked_pentestable_entities
      ...BlockedPentestableEntityFragment
```

Variables

```
{"host_tab_xop_uuids": ["abc123"]}
```

Response

```
{
  "data": {
    "authorize_ips": {
      "pentestable_entities": [PentestableEntity],
      "blocked_pentestable_entities": [
        BlockedPentestableEntity
      ]
    }
  }
}
```

Mutations

BULK_AUTHORIZE_DOMAINS

Description

Bulk authorize AssetGroup domains for external pentesting.

Response

Returns a `PentestableEntitiesBulkOutput!`

Arguments

Name	Description
`op_series_uuid` - `String!`	ID of op series.
`configured_domains_only` - `Boolean`	If true, only domains configured in the AssetGroup's scope will be authorized. Default is false.
`page_input` - `PageInput`	If filters are provided, only domains that match the filters will be authorized.

Example

Query

```
mutation bulk_authorize_domains
  $op_series_uuid  String!
  $configured_domains_only  Boolean,
  $page_input  PageInput

  bulk_authorize_domains
                    $op_series_uuid
                              $configured_domains_only
                $page_input

    pentestable_entities_count
```

Variables

```
{
  "op_series_uuid": "abc123",
  "configured_domains_only": false,
  "page_input": PageInput
}
```

Response

```
{"data": {"bulk_authorize_domains": {"pentestable_entities_count": 123}}}
```

Mutations

BULK_AUTHORIZE_IPS

Description

Bulk authorize AssetGroup IPs for external pentesting.

Response

Returns a `PentestableEntitiesBulkOutput!`

Arguments

Name	Description
`op_series_uuid` - `String!`	ID of op series.
`page_input` - `PageInput`	If filters are provided, only IPs that match the filters will be authorized.

Example

Query

```
mutation bulk_authorize_ips
  $op_series_uuid  String!
  $page_input  PageInput

  bulk_authorize_ips
                   $op_series_uuid
                $page_input

    pentestable_entities_count
```

Variables

```
{
  "op_series_uuid": "xyz789",
  "page_input": PageInput
}
```

Response

```
{"data": {"bulk_authorize_ips": {"pentestable_entities_count": 123}}}
```

Mutations

BULK_DEAUTHORIZE_DOMAINS

Description

Bulk DE-authorize AssetGroup domains for external pentesting.

Response

Returns a `PentestableEntitiesBulkOutput!`

Arguments

Name	Description
`op_series_uuid` - `String!`	ID of op series.
`page_input` - `PageInput`	If filters are provided, only domains that match the filters will be de-authorized.

Example

Query

```
mutation bulk_deauthorize_domains
  $op_series_uuid  String!
  $page_input  PageInput

  bulk_deauthorize_domains
                   $op_series_uuid
               $page_input

    pentestable_entities_count
```

Variables

```
{
  "op_series_uuid": "abc123",
  "page_input": PageInput
}
```

Response

```
{"data": {"bulk_deauthorize_domains": {"pentestable_entities_count": 123}}}
```

Mutations

BULK_DEAUTHORIZE_IPS

Description

Bulk DE-authorize AssetGroup IPs for external pentesting.

Response

Returns a `PentestableEntitiesBulkOutput!`

Arguments

Name	Description
`op_series_uuid` - `String!`	ID of op series.
`page_input` - `PageInput`	If filters are provided, only IPs that match the filters will be de-authorized.

Example

Query

```
mutation bulk_deauthorize_ips
  $op_series_uuid  String!
  $page_input  PageInput

  bulk_deauthorize_ips
                   $op_series_uuid
               $page_input

    pentestable_entities_count
```

Variables

```
{
  "op_series_uuid": "abc123",
  "page_input": PageInput
}
```

Response

```
{"data": {"bulk_deauthorize_ips": {"pentestable_entities_count": 987}}}
```

Mutations

`CANCEL_OP`

Description

Cancel a pentest, if its current state allows it.

Response

Returns an `OpTab!`

Arguments

Name	Description
`op_id - String!`	

Example

Query

```
mutation cancel_op $op_id  String!
  cancel_op        $op_id
    uuid
    op_id
    op_state
    op_name
    scheduled_timestamp
    scheduled_at
    scheduled_at_date
    scheduled_timestamp_iso
    create_timestamp
    create_timestamp_iso
    launched_timestamp
    launched_timestamp_iso
    launched_at
    completed_at
    completed_timestamp
    completed_timestamp_iso
    canceled_at
    canceled_timestamp
    canceled_timestamp_iso
    duration_hms
    duration_humanize
    op_type
    weakness_types_count
    weaknesses_count
    host_tabs_count
    domain_controllers_count
    credentials_count
    proven_credentials_count
    confirmed_credentials_count
    unproven_credentials_count
    activedir_passwords_count
    enabled_activedir_passwords_count
    disabled_activedir_passwords_count
    feature_flags
      ...FeatureFlagFragment

    impacts_headline_count
    impact_paths_count
    nodezero_script_url
    nodezero_ip
    etl_completed_at
    start_paused
    minimum_run_time
    maximum_run_time
    paused_at
    paused_by_user_account_uuid
    paused_by_user_account
      ...UserAccountFragment

    op_template_name
    excluded_ips
      ...ExcludedIPFragment

    excluded_domains
      ...ExcludedDomainFragment

    runner_name
    runner
      ...AgentFragment

    run_nodezero_command
      ...AgentCommandFragment

    schedule_name
    auto_injected_credential_uuids
```

Variables


```
{"op_id": "abc123"}
```


Response


```
{
  "data": {
    "cancel_op": {
      "uuid": "12341234-1234-1234-1234-123412341234",
      "op_id": "12341234-1234-1234-1234-123412341234",
      "op_state": "running",
      "op_name": "your op name",
      "scheduled_timestamp": "1600793100.0",
      "scheduled_at": "2021-07-22T05:02:40.294996",
      "scheduled_at_date": "2024-05-01T18:52:27.602Z",
      "scheduled_timestamp_iso": "2021-07-22T05:02:40.294996",
      "create_timestamp": "1600793100.0",
      "create_timestamp_iso": "2021-07-22T05:02:40.294996",
      "launched_timestamp": "1600793100.0",
      "launched_timestamp_iso": "2021-07-22T05:02:40.294996",
      "launched_at": "2021-07-22T05:02:40.294996",
      "completed_at": "2021-07-22T05:02:40.294996",
      "completed_timestamp": "1600793100.0",
      "completed_timestamp_iso": "2021-07-22T05:02:40.294996",
      "canceled_at": "2021-07-22T05:02:40.294996",
      "canceled_timestamp": "1600793100.0",
      "canceled_timestamp_iso": "2021-07-22T05:02:40.294996",
      "duration_hms": "22:35:09",
      "duration_humanize": "2 hours, 23 minutes",
      "op_type": "NodeZero",
      "weakness_types_count": 123,
      "weaknesses_count": 987,
      "host_tabs_count": 987,
      "domain_controllers_count": 987,
      "credentials_count": 123,
      "proven_credentials_count": 123,
      "confirmed_credentials_count": 123,
      "unproven_credentials_count": 987,
      "activedir_passwords_count": 987,
      "enabled_activedir_passwords_count": 987,
      "disabled_activedir_passwords_count": 123,
      "feature_flags": [FeatureFlag],
      "impacts_headline_count": 123,
      "impact_paths_count": 123,
      "nodezero_script_url": "https://example.com/example",
      "nodezero_ip": "123.45.67.89",
      "etl_completed_at": "2021-07-22T05:02:40.294996",
      "start_paused": false,
      "minimum_run_time": 123,
      "maximum_run_time": 987,
      "paused_at": "2021-07-22T05:02:40.294996",
      "paused_by_user_account_uuid": "12341234-1234-1234-1234-123412341234",
      "paused_by_user_account": UserAccount,
      "op_template_name": "xyz789",
      "excluded_ips": [ExcludedIP],
      "excluded_domains": [ExcludedDomain],
      "runner_name": "xyz789",
      "runner": Agent,
      "run_nodezero_command": AgentCommand,
      "schedule_name": "xyz789",
      "auto_injected_credential_uuids": [
        "abc123"
      ]
    }
  }
}
```

Mutations

`CREATE_ASSET_GROUP`

Description

Create an asset group.
The asset group uses the scope configured in the given `schedule_op_form` to discover assets.

Response

Returns an `AssetGroupOutput!`

Arguments

Name	Description
`schedule_op_form` - `ScheduleOpFormInput!`	Scope for discovering assets.

Example
Query

```
mutation create_asset_group $schedule_op_form  ScheduleOpFormInput!
  create_asset_group                  $schedule_op_form
    asset_group
      ...AssetGroupFragment
```

Variables

```
{"schedule_op_form": ScheduleOpFormInput}
```

Response

```
{
  "data": {
    "create_asset_group": {"asset_group": AssetGroup}
  }
}
```

Mutations

CREATE_CLIENT_ACCOUNT

Description

Create a client account and grant the current user access.

Response

Returns a `ClientAccountOutput!`

Arguments

Name	Description
`client_account_input` - `ClientAccountInput!`	

Example

Query

```
mutation create_client_account $client_account_input  ClientAccountInput!
  create_client_account                       $client_account_input
    client_account
      ...ClientAccountFragment
```

Variables

```
{"client_account_input": ClientAccountInput}
```

Response

```
{
  "data": {
    "create_client_account": {
      "client_account": ClientAccount
    }
  }
}
```

Mutations

`CREATE_USER_ACCOUNT`

Description

Add a user to a given client account. The user will receive an invitation email along with temporary login credentials, if needed.

Response

Returns a `UserAccount`

Arguments

Name	Description
`input` - `CreateUserAccountInput!`	Input data for creating user.

Example
Query

```
mutation create_user_account $input  CreateUserAccountInput!
  create_user_account        $input
    uuid
    email
    name
    user_role_id
    sign_in_type
    last_signed_in_at
```

Variables

```
{"input": CreateUserAccountInput}
```

Response

```
{
  "data": {
    "create_user_account": {
      "uuid": "12341234-1234-1234-1234-123412341234",
      "email": "john.smith@example.com",
      "name": "John Smith",
      "user_role_id": "USER",
      "sign_in_type": "BASIC",
      "last_signed_in_at": "2021-07-22T05:02:40.294996"
    }
  }
}
```

Mutations

DEAUTHORIZE_DOMAINS

Description

DE-authorize the given AssetGroup domains for external pentesting.

Response

Returns a `PentestableEntitiesOutput!`

Arguments

Name	Description
`external_domain_xop_uuids - [String!]!`	

Example

Query

```
mutation deauthorize_domains $external_domain_xop_uuids  [String!
  deauthorize_domains                                    $external_domain_xop_uuids
    pentestable_entities
      ...PentestableEntityFragment

    blocked_pentestable_entities
      ...BlockedPentestableEntityFragment
```

Variables

```
{"external_domain_xop_uuids": ["xyz789"]}
```

Response

```
{
  "data": {
    "deauthorize_domains": {
      "pentestable_entities": [PentestableEntity],
      "blocked_pentestable_entities": [
        BlockedPentestableEntity
      ]
    }
  }
}
```

Mutations

DEAUTHORIZE_IPS

Description

DE-authorize the given AssetGroup IPs for external pentesting.

Response

Returns a `PentestableEntitiesOutput!`

Arguments

Name	Description
`host_tab_xop_uuids` - `[String!]!`	

Example

Query

```
mutation deauthorize_ips $host_tab_xop_uuids  [String!
  deauthorize_ips                     $host_tab_xop_uuids
    pentestable_entities
      ...PentestableEntityFragment

    blocked_pentestable_entities
      ...BlockedPentestableEntityFragment
```

Variables

```
{"host_tab_xop_uuids": ["abc123"]}
```

Response

```
{
  "data": {
    "deauthorize_ips": {
      "pentestable_entities": [PentestableEntity],
      "blocked_pentestable_entities": [
        BlockedPentestableEntity
      ]
    }
  }
}
```

Mutations

`DELETE_CLIENT_ACCOUNT`

Description

Delete a client account.

Response

Returns a `Deleted!`

Arguments

Name	Description
`client_account_uuid` - `String!`	Client account to delete.

Example

Query

```
mutation delete_client_account $client_account_uuid  String!
  delete_client_account                      $client_account_uuid
    success
```

Variables

```
{"client_account_uuid": "abc123"}
```

Response

```
{"data": {"delete_client_account": {"success": false}}}
```

Mutations

DELETE_OP_TEMPLATE

Description

Delete a pentest (aka "op") template.

Response

Returns a `DeleteOpTemplateOutput!`

Arguments

Name	Description
`op_template_name` - `String!`	Name of pentest template.
`op_type` - `String`	Type of pentest, see `OpType` for options.

Example

Query

```
mutation delete_op_template
  $op_template_name  String!
  $op_type  String

  delete_op_template
                     $op_template_name
            $op_type

    op_template
      ...OpTemplateFragment
```

Variables

```
{
  "op_template_name": "xyz789",
  "op_type": "xyz789"
}
```

Response

```
{
  "data": {
    "delete_op_template": {"op_template": OpTemplate}
  }
}
```

Mutations

DELETE_USER_ACCOUNT

Description

Delete a user from a given client account.

Response

Returns a `UserAccount`

Arguments

Name	Description
`email` - `EmailAddress!`	Email address of user to delete.
`client_account_uuid` - `String`	ID of client account to delete the user from. Defaults to the caller's client account.

Example

Query

```
mutation delete_user_account
  $email  EmailAddress!
  $client_account_uuid  String

  delete_user_account
           $email
                          $client_account_uuid

    uuid
    email
    name
    user_role_id
    sign_in_type
    last_signed_in_at
```

Variables

```
{
  "email": EmailAddress,
  "client_account_uuid": "xyz789"
}
```

Response

```
{
  "data": {
    "delete_user_account": {
      "uuid": "12341234-1234-1234-1234-123412341234",
      "email": "john.smith@example.com",
      "name": "John Smith",
      "user_role_id": "USER",
      "sign_in_type": "BASIC",
      "last_signed_in_at": "2021-07-22T05:02:40.294996"
    }
  }
}
```

Mutations

`PAUSE_OP`

Description

Pause a pentest, if its current state allows it.

Response

Returns an `OpTab!`

Arguments

Name	Description
`op_id - String!`	

Example

Query

```
mutation pause_op $op_id  String!
  pause_op        $op_id
    uuid
    op_id
    op_state
    op_name
    scheduled_timestamp
    scheduled_at
    scheduled_at_date
    scheduled_timestamp_iso
    create_timestamp
    create_timestamp_iso
    launched_timestamp
    launched_timestamp_iso
    launched_at
    completed_at
    completed_timestamp
    completed_timestamp_iso
    canceled_at
    canceled_timestamp
    canceled_timestamp_iso
    duration_hms
    duration_humanize
    op_type
    weakness_types_count
    weaknesses_count
    host_tabs_count
    domain_controllers_count
    credentials_count
    proven_credentials_count
    confirmed_credentials_count
    unproven_credentials_count
    activedir_passwords_count
    enabled_activedir_passwords_count
    disabled_activedir_passwords_count
    feature_flags
      ...FeatureFlagFragment

    impacts_headline_count
    impact_paths_count
    nodezero_script_url
    nodezero_ip
    etl_completed_at
    start_paused
    minimum_run_time
    maximum_run_time
    paused_at
    paused_by_user_account_uuid
    paused_by_user_account
      ...UserAccountFragment

    op_template_name
    excluded_ips
      ...ExcludedIPFragment

    excluded_domains
      ...ExcludedDomainFragment

    runner_name
    runner
      ...AgentFragment

    run_nodezero_command
      ...AgentCommandFragment

    schedule_name
    auto_injected_credential_uuids
```

Variables

```
{"op_id": "abc123"}
```

Response

```
{
  "data": {
    "pause_op": {
      "uuid": "12341234-1234-1234-1234-123412341234",
      "op_id": "12341234-1234-1234-1234-123412341234",
      "op_state": "running",
      "op_name": "your op name",
      "scheduled_timestamp": "1600793100.0",
      "scheduled_at": "2021-07-22T05:02:40.294996",
      "scheduled_at_date": "2023-11-01T18:52:27.602Z",
      "scheduled_timestamp_iso": "2021-07-22T05:02:40.294996",
      "create_timestamp": "1600793100.0",
      "create_timestamp_iso": "2021-07-22T05:02:40.294996",
      "launched_timestamp": "1600793100.0",
      "launched_timestamp_iso": "2021-07-22T05:02:40.294996",
      "launched_at": "2021-07-22T05:02:40.294996",
      "completed_at": "2021-07-22T05:02:40.294996",
      "completed_timestamp": "1600793100.0",
      "completed_timestamp_iso": "2021-07-22T05:02:40.294996",
      "canceled_at": "2021-07-22T05:02:40.294996",
      "canceled_timestamp": "1600793100.0",
      "canceled_timestamp_iso": "2021-07-22T05:02:40.294996",
      "duration_hms": "22:35:09",
      "duration_humanize": "2 hours, 23 minutes",
      "op_type": "NodeZero",
      "weakness_types_count": 987,
      "weaknesses_count": 987,
      "host_tabs_count": 123,
      "domain_controllers_count": 987,
      "credentials_count": 123,
      "proven_credentials_count": 987,
      "confirmed_credentials_count": 987,
      "unproven_credentials_count": 987,
      "activedir_passwords_count": 987,
      "enabled_activedir_passwords_count": 987,
      "disabled_activedir_passwords_count": 123,
      "feature_flags": [FeatureFlag],
      "impacts_headline_count": 123,
      "impact_paths_count": 987,
      "nodezero_script_url": "https://example.com/example",
      "nodezero_ip": "123.45.67.89",
      "etl_completed_at": "2021-07-22T05:02:40.294996",
      "start_paused": false,
      "minimum_run_time": 987,
      "maximum_run_time": 987,
      "paused_at": "2021-07-22T05:02:40.294996",
      "paused_by_user_account_uuid": "12341234-1234-1234-1234-123412341234",
      "paused_by_user_account": UserAccount,
      "op_template_name": "abc123",
      "excluded_ips": [ExcludedIP],
      "excluded_domains": [ExcludedDomain],
      "runner_name": "abc123",
      "runner": Agent,
      "run_nodezero_command": AgentCommand,
      "schedule_name": "xyz789",
      "auto_injected_credential_uuids": [
        "xyz789"
      ]
    }
  }
}
```

Mutations

`REMOVE_DOMAINS_FROM_ASSET_GROUP`

Description

Remove domains from an asset group.

Response

Returns an `AssetGroupOutput!`

Arguments

Name	Description
`asset_group_uuid` - `String!`	ID of asset group to update.
`domains` - `[StringNotEmpty]!`	List of domains to remove from the asset group's scope.

Example

Query

```
mutation remove_domains_from_asset_group
  $asset_group_uuid  String!
  $domains  [StringNotEmpty]

  remove_domains_from_asset_group
                     $asset_group_uuid
            $domains

    asset_group
      ...AssetGroupFragment
```

Variables

```
{
  "asset_group_uuid": "xyz789",
  "domains": [StringNotEmpty]
}
```

Response

```
{
  "data": {
    "remove_domains_from_asset_group": {
      "asset_group": AssetGroup
    }
  }
}
```

Mutations

`RESUME_OP`

Description

Resume a pentest, if its current state allows it.

Response

Returns an `OpTab!`

Arguments

Name	Description
`op_id - String!`	

Example
Query

```
mutation resume_op $op_id  String!
  resume_op        $op_id
    uuid
    op_id
    op_state
    op_name
    scheduled_timestamp
    scheduled_at
    scheduled_at_date
    scheduled_timestamp_iso
    create_timestamp
    create_timestamp_iso
    launched_timestamp
    launched_timestamp_iso
    launched_at
    completed_at
    completed_timestamp
    completed_timestamp_iso
    canceled_at
    canceled_timestamp
    canceled_timestamp_iso
    duration_hms
    duration_humanize
    op_type
    weakness_types_count
    weaknesses_count
    host_tabs_count
    domain_controllers_count
    credentials_count
    proven_credentials_count
    confirmed_credentials_count
    unproven_credentials_count
    activedir_passwords_count
    enabled_activedir_passwords_count
    disabled_activedir_passwords_count
    feature_flags
      ...FeatureFlagFragment

    impacts_headline_count
    impact_paths_count
    nodezero_script_url
    nodezero_ip
    etl_completed_at
    start_paused
    minimum_run_time
    maximum_run_time
    paused_at
    paused_by_user_account_uuid
    paused_by_user_account
      ...UserAccountFragment

    op_template_name
    excluded_ips
      ...ExcludedIPFragment

    excluded_domains
      ...ExcludedDomainFragment

    runner_name
    runner
      ...AgentFragment

    run_nodezero_command
      ...AgentCommandFragment

    schedule_name
    auto_injected_credential_uuids
```

Variables

```
{"op_id": "abc123"}
```

Response

```
{
  "data": {
    "resume_op": {
      "uuid": "12341234-1234-1234-1234-123412341234",
      "op_id": "12341234-1234-1234-1234-123412341234",
      "op_state": "running",
      "op_name": "your op name",
      "scheduled_timestamp": "1600793100.0",
      "scheduled_at": "2021-07-22T05:02:40.294996",
      "scheduled_at_date": "2023-11-01T18:52:27.602Z",
      "scheduled_timestamp_iso": "2021-07-22T05:02:40.294996",
      "create_timestamp": "1600793100.0",
      "create_timestamp_iso": "2021-07-22T05:02:40.294996",
      "launched_timestamp": "1600793100.0",
      "launched_timestamp_iso": "2021-07-22T05:02:40.294996",
      "launched_at": "2021-07-22T05:02:40.294996",
      "completed_at": "2021-07-22T05:02:40.294996",
      "completed_timestamp": "1600793100.0",
      "completed_timestamp_iso": "2021-07-22T05:02:40.294996",
      "canceled_at": "2021-07-22T05:02:40.294996",
      "canceled_timestamp": "1600793100.0",
      "canceled_timestamp_iso": "2021-07-22T05:02:40.294996",
      "duration_hms": "22:35:09",
      "duration_humanize": "2 hours, 23 minutes",
      "op_type": "NodeZero",
      "weakness_types_count": 987,
      "weaknesses_count": 123,
      "host_tabs_count": 987,
      "domain_controllers_count": 987,
      "credentials_count": 987,
      "proven_credentials_count": 123,
      "confirmed_credentials_count": 987,
      "unproven_credentials_count": 987,
      "activedir_passwords_count": 123,
      "enabled_activedir_passwords_count": 123,
      "disabled_activedir_passwords_count": 987,
      "feature_flags": [FeatureFlag],
      "impacts_headline_count": 987,
      "impact_paths_count": 987,
      "nodezero_script_url": "https://example.com/example",
      "nodezero_ip": "123.45.67.89",
      "etl_completed_at": "2021-07-22T05:02:40.294996",
      "start_paused": true,
      "minimum_run_time": 987,
      "maximum_run_time": 123,
      "paused_at": "2021-07-22T05:02:40.294996",
      "paused_by_user_account_uuid": "12341234-1234-1234-1234-123412341234",
      "paused_by_user_account": UserAccount,
      "op_template_name": "xyz789",
      "excluded_ips": [ExcludedIP],
      "excluded_domains": [ExcludedDomain],
      "runner_name": "xyz789",
      "runner": Agent,
      "run_nodezero_command": AgentCommand,
      "schedule_name": "xyz789",
      "auto_injected_credential_uuids": [
        "abc123"
      ]
    }
  }
}
```

Mutations

SAVE_OP_TEMPLATE

Description

Create or update a pentest (aka "op") template.

Response

Returns a `SaveOpTemplateOutput!`

Arguments

Name	Description
`op_template_name` - `String!`	Name of pentest template. The name uniquely identifies the template in the client account.
`schedule_op_form` - `ScheduleOpFormInput!`	Input data for scheduling pentest.

Example

Query

```
mutation save_op_template
  $op_template_name  String!
  $schedule_op_form  ScheduleOpFormInput!

  save_op_template
                     $op_template_name
                     $schedule_op_form

    op_template
      ...OpTemplateFragment
```

Variables

```
{
  "op_template_name": "abc123",
  "schedule_op_form": ScheduleOpFormInput
}
```

Response

```
{
  "data": {
    "save_op_template": {"op_template": OpTemplate}
  }
}
```

Mutations

`SCHEDULE_OP_TEMPLATE`

Description

Schedule a pentest from a template.

Response

Returns a `ScheduleOpOutput!`

Arguments

Name	Description
`op_template_name` - `String`	Name of pentest template. If not specified, defaults to the default template for the op_type specified in schedule_op_form.op_type, which in turn defaults to NodeZero.
`op_name` - `String`	Name of the pentest, defaults to the name in pentest template.
`schedule_op_form` - `ScheduleOpFormInput`	Optional parameters that, if set, override the corresponding parameters in the pentest template.
`agent_name` - `String`	Assign an h3-cli agent to automatically launch NodeZero for this pentest
`schedule_name` - `String`	The automated schedule to which this pentest is assigned. This is used internally by automated scheduling.

Example

Query

```
mutation schedule_op_template
  $op_template_name  String,
  $op_name  String,
  $schedule_op_form  ScheduleOpFormInput,
  $agent_name  String,
  $schedule_name  String

  schedule_op_template
                     $op_template_name
            $op_name
                     $schedule_op_form
                $agent_name
                   $schedule_name

    op
      ...OpFragment
```

Variables

```
{
  "op_template_name": "xyz789",
  "op_name": "abc123",
  "schedule_op_form": ScheduleOpFormInput,
  "agent_name": "abc123",
  "schedule_name": "abc123"
}
```

Response

```
{"data": {"schedule_op_template": {"op": Op}}}
```

Mutations

UPDATE_ASSET_GROUP_TEMPLATE

Description

Update an asset group's configuration.

Response

Returns a `SaveOpTemplateOutput!`

Arguments

Name	Description
`asset_group_uuid` - `String!`	ID of asset group.
`schedule_op_form` - `ScheduleOpFormInput!`	Scope for discovering assets.

Example

Query

```
mutation update_asset_group_template
  $asset_group_uuid  String!
  $schedule_op_form  ScheduleOpFormInput!

  update_asset_group_template
                     $asset_group_uuid
                     $schedule_op_form

    op_template
      ...OpTemplateFragment
```

Variables

```
{
  "asset_group_uuid": "xyz789",
  "schedule_op_form": ScheduleOpFormInput
}
```

Response

```
{
  "data": {
    "update_asset_group_template": {
      "op_template": OpTemplate
    }
  }
}
```

Mutations

`UPDATE_CLIENT_ACCOUNT`

Description

Update a client account.

Response

Returns a `ClientAccount!`

Arguments

Name	Description
`client_account_input` - `ClientAccountUpdateInput!`	

Example

Query

```
mutation update_client_account $client_account_input  ClientAccountUpdateInput!
  update_client_account                       $client_account_input
    uuid
    parent_uuid
    child_client_accounts
      ...ClientAccountFragment

    company_name
    company_short_name
    company_logo_url
    secondary_company_logo_url
    company_colors
      ...BrandColorFragment

    white_label_reports_enabled
    white_label_reports_cascade
    row_created_at
    session_user_role_id
```

Variables

```
{"client_account_input": ClientAccountUpdateInput}
```

Response

```
{
  "data": {
    "update_client_account": {
      "uuid": "12341234-1234-1234-1234-123412341234",
      "parent_uuid": "12341234-1234-1234-1234-123412341234",
      "child_client_accounts": [ClientAccount],
      "company_name": "Horizon3 AI, Inc",
      "company_short_name": "H3",
      "company_logo_url": "https://example.com/example",
      "secondary_company_logo_url": "https://example.com/example",
      "company_colors": [BrandColor],
      "white_label_reports_enabled": false,
      "white_label_reports_cascade": true,
      "row_created_at": "2021-07-22T05:02:40.294996",
      "session_user_role_id": "USER"
    }
  }
}
```

Mutations

UPDATE_USER_ACCOUNT

Description

Update a user and their role for a given client account.

Response

Returns a `UserAccount!`

Arguments

Name	Description
`input` - `UpdateUserAccountInput!`	Input data for updating user.

Example
Query

```
mutation update_user_account $input  UpdateUserAccountInput!
  update_user_account        $input
    uuid
    email
    name
    user_role_id
    sign_in_type
    last_signed_in_at
```

Variables

```
{"input": UpdateUserAccountInput}
```

Response

```
{
  "data": {
    "update_user_account": {
      "uuid": "12341234-1234-1234-1234-123412341234",
      "email": "john.smith@example.com",
      "name": "John Smith",
      "user_role_id": "USER",
      "sign_in_type": "BASIC",
      "last_signed_in_at": "2021-07-22T05:02:40.294996"
    }
  }
}
```

Types

AWSACCOUNTID

Description

`String` scalar type with AWS Account ID format required (a 12-digit number).

Example

```
AWSAccountId
```

Types

ACCESSLEVEL

Description

Client account access level.

Values

Enum Value	Description
`FREE_TRIAL`	Free trial access with certain limitations.
`READONLY`	Read-only access.
`FULL`	Full access.
`POV`	Proof-of-value access
`C_PLUS`	Consulting+ access.
`MSP`	Managed Service Provider

Example

```
"FREE_TRIAL"
```

Types

ACTIONLOG

Description

Action log data that represents an entry in the pentest audit log. Each log entry represents an action taken against a host during a pentest.

Fields

Field Name	Description
`uuid` - `String!`	ID of action log.
`start_time` - `Datetime!`	Timestamp at the start of action, in ISO format (UTC).
`endpoint_uuid` - `String`	ID of host targeted by action.
`endpoint_ip` - `String`	IP address of host targeted by action.
`end_time` - `Datetime!`	Timestamp at the end of action, in ISO format (UTC).
`cmd` - `String!`	Attack command run in action.
`module_id` - `String`	Name of attack module used in action.
`module_name` - `String`	Title name of attack module used in action.
`module_description` - `String`	Description of attack module used in action.
`module_meta` - `ModuleMeta`	Metadata for the attack module used in action.
`target_h3_names` - `[String]`	List of assets and weaknesses associated with action. Assets include the IP, service, application, URL, or data store being targeted.
`exit_code` - `String!`	Exit code from attack command run in action.
`op_id` - `String!`	ID of pentest associated with action.
`op_snapshot_id` - `String!`	ID of pentest snapshot associated with action.

Example

```
{
  "uuid": "12341234-1234-1234-1234-123412341234",
  "start_time": Datetime,
  "endpoint_uuid": "12341234-1234-1234-1234-123412341234",
  "endpoint_ip": "123.45.67.89",
  "end_time": Datetime,
  "cmd": "abc123",
  "module_id": "xyz789",
  "module_name": "abc123",
  "module_description": "abc123",
  "module_meta": ModuleMeta,
  "target_h3_names": ["abc123"],
  "exit_code": "abc123",
  "op_id": "12341234-1234-1234-1234-123412341234",
  "op_snapshot_id": "xyz789"
}
```

Types

ACTIONLOGSPAGE

Description

Paginated data of action logs.

Fields

Field Name	Description
`page_info` - `PageInfo`	Pagination of response.
`action_logs` - `[ActionLog!]!`	List of action logs.

Example

```
{
  "page_info": PageInfo,
  "action_logs": [ActionLog]
}
```

Types

AGENT

Description

Represents an Agent aka NodeZero Runner.

Fields

Field Name	Description
`uuid` - `String!`	Unique identifier
`name` - `String!`	The name of the agent. The name is used when assigning scheduled ops to an agent.
`uname` - `String`	The output of `uname` on the agent machine
`log_file` - `String`	The file the agent is logging to on the local machine
`last_heartbeat_at` - `Datetime!`	The last time the agent made contact with H3 (via GQL)
`last_heartbeat_time_ago` - `String!`	`last_heartbeat_at` as a human-friendly "time ago" statement, eg. "3 minutes ago"
`last_command` - `AgentCommand`	The last (or next) command for the agent.
`commands` - `[AgentCommand]`	The last n command for the agent.
Arguments `last_n` - `Int`	
`created_at` - `Datetime!`	When the agent was first created

Example


```
{
  "uuid": "12341234-1234-1234-1234-123412341234",
  "name": "abc123",
  "uname": "xyz789",
  "log_file": "abc123",
  "last_heartbeat_at": "2021-07-22T05:02:40.294996",
  "last_heartbeat_time_ago": "abc123",
  "last_command": AgentCommand,
  "commands": [AgentCommand],
  "created_at": "2021-07-22T05:02:40.294996"
}
```

Types

AGENTCOMMAND

Description

Represents a command run by an Agent aka NodeZero Runner (see Agent).

Fields

Field Name	Description
`uuid` - `String!`	Unique identifier
`agent_uuid` - `String!`	The agent to run the command.
`command` - `String!`	The command to run
`received_at` - `Datetime`	When the agent received the command.
`completed_at` - `Datetime`	When the agent completed the command.
`exit_status` - `String`	The exit status/return code from the command process.
`log` - `String`	The stdout+stderr log from the command process.
`created_at` - `Datetime!`	When the command was created

Example


```
{
  "uuid": "12341234-1234-1234-1234-123412341234",
  "agent_uuid": "12341234-1234-1234-1234-123412341234",
  "command": "abc123",
  "received_at": "2021-07-22T05:02:40.294996",
  "completed_at": "2021-07-22T05:02:40.294996",
  "exit_status": "xyz789",
  "log": "xyz789",
  "created_at": "2021-07-22T05:02:40.294996"
}
```

Types

ASSETGROUP

Description

An AssetGroup represents a set of assets in a pentest environment. Assets are discovered by scanning the environment using the scope defined in the associated op template.

Fields

Field Name	Description
`uuid` - `String!`	ID of asset group
`name` - `String!`	Name of asset group.
`op_template_uuid` - `String!`	ID of pentest template for asset group.
`op_template` - `OpTemplate!`	Data of pentest template for asset group.
`user_account_uuid` - `String!`	ID of user that created the asset group.
`user_account_name` - `String!`	Name of user account that created the asset group.
`client_account_uuid` - `String!`	ID of client account that created the asset group.
`client_account_company_name` - `String!`	Company name on the client account that created the asset group.
`last_ead_etl_completed_at` - `Datetime`	Timestamp when the last asset discovery completed for this asset group.
`created_at` - `Datetime!`	Timestamp when the asset group was created.
`updated_at` - `Datetime`	Timestamp when the asset group was last updated.
`op_series_uuid` - `String!`	ID of the ExternalAssetDiscovery op series associated with the asset group.
`assets_count` - `Int!`	The number of domain and IP assets in this asset group.
`authorized_assets_count` - `Int!`	The number of authorized domain and IP assets in this asset group.
`external_domain_xops_count` - `Int!`	The number of domains in this asset group.
`authorized_external_domain_xops_count` - `Int!`	The number of authorized domains in this asset group.
`in_scope_host_tab_xops_count` - `Int!`	The number of in-scope IPs in this asset group.
`authorized_host_tab_xops_count` - `Int!`	The number of authorized IPs in this asset group.

Example


```
{
  "uuid": "12341234-1234-1234-1234-123412341234",
  "name": "xyz789",
  "op_template_uuid": "12341234-1234-1234-1234-123412341234",
  "op_template": OpTemplate,
  "user_account_uuid": "12341234-1234-1234-1234-123412341234",
  "user_account_name": "abc123",
  "client_account_uuid": "12341234-1234-1234-1234-123412341234",
  "client_account_company_name": "xyz789",
  "last_ead_etl_completed_at": "2021-07-22T05:02:40.294996",
  "created_at": "2021-07-22T05:02:40.294996",
  "updated_at": "2021-07-22T05:02:40.294996",
  "op_series_uuid": "12341234-1234-1234-1234-123412341234",
  "assets_count": 987,
  "authorized_assets_count": 123,
  "external_domain_xops_count": 987,
  "authorized_external_domain_xops_count": 123,
  "in_scope_host_tab_xops_count": 987,
  "authorized_host_tab_xops_count": 123
}
```

Types

ASSETGROUPOUTPUT

Description

Asset group return type.

Fields

Field Name	Description
`asset_group` - `AssetGroup!`	The asset group.

Example

```
{"asset_group": AssetGroup}
```

Types

ASSETGROUPSPAGE

Fields

Field Name	Description
`page_info` - `PageInfo`	Pagination of response.
`asset_groups` - `[AssetGroup!]!`	List of asset groups.

Example

```
{
  "page_info": PageInfo,
  "asset_groups": [AssetGroup]
}
```

Types

AUTHZROLE

Description

Authorization role of a user within a client account. A user's role may change with the client account.

Values

Enum Value	Description
`NOT_ASSOCIATED`	User has no association with the client account.
`DEMO_FREE_TRIAL`	User has free-trial access in the client account.
`USER`	User has non-admin access in the client account.
`READONLY`	User has read-only access in the client account.
`ORG_ADMIN`	User has organizational admin access in the client account.
`NODEZERO_RUNNER`	This role is used with the h3-cli for configuring an automated runner for NodeZero.
`PHISHER`	This role is used for pentests in which phishing is conducted, enable POSTing of phished credentials.
`H3_SERVICE_ACCOUNT`	User is a service account performing actions with an API key
`H3_ADMIN`	User has Horizon3.ai admin access in the client account.
`DELETED`	User has been deleted from the client account.
`H3ADMIN`	Use `H3_ADMIN` instead.
`ORG_READONLY`	Use `READONLY` instead.
`DEMO_READONLY`	Use `READONLY` instead.

Example

```
"NOT_ASSOCIATED"
```

Types

BLOCKEDPENTESTABLEENTITY

Description

A BlockedPentestableEntity links an external domain or IP asset (the PentestableEntity) with its PentestableRules. The PentestableRules determine whether or not we allow external pentesting of the asset. Assets are blocked if we determine they are owned by providers that do not permit pentesting.

Fields

Field Name	Description
`pentestable_entity` - `PentestableEntity`	The pentestable entity that was blocked.
`pentestable_rules` - `PentestableRules`	Then rules that determined why we blocked the asset from external pentesting.

Example

```
{
  "pentestable_entity": PentestableEntity,
  "pentestable_rules": PentestableRules
}
```

Types

BOOLEAN

Description

The `Boolean` scalar type represents `true` or `false`.

Types

BRANDCOLOR

Fields

Field Name	Description
`type` - `BrandColorType!`	Primary or Secondary.
`color` - `HexColor!`	The color in hex notation.

Example

```
{"type": "primary", "color": "#bb032d"}
```

Types

BRANDCOLORINPUT

Fields

Input Field	Description
`type` - `BrandColorType!`	Primary or Secondary.
`color` - `HexColor!`	The color in hex notation.

Example

```
{"type": "primary", "color": "#bb032d"}
```

Types

BRANDCOLORTYPE

Values

Enum Value	Description
`primary`	Primary color.
`secondary`	Secondary color.

Example

```
"primary"
```

Types

CLIENTACCOUNT

Description

Represents a client account.

Fields

Field Name	Description
`uuid` - `String!`	ID of client account.
`parent_uuid` - `String`	ID of parent client account.
`child_client_accounts` - `[ClientAccount!]!`	List of child client accounts, also known as sub-client accounts.
`company_name` - `String!`	Full company name.
`company_short_name` - `String!`	Short name of company.
`company_logo_url` - `String`	Company logo URL.
`secondary_company_logo_url` - `String`	Secondary company logo URL.
`company_colors` - `[BrandColor!]`	Company color scheme.
`white_label_reports_enabled` - `Boolean`	Flag to enable co-branded reports.
`white_label_reports_cascade` - `Boolean`	Flag to cascade co-branded reports to child accounts, if applicable.
`row_created_at` - `Datetime`	Timestamp when the account was created, in ISO format (UTC).
`session_user_role_id` - `AuthzRole`	Role of the current user in this client account.

Example


```
{
  "uuid": "12341234-1234-1234-1234-123412341234",
  "parent_uuid": "12341234-1234-1234-1234-123412341234",
  "child_client_accounts": [ClientAccount],
  "company_name": "Horizon3 AI, Inc",
  "company_short_name": "H3",
  "company_logo_url": "https://example.com/example",
  "secondary_company_logo_url": "https://example.com/example",
  "company_colors": [BrandColor],
  "white_label_reports_enabled": false,
  "white_label_reports_cascade": false,
  "row_created_at": "2021-07-22T05:02:40.294996",
  "session_user_role_id": "USER"
}
```

Types

CLIENTACCOUNTINPUT

Description

Arguments to create client account.

Fields

Input Field	Description
`CompanyName` - `StringNotEmpty!`	Full company name.
`company_short_name` - `StringNotEmpty`	Short company name.
`access_level` - `AccessLevel!`	Access level of client account.
`parent_uuid` - `String`	ID of client account to assign as parent.

Example

```
{
  "CompanyName": StringNotEmpty,
  "company_short_name": StringNotEmpty,
  "access_level": "FREE_TRIAL",
  "parent_uuid": "12341234-1234-1234-1234-123412341234"
}
```

Types

CLIENTACCOUNTOUTPUT

Description

Client account data wrapper.

Fields

Field Name	Description
`client_account` - `ClientAccount`	The client account.

Example

```
{"client_account": ClientAccount}
```

Types

CLIENTACCOUNTUPDATEINPUT

Description

Arguments to update client account.

Fields

Input Field	Description
`uuid` - `String!`	Client account to update.
`CompanyName` - `StringNotEmpty`	Full company name.
`company_short_name` - `StringNotEmpty`	Short company name.
`company_colors` - `[BrandColorInput!]`	Company color scheme.
`white_label_reports_enabled` - `Boolean`	Flag to enable co-branded reports.
`white_label_reports_cascade` - `Boolean`	Flag to cascade co-branded reports to child accounts, if applicable.

Example

```
{
  "uuid": "12341234-1234-1234-1234-123412341234",
  "CompanyName": StringNotEmpty,
  "company_short_name": StringNotEmpty,
  "company_colors": [BrandColorInput],
  "white_label_reports_enabled": true,
  "white_label_reports_cascade": true
}
```

Types

CLIENTACCOUNTSPAGE

Fields

Field Name	Description
`page_info` - `PageInfo`	Pagination of response.
`client_accounts` - `[ClientAccount!]!`	List of client accounts.

Example

```
{
  "page_info": PageInfo,
  "client_accounts": [ClientAccount]
}
```

Types

CREATEUSERACCOUNTINPUT

Description

Arguments to create user account.

Fields

Input Field	Description
`name` - `String!`	Name of user.
`email` - `EmailAddress!`	Email of user.
`client_account_uuid` - `String`	ID of client account to add the user to. Defaults to the current user client account.
`user_role_id` - `AuthzRole!`	Role of user in the given client account.
`is_sso` - `Boolean`	If set, a BASIC auth username/password account will not be created for the user.

Example

```
{
  "name": "John Smith",
  "email": "john.smith@example.com",
  "client_account_uuid": "12341234-1234-1234-1234-123412341234",
  "user_role_id": "USER",
  "is_sso": true
}
```

Types

DATE

Description

`String` scalar type with date ISO format serialization, eg. 2021-07-22.

Example

```
"2023-11-01T18:52:27.602Z"
```

Types

DATETIME

Description

`String` scalar type with datetime ISO format serialization, eg. 2021-07-22T05:02:40.294996.

Example

```
Datetime
```

Types

DELETEOPTEMPLATEOUTPUT

Fields

Field Name	Description
`op_template` - `OpTemplate`	The deleted op template

Example

```
{"op_template": OpTemplate}
```

Types

DELETED

Fields

Field Name	Description
`success` - `Boolean!`	The result of the operation.

Example

```
{"success": true}
```

Types

EMAILADDRESS

Description

`String` scalar type with email address format required.

Example

```
EmailAddress
```

Types

EXCLUDEDDOMAIN

Description

Represents an external domain that was excluded from a pentest due to the given `reason`.

Fields

Field Name	Description
`op_id` - `String!`	The op that excluded the domain.
`domain` - `String!`	The external domain that was excluded.

Example

```
{
  "op_id": "12341234-1234-1234-1234-123412341234",
  "domain": "xyz789"
}
```

Types

EXCLUDEDIP

Description

Represents an IP address that was excluded from a pentest due to the given `reason`.

Fields

Field Name	Description
`op_id` - `String!`	The op that excluded the IP.
`ip` - `String!`	The IP that was excluded.

Example

```
{"op_id": "12341234-1234-1234-1234-123412341234", "ip": "123.45.67.89"}
```

Types

EXTERNALDOMAINXOP

Description

This type is an abstraction over the ExternalDomain type representing a uniquely identified ExternalDomain asset across an OpSeries of ops. The asset is id'ed via the xop_id field. The xop_id field is set to the domain name.

Fields

Field Name	Description
`uuid` - `String!`	{op_series_uuid}/{xop_id}
`op_series_uuid` - `String!`	This ExternalDomainXop type represents an asset being tracked across this OpSeries.
`xop_id` - `String!`	The durable xop identity. Always derived from the data itself (ie. not an arbitrary uuid) For ExternalDomainXop, the xop_id is the domain name.
`last_op_id` - `String`	The last op this domain name appeared in.
`current_op_id` - `String`	The most recently run op in the OpSeries.
`is_authorized` - `Boolean`	Indicates whether or not the domain has been marked "authorized for pentesting" by the user.
`pentestable_rules` - `PentestableRules`	Apply rules to determine whether we allow the user to authorizie this ExternalDomain for pentesting.
`is_dynamic_ip` - `Boolean`	Indicates whether the xop has been marked as "uses dynamic IPs" by the user
`excluded_domain_from_last_pentest` - `ExcludedDomain`	The ExcludedDomain record from the LAST ExternalAttack pentest against this AssetGroup/OpSeries. This will be non-null if the asset was authorized for pentesting, but was excluded (moved out of scope) in the last ExternalAttack pentest, due to drift or unreachability.
`third_party_aliases` - `[String]`	Full list of 3rd-party aliases. Includes all subdomains NOT covered by a TLD in the AssetGroup config. Aliases include CNAMEs (ExternalDomain.cname_chain) and DNS Reverse-Lookup Names (ExternalDomain.endpoint_dns_hostnames).
`third_party_certificate_subject_cns` - `[String]`	3rd-party certificate subject CNs. Includes all CNs NOT covered by a TLD in the AssetGroup config.

Example

```
{
  "uuid": "12341234-1234-1234-1234-123412341234",
  "op_series_uuid": "12341234-1234-1234-1234-123412341234",
  "xop_id": "abc123",
  "last_op_id": "abc123",
  "current_op_id": "xyz789",
  "is_authorized": false,
  "pentestable_rules": PentestableRules,
  "is_dynamic_ip": "123.45.67.89",
  "excluded_domain_from_last_pentest": ExcludedDomain,
  "third_party_aliases": ["xyz789"],
  "third_party_certificate_subject_cns": [
    "xyz789"
  ]
}
```

Types

FEATUREFLAG

Description

Represents Advanced Configuration settings.

Fields

Field Name	Description
`name` - `String!`	The name of the advanced configuration option.
`value` - `Boolean!`	Indicates whether the option is enabled or disabled.
`property_name` - `String`	Human-friendly name for this advanced configuration option.
`property_description` - `String`	Description for this advanced configuration option.
`category_name` - `String`	Category of this advanced configuration option.
`category_description` - `String`	Category description.
`risk` - `FeatureFlagRiskType`	The disruption risk associated with this advanced configuration option.
`enables_min_runtime` - `Boolean`	Indicates whether this advanced configuration option should be used in conjunction with ScheduleOpFormInput.minimum_run_time.
`enables_password_spray` - `Boolean`	Indicates whether this advanced configuration option should be used in conjunction with ScheduleOpFormInput.passwords_to_spray.
`is_new` - `Boolean!`	Indicates the FeatureFlag was introduced after the last time the containing OpTemplate was saved. This helps alert the user about new Attack Config options available in their OpTemplate. Once the OpTemplate is re-saved with the new options, the `is_new` field is reset.

Example

```
{
  "name": "abc123",
  "value": true,
  "property_name": "xyz789",
  "property_description": "xyz789",
  "category_name": "xyz789",
  "category_description": "xyz789",
  "risk": "none",
  "enables_min_runtime": false,
  "enables_password_spray": false,
  "is_new": true
}
```

Types

FEATUREFLAGINPUT

Description

Represents Advanced Configuration input.

Fields

Input Field	Description
`name` - `String!`	The name of the advanced configuration option.
`value` - `Boolean!`	Indicates whether the option is enabled or disabled.

Example

```
{"name": "xyz789", "value": true}
```

Types

FEATUREFLAGRISKTYPE

Values

Enum Value	Description
`none`	The feature has no risk of disruption to your environment.
`low`	The feature has low risk of disruption to your environment.
`moderate`	The feature has moderate risk of disruption to your environment.
`high`	The feature has high risk of disruption to your environment.

Example

```
"none"
```

Types

FILTERBY

Description

Filtering of response data. Corresponding type for `FilterByInput`.

Fields

Field Name	Description
`field_name` - `StringNoWhitespace!`	Name of parameter being filtered by.
`values` - `[String]`	Values being filtered for.

Example

```
{
  "field_name": StringNoWhitespace,
  "values": ["abc123"]
}
```

Types

FILTERBYINPUT

Description

Filter by parameter, e.g. fetch results where `field_name` in `values`.

Fields

Input Field	Description
`field_name` - `StringNoWhitespace!`	Name of parameter to filter by.
`values` - `[String]`	Values to filter for.

Example

```
{
  "field_name": StringNoWhitespace,
  "values": ["abc123"]
}
```

Types

FLOAT

Description

The `Float` scalar type represents signed double-precision fractional values as specified by IEEE 754.

Example

```
987.65
```

Types

GITACCOUNT

Description

Corresponding (non-input) type. Note that GitAccounts are also created by the op.

Fields

Field Name	Description
`name` - `String!`	Name of the git account.
`source` - `GitAccountSource!`	The git service associated with this git account.

Example

```
{"name": "abc123", "source": "GitLab"}
```

Types

GITACCOUNTINPUT

Description

git input type

Fields

Input Field	Description
`name` - `StringNoWhitespace!`	Name of the git account.
`source` - `GitAccountSource!`	The git service associated with this git account.

Example

```
{"name": StringNoWhitespace, "source": "GitLab"}
```

Types

GITACCOUNTSOURCE

Values

Enum Value	Description
`GitLab`	The git account is associated with GitLab
`GitHub`	The git account is associated with GitHub
`Bitbucket`	The git account is associated with BitBucket.

Example

```
"GitLab"
```

Types

HEXCOLOR

Description

Hex color string, eg. #def or #bb032b.

Example

```
HexColor
```

Types

HOSTCSV

Description

`String` scalar type representing a HostCSV row with columns:

- FirstSeen: Datetime
- Subnet: String
- SubnetSource: String
- IP: String
- Hostname: String
- DNSHostnames: String
- LDAPHostname: String
- InScope: Boolean
- OS: String
- Hardware: String
- Device: String
- NumWeaknesses: Int
- NumConfirmedWeaknesses: Int
- NumDataResources: Int
- NumCredentials: Int
- NumConfirmedCredentials: Int
- NumServices: Int
- NumWebShares: Int
- RiskScore: Float
- RiskScoreDescription: String
- OpID: String

Example

```
HostCSV
```

Types

HOSTTABXOP

Description

This type is an abstraction over the HostTab type representing a uniquely identified HostTab asset across an OpSeries of ops. The asset is ID'ed via the xop_id field. A HostTab's xop_id is set to its ip.

Fields

Field Name	Description
`uuid` - `String!`	{op_series_uuid}/{xop_id}
`op_series_uuid` - `String!`	This HostTabXop type represents an asset being tracked across this OpSeries.
`xop_id` - `String!`	The durable xop identity. Always derived from the data itself (ie. not an arbitrary uuid). For HostTabXops that represent AssetGroup IPs, the xop_id is the IP.
`ip` - `String!`	The ip of this host.
`last_op_id` - `String`	The last op this IP appeared in.
`current_op_id` - `String`	The most recently run op in the OpSeries.
`is_authorized` - `Boolean`	Indicates whether or not this IP has been marked "authorized for pentest" by the user.
`excluded_ip_from_last_pentest` - `ExcludedIP`	The ExcludedIP record from the LAST ExternalAttack pentest against this AssetGroup/OpSeries. This will be non-null if the asset was authorized for pentesting, but was excluded (moved out of scope) in the last ExternalAttack pentest, due to drift or unreachability.
`pentestable_rules` - `PentestableRules`	Apply rules to determine whether we allow the user to authorize this IP for pentesting.
`third_party_aliases` - `[String]`	Full list of 3rd-party aliases. Includes all subdomains NOT covered by a TLD in the AssetGroup config. Aliases include CNAMEs (ExternalDomain.cname_chain) and DNS Reverse-Lookup Names (ExternalDomain.endpoint_dns_hostnames).
`third_party_certificate_subject_cns` - `[String]`	3rd-party certificate subject CNs. Includes all CNs NOT covered by a TLD in the AssetGroup config.

Example

```
{
  "uuid": "12341234-1234-1234-1234-123412341234",
  "op_series_uuid": "12341234-1234-1234-1234-123412341234",
  "xop_id": "xyz789",
  "ip": "123.45.67.89",
  "last_op_id": "abc123",
  "current_op_id": "abc123",
  "is_authorized": true,
  "excluded_ip_from_last_pentest": ExcludedIP,
  "pentestable_rules": PentestableRules,
  "third_party_aliases": ["xyz789"],
  "third_party_certificate_subject_cns": [
    "abc123"
  ]
}
```

Types

IMPACTTYPE

Description

Impact types.

Values

Enum Value	Description
`AWSUserRoleCompromise`	Once an AWS user or role is compromised, anything that user or role has access to including cloud resources, cloud services, and data should be considered compromised.
`AWSAccountCompromise`	Once an AWS account is fully compromised, all cloud resources, cloud services, and data that exists in that AWS account should be considered fully compromised.
`AzureADUserCompromise`	DEPRECATED.
`MicrosoftEntraUserCompromise`	Once a Microsoft Entra user is compromised, anything that user has access to should be considered compromised. This could include access to the Microsoft Entra tenant, Microsoft 365, and even access to Azure subscriptions.
`MicrosoftEntraAccountCompromise`	Once an Entra (AzureAD) tenant is fully compromised, any application, service, or resource that utilizes the Entra Entra tenant for Identity and Access Management (IAM) should be considered compromised. This includes cloud services such as Microsoft 365 and Azure-hosted resources such as virtual machines or databases.
`BrandCompromise`	Brand compromise covers ways in which an attacker can harm your company's reputation by, for instance, defacing the company's website, hosting malware off the company's domain, or carrying out phishing attacks that appear to originate from the company.
`BusinessEmailCompromise`	Business email compromise allows attackers to send and receive emails under the guise of that user. Attackers commonly leverage email access to conduct business accounting fraud, conduct highly targeted phishing attacks, gain access to sensitive information, and elicit trusting coworkers to perform actions on their behalf.
`CloudCompromise`	DEPRECATED.
`CloudServiceCompromise`	DEPRECATED.
`CriticalInfrastructureCompromise`	Critical infrastructure consists of key devices and applications that provide attackers a privileged position in the network from which they can access a wealth of sensitive data and launch further attacks.
`DomainCompromise`	Once a domain is fully compromised, all hosts, domain user accounts, data, infrastructure and applications tied to that domain should be considered fully compromised. Additionally, applications running on a domain-joined machine or any application that uses Active Directory integration to authenticate users should be considered fully compromised.
`DomainUserCompromise`	Once a domain user is compromised, anything that user account has access to should be considered compromised.
`HostCompromise`	Host compromise can lead to attackers gaining access to sensitive information, maintaining persistence within your network, and obtaining lateral movement within your networks.
`PerimeterBreach`	Perimeter breach can lead to attackers gaining access to your internal network from the public internet.
`RansomwareExposure`	Ransomware exposures can be used by attackers to obtain access to business-critical data stores, encrypt them with a secret key, and demand a ransom payment from your company before releasing the decryption key. Ransomware attacks can cause severe disruption to your business operations, even after the ransom is paid, as data stores must be decrypted and affected services restored.
`SensitiveDataExposure`	Sensitive data exposures can be used by attackers to obtain user credentials, PII (Personally identifiable information), financial account data, and other business-critical information to further exploit or gain profit.

Example

```
"AWSUserRoleCompromise"
```

Types

INT

Description

The `Int` scalar type represents non-fractional signed whole numeric values. Int can represent values between -(2^31) and 2^31 - 1.

Example

```
987
```

Types

LONG

Description

`Int` scalar type alias.

Example

```
Long
```

Types

MITREMAPPING

Description

A MitreMapping consists of a specific combination of a MITRE Tactic, Technique, and optionally a Subtechnique. MitreMappings are associated with attack modules via ModuleMeta.

Fields

Field Name	Description
`mitre_tactic_id` - `String!`	The MITRE Tactic ID
`mitre_tactic` - `MitreTactic`	The MITRE Tactic
`mitre_technique_id` - `String!`	The MITRE Technique ID
`mitre_technique` - `MitreTechnique`	The MITRE Technique
`mitre_subtechnique_id` - `String`	The MITRE Subtechnique ID
`mitre_subtechnique` - `MitreSubtechnique`	The MITRE Subtechnique

Example

```
{
  "mitre_tactic_id": "TA0043",
  "mitre_tactic": MitreTactic,
  "mitre_technique_id": "T1595",
  "mitre_technique": MitreTechnique,
  "mitre_subtechnique_id": "T1595.002",
  "mitre_subtechnique": MitreSubtechnique
}
```

Types

MITRESUBTECHNIQUE

Description

Represents a MITRE Subtechnique

Fields

Field Name	Description
`id` - `String!`	The MITRE Subtechnique ID
`mitre_technique_id` - `String`	The MITRE Technique ID that is the parent of this Subtechnique
`name` - `String`	The MITRE Subtechnique name
`description` - `String`	The MITRE Subtechnique description
`url` - `String`	The documentation URL for the MITRE Subtechnique

Example

```
{
  "id": "T1595.001",
  "mitre_technique_id": "T1595",
  "name": "Scanning IP Blocks",
  "description": "Adversaries may scan victim IP blocks...",
  "url": "https://attack.mitre.org/techniques/T1595/001/"
}
```

Types

MITRETACTIC

Description

Represents a MITRE Tactic

Fields

Field Name	Description
`id` - `String!`	The MITRE Tactic ID
`name` - `String`	The MITRE Tactic name
`description` - `String`	The MITRE Tactic description
`url` - `String`	The documentation URL for the MITRE Tactic

Example

```
{
  "id": "TA0043",
  "name": "Reconnaissance",
  "description": "The adversary is trying to gather information...",
  "url": "https://attack.mitre.org/tactics/TA0043/"
}
```

Types

MITRETECHNIQUE

Description

Represents a MITRE Technique

Fields

Field Name	Description
`id` - `String!`	The MITRE Technique ID
`name` - `String`	The MITRE Technique name
`description` - `String`	The MITRE Technique description
`url` - `String`	The documentation URL for the MITRE Technique

Example

```
{
  "id": "T1595",
  "name": "Active Scanning",
  "description": "Adversaries may execute active reconnaissance scans...",
  "url": "https://attack.mitre.org/techniques/T1595/"
}
```

Types

MODULEMETA

Description

Attack module metadata.

Fields

Field Name	Description
`id` - `String!`	ID of attack module.
`name` - `String`	Name of attack module.
`description` - `String`	Description of attack module.
`mitre_mappings` - `[MitreMapping]`	MITRE Tactics, Techniques, and Subtechniques associated with this attack module

Example

```
{
  "id": "host_discovery",
  "name": "Host Discovery",
  "description": "The Host Discovery module finds assets on the network...",
  "mitre_mappings": [MitreMapping]
}
```

Types

OP

Description

Pentest data.

Fields

Field Name	Description
`op_id` - `String`	ID of pentest.
`op_type` - `OpType`	Type of pentest.
`op_state` - `String!`	State of pentest.
`op_name` - `String!`	Name of pentest.
`scheduled_timestamp_iso` - `String!`	Timestamp of pentest scheduling, in ISO format (UTC).
`scheduled_at` - `Datetime!`	Timestamp of pentest scheduling
`scheduled_at_date` - `Date!`	Date of pentest scheduling
`completed_timestamp_iso` - `String`	Timestamp of pentest completion, in ISO format (UTC).
`launched_timestamp_iso` - `String`	Timestamp of pentest launch, in ISO format (UTC).
`confirmed_credentials_count` - `Int`	Number of confirmed credentials found.
`weaknesses_count` - `Int`	Number of weaknesses found (weakness INSTANCEs, not unique weakness IDs).
`in_scope_hosts_count` - `Int`	Number of hosts found in scope.
`feature_flags` - `[FeatureFlag]`	Advanced configuration settings for this pentest.
`nodezero_script_url` - `String`	URL of the script that downloads and launches NodeZero.
`duration_hms` - `String`	Pentest duration in HH:MM:SS format.
`duration_humanize` - `String`	Pentest duration in "humanized" format .
`op_template_name` - `String`	The OpTemplate used for this pentest.
`impact_paths_count` - `Int`	Number of impact paths in the pentest.
`runner_name` - `String`	The NodeZero Runner the op is assigned to.
`runner` - `Agent`	The NodeZero Runner the op is assigned to.
`schedule_name` - `String`	The schedule the op is assigned to.

Example


```
{
  "op_id": "12341234-1234-1234-1234-123412341234",
  "op_type": "NodeZero",
  "op_state": "running",
  "op_name": "your op name",
  "scheduled_timestamp_iso": "2021-07-22T05:02:40.294996",
  "scheduled_at": "2021-07-22T05:02:40.294996",
  "scheduled_at_date": "2024-05-01T18:52:27.602Z",
  "completed_timestamp_iso": "2021-07-22T05:02:40.294996",
  "launched_timestamp_iso": "2021-07-22T05:02:40.294996",
  "confirmed_credentials_count": 123,
  "weaknesses_count": 987,
  "in_scope_hosts_count": 123,
  "feature_flags": [FeatureFlag],
  "nodezero_script_url": "https://example.com/example",
  "duration_hms": "22:05:21",
  "duration_humanize": "2 hours, 23 minutes",
  "op_template_name": "xyz789",
  "impact_paths_count": 987,
  "runner_name": "abc123",
  "runner": Agent,
```

```
  "schedule_name": "abc123"
}
```

Types

OPINPUT

Description

Pentest input arguments.

Fields

Input Field	Description
`op_id` - `String!`	ID of pentest.

Example

```
{"op_id": "12341234-1234-1234-1234-123412341234"}
```

Types

OPTAB

Description

Pentest data.

Fields

Field Name	Description
`uuid` - `String`	ID of pentest.
`op_id` - `String`	ID of pentest. Same as `uuid`.
`op_state` - `String!`	State of pentest: • scheduled • provisioning • ready • running • complete • post-processing • done • cancelling • canceled • paused • error
`op_name` - `String!`	Name of pentest.
`scheduled_timestamp` - `Float!`	Timestamp of pentest scheduling, in epoch seconds.
`scheduled_at` - `Datetime!`	Timestamp of pentest scheduling
`scheduled_at_date` - `Date!`	Date of pentest scheduling
`scheduled_timestamp_iso` - `String!`	Timestamp of pentest scheduling, in ISO format (UTC).
`create_timestamp` - `Int!`	Timestamp of pentest creation, in epoch seconds.
`create_timestamp_iso` - `String!`	Timestamp of pentest creation, in ISO format (UTC).
`launched_timestamp` - `Int`	Timestamp of pentest launching, in epoch seconds.
`launched_timestamp_iso` - `String`	Timestamp of pentest launching, in ISO format (UTC).
`launched_at` - `Datetime`	Timestamp of pentest launching
`completed_at` - `Datetime`	Timestamp of pentest completion.
`completed_timestamp` - `Float`	Timestamp of pentest completion, in epoch seconds.
`completed_timestamp_iso` - `String`	Timestamp of pentest completion, in ISO format (UTC).
`canceled_at` - `Datetime`	Timestamp of pentest cancellation.
`canceled_timestamp` - `Int`	Timestamp of pentest cancellation, in epoch seconds.
`canceled_timestamp_iso` - `String`	Timestamp of pentest cancellation, in ISO format (UTC).
`duration_hms` - `String`	Pentest duration in HH:MM:SS format .
`duration_humanize` - `String`	Pentest duration in "humanized" format .
`op_type` - `OpType`	Type of pentest.
`weakness_types_count` - `Int`	Number of unique weakness IDs found.
`weaknesses_count` - `Int`	Number of weaknesses found (weakness INSTANCEs, not unique weakness IDs).
`host_tabs_count` - `Int`	Number of hosts found in scope. Same as `in_scope_endpoints_count`.
`domain_controllers_count` - `Int`	Number of domain controllers found in scope.

Field Name	Description
`credentials_count` - `Int`	Number of credentials and potential credentials found.
`proven_credentials_count` - `Int`	Number of credentials and potential credentials found with proof.
`confirmed_credentials_count` - `Int`	Number of credentials with proof. Alias of `proven_credentials_count`.
`unproven_credentials_count` - `Int`	Number of credentials without proof.
`activedir_passwords_count` - `Int`	Number of credentials found in Active Directory Password Audit(s).
`enabled_activedir_passwords_count` - `Int`	Number of enabled credentials found in Active Directory Password Audit(s).
`disabled_activedir_passwords_count` - `Int`	Number of disabled credentials found in Active Directory Password Audit(s).
`feature_flags` - `[FeatureFlag]`	Advanced configuration settings for this pentest.
`impacts_headline_count` - `Int`	Number of impacts in the pentest, which generally refers to the number of assets affected by an impact.
`impact_paths_count` - `Int`	Number of impact paths in the pentest.
`nodezero_script_url` - `String`	URL of the script that downloads and launches NodeZero.
`nodezero_ip` - `String`	The IP address where NodeZero was launched.
`etl_completed_at` - `Datetime`	Timestamp of pentest post-processing completion, in ISO format (UTC).
`start_paused` - `Boolean`	Start paused.
`minimum_run_time` - `Int`	Op minimum run-time, in minutes.
`maximum_run_time` - `Int`	Op maximum run-time, in minutes.
`paused_at` - `Datetime`	Time when the pentest was most recently paused.
`paused_by_user_account_uuid` - `String`	ID of the user that most recently paused the pentest.
`paused_by_user_account` - `UserAccount`	Data of user account that most recently paused the pentest.
`op_template_name` - `String`	The OpTemplate used for this pentest.
`excluded_ips` - `[ExcludedIP]`	IPs that were excluded from the pentest scope. Applies to external pentests only.
`excluded_domains` - `[ExcludedDomain]`	Domains that were excluded from the pentest scope. Applies to external pentests only.
`runner_name` - `String`	The NodeZero Runner the op is assigned to.
`runner` - `Agent`	The NodeZero Runner the op is assigned to.
`run_nodezero_command` - `AgentCommand`	The run-nodezero command sent to the NodeZero Runner. This field can be used to track status.
`schedule_name` - `String`	The schedule the op is assigned to.
`auto_injected_credential_uuids` - `[String]`	The set of credentials to be auto-injected (by a NodeZero Runner) into the op.

Example


```
{
  "uuid": "12341234-1234-1234-1234-123412341234",
  "op_id": "12341234-1234-1234-1234-123412341234",
  "op_state": "running",
  "op_name": "your op name",
  "scheduled_timestamp": "1600793100.0",
  "scheduled_at": "2021-07-22T05:02:40.294996",
  "scheduled_at_date": "2024-05-01T18:52:27.602Z",
```

```
  "scheduled_timestamp_iso": "2021-07-22T05:02:40.294996",
  "create_timestamp": "1600793100.0",
  "create_timestamp_iso": "2021-07-22T05:02:40.294996",
  "launched_timestamp": "1600793100.0",
  "launched_timestamp_iso": "2021-07-22T05:02:40.294996",
  "launched_at": "2021-07-22T05:02:40.294996",
  "completed_at": "2021-07-22T05:02:40.294996",
  "completed_timestamp": "1600793100.0",
  "completed_timestamp_iso": "2021-07-22T05:02:40.294996",
  "canceled_at": "2021-07-22T05:02:40.294996",
  "canceled_timestamp": "1600793100.0",
  "canceled_timestamp_iso": "2021-07-22T05:02:40.294996",
  "duration_hms": "22:35:09",
  "duration_humanize": "2 hours, 23 minutes",
  "op_type": "NodeZero",
  "weakness_types_count": 987,
  "weaknesses_count": 987,
  "host_tabs_count": 123,
  "domain_controllers_count": 123,
  "credentials_count": 987,
  "proven_credentials_count": 987,
  "confirmed_credentials_count": 987,
  "unproven_credentials_count": 123,
  "activedir_passwords_count": 987,
  "enabled_activedir_passwords_count": 987,
  "disabled_activedir_passwords_count": 987,
  "feature_flags": [FeatureFlag],
  "impacts_headline_count": 123,
  "impact_paths_count": 987,
  "nodezero_script_url": "https://example.com/example",
  "nodezero_ip": "123.45.67.89",
  "etl_completed_at": "2021-07-22T05:02:40.294996",
  "start_paused": false,
  "minimum_run_time": 123,
  "maximum_run_time": 123,
  "paused_at": "2021-07-22T05:02:40.294996",
  "paused_by_user_account_uuid": "12341234-1234-1234-1234-123412341234",
  "paused_by_user_account": UserAccount,
  "op_template_name": "xyz789",
  "excluded_ips": [ExcludedIP],
  "excluded_domains": [ExcludedDomain],
  "runner_name": "xyz789",
  "runner": Agent,
  "run_nodezero_command": AgentCommand,
  "schedule_name": "abc123",
  "auto_injected_credential_uuids": [
    "xyz789"
  ]
}
```

Types

OPTABSPAGE

Description

Paginated data of pentests.

Fields

Field Name	Description
`page_info` - `PageInfo`	Pagination of response.
`op_tabs` - `[OpTab!]!`	List of pentests.

Example

```
{
  "page_info": PageInfo,
  "op_tabs": [OpTab]
}
```

Types

OPTEMPLATE

Description

Pentest (aka "Op") template information.

Fields

Field Name	Description
`uuid` - `String!`	ID of pentest template.
`user_account_uuid` - `String!`	ID of user account that created the pentest template.
`client_account_uuid` - `String!`	ID of client account that created pentest template.
`op_template_name` - `String!`	Name of template.
`op_type` - `OpType!`	Type of pentest this template applies to. "NodeZero" for internal pentests; "ExternalAttack" for external pentests.
`schedule_op_form` - `ScheduleOpForm!`	Template data for scheduling pentests.
`row_created_at` - `Datetime!`	Timestamp when the template was created.
`row_updated_at` - `Datetime`	Timestamp when the template was last updated.

Example


```
{
  "uuid": "12341234-1234-1234-1234-123412341234",
  "user_account_uuid": "12341234-1234-1234-1234-123412341234",
  "client_account_uuid": "12341234-1234-1234-1234-123412341234",
  "op_template_name": "abc123",
  "op_type": "NodeZero",
  "schedule_op_form": ScheduleOpForm,
  "row_created_at": "2021-07-22T05:02:40.294996",
  "row_updated_at": "2021-07-22T05:02:40.294996"
}
```

Types

OPTYPE

Description

Operation type.

Values

Enum Value	Description
`NodeZero`	Internal pentest.
`ExternalAssetDiscovery`	External asset discovery.
`ExternalAttack`	External pentest.
`NetworkEnumeration`	Network Enumeration.
`ADPasswordAudit`	AD Password Audit.
`Phishing`	Phishing.
`AWSPentest`	AWS pentest.
`K8sPentest`	K8s pentest.

Example

```
"NodeZero"
```

Types

PAGEINFO

Description

Pagination of response data.

Fields

Field Name	Description
`page_num` - `Int`	Page number returned by query.
`page_size` - `Int`	Maximum number of items in page.
`text_search` - `String`	Text searched across all text columns in the target table.
`filter_by_inputs` - `[FilterBy]`	List of filters used with AND condition.

Example

```
{
  "page_num": 1,
  "page_size": 20,
  "text_search": "xyz789",
  "filter_by_inputs": [FilterBy]
}
```

Types

PAGEINPUT

Description

Pagination inputs.

Fields

Input Field	Description
`page_num` - `Int`	Page number to query.
`page_size` - `Int`	Maximum number of items per page.
`order_by` - `String`	Name of parameter to order by.
`sort_order` - `SortOrder`	Method to order by.
`sort_inputs` - `[SortInput]`	List of parameters to sort by.
`filter_by_inputs` - `[FilterByInput]`	List of filters to use with AND condition.
`text_search` - `String`	Searches across all text columns in the target table.

Example

```
{
  "page_num": 1,
  "page_size": 10,
  "order_by": "name",
  "sort_order": "ASC",
  "sort_inputs": [SortInput],
  "filter_by_inputs": [FilterByInput],
  "text_search": "abc123"
}
```

Types

PENTEST

Description

Pentest data.

Fields

Field Name	Description
`op_id` - `String!`	ID of pentest.
`op_type` - `OpType`	Type of pentest.
`name` - `String!`	Name assigned to the pentest.
`state` - `PortalOpState!`	op state used exclusively in portal
`user_name` - `String!`	The user who scheduled the pentest.
`client_name` - `String!`	The client account that owns the pentest.
`min_scope` - `[String]`	The "minimum" scope is used in combination with "auto-expand scope" to specify the set of IPs and CIDR ranges that positively should be scanned, in addition to whatever other IPs are discovered during the pentest (up to and including the `max_scope`, if defined).
`max_scope` - `[String]`	The "maximum" scope specifies the full range of IPs and CIDR ranges that are allowed to be discovered and scanned during the pentest.
`exclude_scope` - `[String]`	IPs and CIDR ranges that are excluded from the pentest.
`git_accounts` - `[GitAccount]`	List of Git accounts to include in pentest.
`aws_account_ids` - `[AWSAccountId]`	List of AWS accounts to include in pentest.
`feature_flags` - `[FeatureFlag]`	Advanced configuration options.
`scheduled_at` - `Datetime!`	Timestamp of pentest scheduling.
`launched_at` - `Datetime`	Timestamp of pentest launching.
`completed_at` - `Datetime`	Timestamp of pentest completion.
`canceled_at` - `Datetime`	Timestamp of pentest cancellation.
`etl_completed_at` - `Datetime`	Timestamp of pentest ETL completion.
`duration_s` - `Int`	Pentest duration in seconds.
`impacts_count` - `Int`	Number of impacts found. Technically counts up the number of unique impact type + affected asset combinations.
`impact_paths_count` - `Int`	Number of impact paths found.
`weakness_types_count` - `Int`	Number of unique weakness IDs found.
`weaknesses_count` - `Int`	Number of weaknesses found.
`hosts_count` - `Int`	Number of in-scope hosts (IPs) scanned during the pentest
`out_of_scope_hosts_count` - `Int`	Number of out-of-scope hosts (IPs) found during the pentest.
`external_domains_count` - `Int`	Number of domains found.
`services_count` - `Int`	Number of services found.
`credentials_count` - `Int`	Number of credentials found, both discovered and confirmed.
`users_count` - `Int`	Number of users found.
`cred_access_count` - `Int`	Counts the number of unique combinations of credential + accessed asset.

Field Name	Description
`data_stores_count` - `Int`	Number of data stores found, eg FileShares, DatabaseRepos, S3Buckets, GitRepos, etc. Excludes websites.
`websites_count` - `Int`	Number of websites found.
`data_resources_count` - `Long`	Count of all data resources (files, DB records, etc) across all data stores.
`nodezero_script_url` - `String`	URL of the script that downloads and launches NodeZero.
`nodezero_ip` - `String`	IP address of NodeZero.
`runner` - `Agent`	The NodeZero Runner the op is assigned to.

Example


```
{
  "op_id": "12341234-1234-1234-1234-123412341234",
  "op_type": "NodeZero",
  "name": "xyz789",
  "state": "done",
  "user_name": "xyz789",
  "client_name": "xyz789",
  "min_scope": ["xyz789"],
  "max_scope": ["xyz789"],
  "exclude_scope": ["xyz789"],
  "git_accounts": [GitAccount],
  "aws_account_ids": [AWSAccountId],
  "feature_flags": [FeatureFlag],
  "scheduled_at": "2021-07-22T05:02:40.294996",
  "launched_at": "2021-07-22T05:02:40.294996",
  "completed_at": "2021-07-22T05:02:40.294996",
  "canceled_at": "2021-07-22T05:02:40.294996",
  "etl_completed_at": "2021-07-22T05:02:40.294996",
  "duration_s": 123,
  "impacts_count": 987,
  "impact_paths_count": 123,
  "weakness_types_count": 987,
  "weaknesses_count": 123,
  "hosts_count": 987,
  "out_of_scope_hosts_count": 987,
  "external_domains_count": 123,
  "services_count": 123,
  "credentials_count": 123,
  "users_count": 987,
  "cred_access_count": 987,
  "data_stores_count": 987,
  "websites_count": 987,
  "data_resources_count": Long,
  "nodezero_script_url": "https://example.com/example",
  "nodezero_ip": "123.45.67.89",
  "runner": Agent
}
```

Types

PENTESTABLEENTITIESBULKOUTPUT

Description

Output type for several Mutations that authorize assets for external pentesting.

Fields

Field Name	Description
`pentestable_entities_count` - `Int!`	Number of domain and IP assets that were updated by this request.

Example

```
{"pentestable_entities_count": 987}
```

Types

PENTESTABLEENTITIESOUTPUT

Description

Output type for several Mutations that authorize assets for external pentesting.

Fields

Field Name	Description
`pentestable_entities` - `[PentestableEntity]`	List of domain and IP assets, along with their authorization status, that were modified by this request.
`blocked_pentestable_entities` - `[BlockedPentestableEntity]`	List of domain and IP assets that were BLOCKED from authorization. Assets are blocked if we determine they are owned by providers that do not permit pentesting. See `BlockedPentestableEntity.pentestable_rules.authz_warning` for more details.

Example

```
{
  "pentestable_entities": [PentestableEntity],
  "blocked_pentestable_entities": [
    BlockedPentestableEntity
  ]
}
```

Types

PENTESTABLEENTITY

Description

A PentestableEntity links a domain (`ExternalDomainXop`) or IP (`HostTabXop`) to its authorization status for external pentesting.

Fields

Field Name	Description
`uuid` - `String!`	ID of pentestable entity.
`is_authorized` - `Boolean!`	Flag to indicate if entity is pentestable.

Example

```
{"uuid": "12341234-1234-1234-1234-123412341234", "is_authorized": true}
```

Types

PENTESTABLERULES

Description

The result of applying the rules for allowing ExternalDomains/HostTabs to be pentested via ExternalAttack.

Fields

Field Name	Description
`action_tooltip` - `String`	Tooltip text explaining why an asset was BLOCKED.
`authz_warning` - `String`	Tooltip text warning the user about authorizing this asset and extra vetting of asset ownership that may be required.
`authz_warning_label` - `String`	Label for auth_warning
`is_allowed` - `Boolean!`	The final, ultimate result of applying the authz rules.

Example

```
{
  "action_tooltip": "abc123",
  "authz_warning": "abc123",
  "authz_warning_label": "abc123",
  "is_allowed": false
}
```

Types

PENTESTSPAGE

Fields

Field Name	Description
`page_info` - `PageInfo`	Pagination of response.
`pentests` - `[Pentest!]!`	List of asset groups.

Example

```
{
  "page_info": PageInfo,
  "pentests": [Pentest]
}
```

Types

PORTALOPSTATE

Description

Op states used in portal

Values

Enum Value	Description
`done`	The pentest is fully complete and results are available in Portal.
`ended`	The pentest was ended early by the user. Results are available in Portal.
`error`	The pentest suffered an error. The H3 team is working on it!
`installation_needed`	The pentest is waiting for NodeZero to be launched.
`start_paused`	The pentest was configured to start in a paused state.
`user_paused`	The pentest is paused by a user.
`paused`	The pentest is paused, possibly due to connectivity issues with NodeZero.
`pausing`	The pentest is in the process of pausing.
`preparing`	Resources are being provisioned for the pentest.
`preparing_start_paused`	Resources are being provisioned for the pentest. Once provisioning is complete, the pentest will start in a paused state.
`processing`	The pentest results are being processed.
`resuming`	The pentest is resuming from a paused state.
`running`	The pentest is live and running.
`scheduled`	The pentest has been scheduled and will begin provisioning resources shortly.
`unknown`	The pentest is in an unknown state.

Example

```
"done"
```

Types

SAVEOPTEMPLATEOUTPUT

Fields

Field Name	Description
`op_template` - `OpTemplate!`	The saved op template

Example

```
{"op_template": OpTemplate}
```

Types

SCHEDULEOPFORM

Description

Data to schedule pentest.

Fields

Field Name	Description
`op_name` - `String!`	Name of pentest.
`op_type` - `OpType`	Type of pentest. Defaults to NodeZero.
`op_param_blacklist` - `String`	This scope is EXCLUDED from the pentest. Hosts and subnets that fall within this scope will NOT be pentested. Scope is defined as comma-separated values. Accepts CIDR ranges or plain IP addresses.
`op_param_min_scope` - `String`	Minimum scope is used in combination with Intelligent Scope and Auto-Expand Scope. It specifies the scope that will be explicitly pentested, in addition to any other hosts and subnets that are organically discovered by NodeZero during the pentest. Scope is specified as comma-separated values. Accepts CIDR ranges or plain IP addresses.
`op_param_max_scope` - `String`	Maximum scope of pentest. This represents the upper limits of the pentest scope. Only hosts and subnets that fall within the maximum scope will be pentested. If not defined, the pentest will default to Intelligent Scope. Scope is specified as comma-separated values. Accepts CIDR ranges or plain IP addresses.
`feature_flags` - `[FeatureFlag]`	Advanced configuration settings that control the types of attacks conducted during the pentest.
`osint_domains` - `[String]`	List of company domains that will be scanned for OSINT (open-source intelligence).
`osint_company_names` - `[String]`	List of company names that will be used for discovering OSINT (open-source intelligence).
`passwords_to_spray` - `[String]`	A set of passwords to use for password spraying.
`git_accounts` - `[GitAccount]`	List of Git accounts to scan during the pentest.
`aws_account_ids` - `[AWSAccountId]`	List of AWS accounts to scan during the pentest.
`asset_group_uuid` - `String`	ID of asset group containing authorized assets that will be pentested. Applies to external pentests only (op_type=ExternalAttack).
`start_paused` - `Boolean`	Start the pentest in paused state. This option is useful for external pentests when NodeZero's IP address must first be known in order to open up firewalls.
`minimum_run_time` - `Int`	Op minimum run-time, in minutes. This option is useful to give extra time for password spraying and/or man-in-the-middle relay attacks.
`maximum_run_time` - `Int`	Op maximum run-time, in minutes.
`runner_name` - `String`	The NodeZero Runner that will launch the op.
`auto_injected_credential_uuids` - `[String!]`	The set of credentials to be auto-injected (by a NodeZero Runner) into the op.

Example

```
{
  "op_name": "your op name",
  "op_type": "NodeZero",
  "op_param_blacklist": "123.45.67.89, 12.3.4.56",
  "op_param_min_scope": "123.45.67.89, 12.3.4.56",
  "op_param_max_scope": "123.45.67.89, 12.3.4.56",
  "feature_flags": [FeatureFlag],
  "osint_domains": ["example.com"],
```

```
  "osint_company_names": ["Horizon3"],
  "passwords_to_spray": ["passw0rd!"],
  "git_accounts": [GitAccount],
  "aws_account_ids": [AWSAccountId],
  "asset_group_uuid": "12341234-1234-1234-1234-123412341234",
  "start_paused": false,
  "minimum_run_time": 987,
  "maximum_run_time": 987,
  "runner_name": "xyz789",
  "auto_injected_credential_uuids": [
    "xyz789"
  ]
}
```

Types

SCHEDULEOPFORMINPUT

Description

Inputs to schedule pentest.

Fields

Input Field	Description
`op_name` - `String`	Name of pentest.
`op_type` - `OpType`	Type of pentest. Defaults to NodeZero.
`op_param_blacklist` - `String`	This scope is EXCLUDED from the pentest. Hosts and subnets that fall within this scope will NOT be pentested. Scope is defined as comma-separated values. Accepts CIDR ranges or plain IP addresses.
`op_param_min_scope` - `String`	Minimum scope is used in combination with Intelligent Scope and Auto-Expand Scope. It specifies the scope that will be explicitly pentested, in addition to any other hosts and subnets that are organically discovered by NodeZero during the pentest. Scope is specified as comma-separated values. Accepts CIDR ranges or plain IP addresses.
`op_param_max_scope` - `String`	Maximum scope of pentest. This represents the upper limits of the pentest scope. Only hosts and subnets that fall within the maximum scope will be pentested. If not defined, the pentest will default to Intelligent Scope. Scope is specified as comma-separated values. Accepts CIDR ranges or plain IP addresses.
`feature_flags` - `[FeatureFlagInput]`	Advanced configuration settings that control the types of attacks conducted during the pentest.
`osint_domains` - `[String]`	List of company domains that will be scanned for OSINT (open-source intelligence).
`osint_company_names` - `[String]`	List of company names that will be used for discovering OSINT (open-source intelligence).
`passwords_to_spray` - `[String]`	A set of passwords to use for password spraying.
`git_accounts` - `[GitAccountInput]`	List of Git accounts to scan during the pentest.
`aws_account_ids` - `[AWSAccountId]`	List of AWS accounts to scan during the pentest.
`asset_group_uuid` - `String`	ID of asset group containing authorized assets that will be pentested. Applies to external pentests only (op_type=ExternalAttack).
`start_paused` - `Boolean`	Start the pentest in paused state. This option is useful for external pentests when NodeZero's IP address must first be known in order to open up firewalls.
`minimum_run_time` - `Int`	Op minimum run-time, in minutes. This option is useful to give extra time for password spraying and/or man-in-the-middle relay attacks.
`maximum_run_time` - `Int`	Op maximum run-time, in minutes.
`runner_name` - `String`	The NodeZero Runner that will launch the op.
`auto_injected_credential_uuids` - `[String!]`	The set of credentials to be auto-injected (by a NodeZero Runner) into the op.

Example

```
{
  "op_name": "your op name",
  "op_type": "NodeZero",
  "op_param_blacklist": "123.45.67.89, 12.3.4.56",
  "op_param_min_scope": "123.45.67.89, 12.3.4.56",
  "op_param_max_scope": "123.45.67.89, 12.3.4.56",
  "feature_flags": [FeatureFlagInput],
  "osint_domains": ["example.com"],
```

```
  "osint_company_names": ["abc123"],
  "passwords_to_spray": ["xyz789"],
  "git_accounts": [GitAccountInput],
  "aws_account_ids": [AWSAccountId],
  "asset_group_uuid": "12341234-1234-1234-1234-123412341234",
  "start_paused": false,
  "minimum_run_time": 123,
  "maximum_run_time": 123,
  "runner_name": "xyz789",
  "auto_injected_credential_uuids": [
    "abc123"
  ]
}
```

Types

SCHEDULEOPOUTPUT

Description

Scheduled pentest data.

Fields

Field Name	Description
`op` - `Op!`	Data for a pentest.

Example

```
{"op": Op}
```

Types

SEVERITY

Description

Severity levels associated with a finding's risk score.

```
       score <= 0  : INFO
0   <  score <  4.0: LOW
4.0 <= score <  7.0: MEDIUM
7.0 <= score <  9.0: HIGH
9.0 <= score       : CRITICAL
```

Values

Enum Value	Description
`INFO`	score <= 0
`LOW`	0 < score < 4.0
`MEDIUM`	4.0 <= score < 7.0
`HIGH`	7.0 <= score < 9.0
`CRITICAL`	9.0 <= score

Example

```
"INFO"
```

Types

SIGNINTYPE

Values

Enum Value	Description
`BASIC`	Basic authentiation using username and password.
`GOOGLE`	Federated authentication via Google.
`LINKED_IN`	Federated authentication via LinkedIn.
`MICROSOFT`	Federated authentication via Microsoft.
`SSO`	Private SSO.

Example

```
"BASIC"
```

Types

SORTINPUT

Description

Parameter to sort by.

Fields

Input Field	Description
`order_by` - `StringNoWhitespace!`	Name of parameter to order by.
`sort_order` - `SortOrder`	Method to order by.
`nulls_first` - `Boolean`	Flag to control if nulls are ordered first or last.

Example

```
{"order_by": "name", "sort_order": "ASC", "nulls_first": false}
```

Types

SORTORDER

Description

Method to order by.

Values

Enum Value	Description
`ASC`	Ascending order.
`DESC`	Descending order.

Example

```
"ASC"
```

Types

STRING

Description

The `String` scalar type represents textual data, represented as UTF-8 character sequences. The String type is most often used by GraphQL to represent free-form human-readable text.

Example

```
"xyz789"
```

Types

STRINGNOWHITESPACE

Description

`String` scalar type that cannot be an empty string or contain whitespace.

Example

```
StringNoWhitespace
```

Types

STRINGNOTEMPTY

Description

`String` scalar type that cannot be an empty string.

Example

```
StringNotEmpty
```

Types

UPDATEUSERACCOUNTINPUT

Description

Arguments to update user account.

Fields

Input Field	Description
`email` - `EmailAddress!`	Email of user to update.
`client_account_uuid` - `String`	ID of client account to update the user for, when applicable. Defaults to the current user client account.
`user_role_id` - `AuthzRole!`	Role of user in the given client account.
`name` - `String!`	Name of user.

Example

```
{
  "email": "john.smith@example.com",
  "client_account_uuid": "12341234-1234-1234-1234-123412341234",
  "user_role_id": "USER",
  "name": "John Smith"
}
```

Types

USERACCOUNT

Description

User account data.

Fields

Field Name	Description
`uuid` - `String`	ID of user account.
`email` - `EmailAddress`	User email address.
`name` - `String`	User name.
`user_role_id` - `AuthzRole`	Role of the user in the calling user's ClientAccount.
`sign_in_type` - `SignInType`	Indicates the type of sign-in credentials used by this UserAccount
`last_signed_in_at` - `Datetime`	Timestamp of last time this user signed in under the current user's ClientAccount.

Example

```
{
  "uuid": "12341234-1234-1234-1234-123412341234",
  "email": "john.smith@example.com",
  "name": "John Smith",
  "user_role_id": "USER",
  "sign_in_type": "BASIC",
  "last_signed_in_at": "2021-07-22T05:02:40.294996"
}
```

Types

VULNCATEGORY

Description

Vuln categories.

Values

Enum Value	Description
`SECURITY_MISCONFIGURATION`	The vuln is related to a security misconfiguration of an asset.
`VULNERABILITY`	A product/asset vulnerability.
`POLICY`	The vuln is related to ineffective or insufficient security policies.
`SECURITY_CONTROLS`	The vuln is related to ineffective or insufficient security controls or policies.
`CREDENTIALS`	The vuln is related to credential management or policies.

Example

```
"SECURITY_MISCONFIGURATION"
```

Types

WEAKNESS

Description

A Weakness record represents an observed vulnerability on an affected asset; ie. there is one Weakness record per unique vuln_id + affected asset combination.

Fields

Field Name	Description
`uuid` - `String!`	Unique ID for this Weakness instance.
`created_at` - `Datetime!`	When this Weakness was observed.
`vuln_id` - `String!`	The Weakness's vuln ID.
`vuln_aliases` - `[String!]`	Well-known aliases for this vuln.
`vuln_category` - `VulnCategory`	The vuln's category
`vuln_name` - `String`	The vuln's official name.
`vuln_short_name` - `String`	A convenient short name for the vuln.
`vuln_cisa_kev` - `Boolean`	True if the vuln is a CISA Known Exploited Vulnerability
`vuln_known_ransomware_campaign_use` - `Boolean`	True if the vuln is a CISA Known to be Used in Ransomware Campaigns.
`op_id` - `String!`	The op_id for the Op in which this Weakness was observed.
`ip` - `String`	The IP address where this Weakness was observed.
`has_proof` - `Boolean`	Indicates whether or not we captured proof of the Weakness.
`proof_failure_code` - `String`	The reason why we failed to capture proof, if applicable. One of: exploit_failed, not_configured, no_exploit, timeout.
`proof_failure_reason` - `String`	The reason why we failed to capture proof, if applicable.
`score` - `Float`	The risk score for this Weakness (equal to `context_score` if set, otherwise `base_score`).
`severity` - `Severity`	Severity level associated with the `score`.
`base_score` - `Float`	Base risk score associated with the vuln ID.
`base_severity` - `Severity`	Severity level associated with the base risk score.
`context_score` - `Float`	Risk score based on the weakness's total impact on the environment.
`context_severity` - `Severity`	Severity level associated with the context score.
`context_score_description_md` - `String`	Description explaining the context score/downstream impact for this weakness (in markdown format).
`context_score_description` - `String`	Description explaining the context score/downstream impact for this weakness.
`time_to_finding_hms` - `String`	Time-to-finding in hh:mm:ss format
`time_to_finding_s` - `Int`	Time-to-finding in seconds
`affected_asset_text` - `String`	The display name of the asset directly affected by this weakness.
`affected_asset_short_text` - `String`	The short name of the asset directly affected by this weakness.
`downstream_impact_types` - `[ImpactType!]`	List of ImpactTypes found downstream of this weakness, equivalent to related impacts.
`downstream_impact_types_and_counts` - `[String!]`	List of ImpactTypes and associated counts found downstream of this weakness, equivalent to related impacts. Each array element is formatted as `ImpactType,count`, e.g. `["RansomwareExposure,2", "SensitiveDataExposure,2"]`
`impact_paths_count` - `Int!`	The number of Impact paths the weakness is included in. This should sum up to downstream_impact_types_and_counts

Example


```
{
  "uuid": "12341234-1234-1234-1234-123412341234",
  "created_at": "2021-07-22T05:02:40.294996",
  "vuln_id": "abc123",
  "vuln_aliases": ["xyz789"],
  "vuln_category": "SECURITY_MISCONFIGURATION",
  "vuln_name": "abc123",
  "vuln_short_name": "abc123",
  "vuln_cisa_kev": true,
  "vuln_known_ransomware_campaign_use": true,
  "op_id": "12341234-1234-1234-1234-123412341234",
  "ip": "123.45.67.89",
  "has_proof": false,
  "proof_failure_code": "abc123",
  "proof_failure_reason": "xyz789",
  "score": 123.45,
  "severity": "INFO",
  "base_score": 123.45,
  "base_severity": "INFO",
  "context_score": 987.65,
  "context_severity": "INFO",
  "context_score_description_md": "abc123",
  "context_score_description": "xyz789",
  "time_to_finding_hms": "22:05:31",
  "time_to_finding_s": 987,
  "affected_asset_text": "abc123",
  "affected_asset_short_text": "abc123",
  "downstream_impact_types": ["AWSUserRoleCompromise"],
  "downstream_impact_types_and_counts": [
    "abc123"
  ],
  "impact_paths_count": 123
}
```

Types

WEAKNESSCSV

Description

`String` scalar type representing a WeaknessCSV row with columns:

- WeaknessID: String
- FirstSeen: Datetime
- Name: String
- RootCause: String
- Severity: String
- ContextScore: Float
- ContextScoreDescription: String
- Confirmed: Boolean
- Hostname: String
- CNAME: String
- OS: String
- IP: String
- Port: Int
- Protocol: String
- ProtocolPort: String
- Service: String
- ServiceType: String
- Product: String
- OpID: String
- Description: String
- AssetID: String
- AllHostnames: String
- Repo: String
- Vhost: String
- ResourceUri: String
- UserName: String
- UserDomainName: String
- ProvenEntityEid: String

Example

```
WeaknessCSV
```

3.3 Automate Scheduling

3.3.1 Automate Scheduling of Pentests

A common use case is running pentests automatically on a recurring basis, for example once a week or once a month. This enables continuous autonomous pentesting without any user intervention - no need to sign-in to the Portal to create the pentest, no need to copy+paste the NodeZero launch script to deploy it.

The instructions below walk through how to create a pentest schedule in the Horizon3.ai Portal.

Looking for scheduling via CLI instead of Portal?

Refer to our `h3-cli` guide on Scheduling from CLI.

How it works

Creating a pentest schedule involves three steps:

1. Select a pentest template
2. Specify the schedule
3. Verify the pentest schedule is working as expected (optional)

Click the **Create Schedule** button in the Portal to get started.

1. SELECT A PENTEST TEMPLATE

A pentest template specifies the scope and attack configuration for a pentest. Pentest templates can be created within the **Run Pentest** wizard in the Portal. Note that you can create a template in the wizard without having to run the pentest.

NodeZero Runner required for internal pentests

Creating a pentest schedule for internal pentests requires first setting up a NodeZero Runner. A NodeZero Runner handles the automated deployment of NodeZero on your Docker Host within your internal network.

Once a NodeZero Runner has been set up, you can save it to your pentest template, or you can select it when you create the pentest schedule.

2. SPECIFY THE SCHEDULE

Once you have created and selected your pentest template, the next step is to specify when you want the pentest to run. You have two options for specifying the schedule:

1. Create a simple weekly or monthly schedule using the convenient scheduling form in the **Create Schedule** wizard.
2. Specify a CRON expression for greater precision and control over the schedule.

CRON expressions are an industry-standard way for specifying a recurring schedule. CRON expressions were originally developed for use with the cron job-scheduling service on Unix systems, but they are now widely supported by a number of modern job-scheduling services.

See the section on CRON expressions below for examples and more information.

Schedule precision: pentest schedules support a **minimum precision of 1 hour**. For example you can schedule a pentest to run at 5:00pm, but you cannot schedule it to run at 5:30pm. Therefore the `minute` field in your CRON expression (the first field) must always be 0.

Schedule time zone: scheduled times use UTC time zone. This applies to both the scheduling form and CRON expressions.

3. TEST AND VERIFY THE PENTEST SCHEDULE (OPTIONAL)

Now that you have created a pentest schedule, you can verify everything is working as expected by clicking the **Trigger Now** button. This will trigger the schedule to run immediately, rather than waiting for its normally scheduled time.

The **Trigger Now** feature gives you an opportunity to test the entire flow and address any issues right away instead of discovering them later when the schedule is normally triggered.

If you don't want to actually run the pentest right now, you can cancel it from Portal once it gets started.

Enabling and disabling a pentest schedule

You can enable or disable a pentest schedule at any time by toggling its status in the Portal.

Troubleshooting

If a pentest schedule fails to execute at its normally scheduled time, you will receive an email notification alerting you about the error. The error message is also surfaced on the pentest schedule in the Portal.

For further assistance, contact H3 support via the chat icon in the Portal.

CRON expressions

CRON expressions are an industry-standard way for specifying a recurring schedule. A CRON expression is structured as a series of 5 elements:

```
{minute} {hour} {day-of-month} {month} {day-of-week}
```

There are a number of articles that document CRON expressions and outline how to build them.

Below are several examples of CRON expressions for various time intervals - weekly, monthly, quarterly, etc. You can use these examples as a guide for building your own CRON expressions.

You can verify your CRON expression by plugging it into an online utility like cronhub or crontab.guru.

- `0 8 1 * *`: runs on the first of every month, at 8am UTC.
- `0 8 * * THU`: runs every Thursday (weekly) at 8am UTC.
- `0 8 * * THU#2`: runs on the 2nd Thursday (`THU#2`) of every month.
- `0 8 1 */3 *`: runs on the 1st day of every 3rd month (quarterly).

Running a one-off pentest at a specific date and time

It is not possible to specify a CRON expression that runs only once. Therefore it is not possible to create a pentest schedule that runs only once. However you can approximate this behavior by setting the CRON expression to a specific date and time, then disabling the schedule after the pentest completes.

For example, to schedule the pentest to run at 8am UTC on June 1st, use the following CRON expression:

```
0 8 1 JUN *
```

Note this will run the schedule every June 1st, annually. So if you want to run the pentest only once, disable the schedule after the pentest completes.

References:

1. h3-cli on GitHub
2. Horizon3.ai Portal

3.4 Attack Configuration

3.4.1 Attack Configuration

When configuring a pentest with NodeZero, users are given the option to enable or disable a set of Attack Configuration Options. These options are controllable because they effect the performance of the pentest, or because they have the potential to disrupt the target environment.

This reference page lists the attack configuration flags available in NodeZero, along with descriptions of how these flags effect NodeZero's behavior.

Info

When all attack config options are **disabled**, the operation is still a pentest.

The following activities are performed:

- Asset Discovery
- Identifying potential vulnerabilities
- Exploiting most vulnerabilities/misconfigurations (that have been vetted to not have an operational impact on the target)
- Limited credential discovery and credential pivoting

The following actitivies are not performed:

- Windows Active Directory attacks
- Man-in-the-middle attacks
- Hash cracking
- Password Spray
- Azure AD pivoting
- Default Cred checking
- OS credential dumping
- Any brute force enumeration
- Any exploits specifically disabled in the advanced config (but most exploits are still executed as described above)

Brute Force

Properties related to modules that carry out brute force attacks.

Name	Description	Risk
DNS	Enables brute forcing of internal DNS records. Only applies if an operation has been scheduled with the Intelligent Scope option. This may place noticeable load on DNS servers in the network.	low
S3	Enables brute-force discovery of S3 buckets using wordlists and top level company domain names. This can add significant time if the pentest has been configured to run against many top-level domains.	none
Subdomains	Enables brute-force discovery of company subdomains using a large wordlist of common subdomain names. This can aid in the discovery of more external assets but significantly extend the time it takes for discovery to complete.	none

Credential Verification

Properties related to modules that use credentials discovered by NodeZero to access services in the environment.

Name	Description	Risk
Azure AD Credential Pivoting	Enables using domain user credentials discovered in an internal pentest against Azure Active Directory services. Requires user-entered Domains from the OSINT step.	none
Azure AD Password Spray	Enables password spraying Azure cloud users with common passwords by NodeZero. By default, a user will only be tried three times every 60 minutes. There is a small chance of locking out accounts.	moderate
Credential Reuse	Checks for access to services and shares using local user (non-domain) authentication.	none
Domain User	Checks for Windows domain user access by authenticating with credentials against the SMB service running on the Windows Domain Controller.	none
Internal Password Spray	Enables password spraying domain users with common passwords by NodeZero. By default, a user will only be tried twice every 60 minutes	moderate

Data

Properties related to data discovery.

Name	Description	Risk
Domain Admin Scanning of SMB Shares	Enables scanning of SMB shares using domain administrator credentials that were injected into the pentest or discovered during the course of the pentest. Enabling this flag provides a more complete picture of data risk but can add significant time to the pentest.	none
Extended Domain User Scanning of SMB Shares	Enables scanning of all SMB shares accessible to domain users whose credentials were injected into the pentest or discovered during the course of the pentest.	none
Verify Permissions on SMB Shares	Verify read, write, list, and delete permissions on an SMB share by writing a test file and deleting it afterwards. Cleanup of the test file may fail in exceptional circumstances.	none

Default Credentials

Properties related to modules that check for default credentials using a dictionary attack with known default credentials.

Name	Description	Risk
FTP	Enables checking default credentials against FTP services found by NodeZero.	low
Microsoft SQL Server	Enables checking default credentials against Microsoft SQL Server databases found by NodeZero. There is a small chance of locking out the `sa` account.	moderate
MongoDB	Enables checking default credentials against MongoDB databases found by NodeZero.	low
MySQL	Enables checking default credentials against MySQL databases found by NodeZero.	low
PostgreSQL	Enables checking default credentials against PostgreSQL databases found by NodeZero.	low
SNMP	Enables checking default SNMPv1 community strings against SNMP services found by NodeZero.	low
SSH	Enables checking default credentials against SSH services found by NodeZero. Against older ESXi servers vulnerable to CVE-2019-5528, this module may trigger a partial denial of service condition in the `hostd` process.	moderate
Telnet	Enables checking default credentials against telnet services found by NodeZero.	low
Web	Enables checking default credentials against HTTP or HTTPS web servers found by NodeZero.	low

Environment Impact

Properties related to modules that change the environment. All modules attempt to clean up after themselves but there is a small chance cleanup may fail.

Name	Description	Risk
ADCS ESC4 Attack - Misconfigured Templates Access Controls	Exploit vulnerable Active Directory Certificate Templates that allow an unprivileged user to overwrite Certificate Template security features -- enabling Subject Alternative Name (SAN). Restoration of original template configuration may fail in exceptional cases.	none
Anonymous Docker Engine Write Check	Checks for write privileges against a Docker Engine instance that allows anonymous (unauthenticated) access. The check attempts to create a Docker container or pull a Docker image and deletes the container or image afterwards.	none
Anonymous Printer Access	Check for anonymous access to printers over port 9100. This check may cause certain printer models to print out pages.	moderate
Anonymous ZooKeeper Write Check	Checks for write privileges against a ZooKeeper instance that allows anonymous (unauthenticated) access. The check writes to a ZooKeeper node and deletes it afterwards.	none
Atlassian Crowd and Crowd Data Center Remote Code Execution (CVE-2019-11580)	Checks for exploitability of CVE-2019-11580 by uploading a malicious JAR file. This upload is likely to be caught by AV software on the host. Cleanup of the JAR file may fail in exceptional cases.	none
CVE-2022-26923 (Certifried) Privilege Escalation - Creation of Machine Account	Attempt to exploit vulnerable Active Directory Certificate Services Privilege Escalation by creating a machine account and manipulating its attributes. Deletion of the machine account may fail in exceptional circumstances.	none
Elasticsearch Write Check	Checks for write privileges against an Elasticsearch cluster. The check attempts to create an index and deletes it afterwards.	none
FTP Write Check	Checks for write privileges against an FTP server. The check creates a remote directory and deletes it afterwards.	none
Insecure JMX (H3-2020-0022)	Tests exploitability of the insecure JMX weakness (H3-2020-0022). The test checks for remote code execution by installing a payload on the vulnerable JMX service, runs a small set of commands using the payload, and uninstalls the payload at the end. There is a small chance that cleanup of the payload may fail.	none
ManageEngine ServiceDesk Plus PreAuth RCE (CVE-2021-44077)	Checks for exploitability of CVE-2021-44077 by uploading a malicious payload through that API, and execute the payload through another API. This upload is likely to be caught by AV software on the host. If successful, this exploit will leave behind a file msiexec.exe in the ManageEngine\ServiceDesk\site24x7 folder.	none
Subdomain Takeover	Proactively takeover and hold onto subdomains that are vulnerable to subdomain takeover (H3-2021-0002) to prevent bad actors from compromising them first.	none
VMWare vCenter Server Access Control Vulnerability (CVE-2020-3952)	Checks for exploitability of CVE-2020-3952 by adding an administrative user and removing it afterwards.	none
VMWare vCenter Server Plugin Remote Code Execution Vulnerability (CVE-2021-21972)	Checks for exploitability of CVE-2021-21972 by installing a webshell, executing a command within the webshell, and removing it afterwards. For vCenter servers running on Linux, it is possible that randomly-named webshells will be left behind on the vulnerable vCenter server if the exploit fails.	none
VMWare vRealize Operations Manager SSRF	Checks for exploitability of CVE-2021-21975 and CVE-2021-21983 by installing a randomly named webshell, executing a command within	none

Name	Description	Risk
Vulnerability (CVE-2021-21975)	the webshell, and removing it afterwards. Cleanup of the webshell may fail in exceptional cases.	
Zoho ManageEngine ADSelfService Plus API Auth Bypass (CVE-2021-40539)	Checks for exploitability of CVE-2021-40539 by uploading a malicious JAR file. This upload is likely to be caught by AV software on the host. Cleanup of the JAR file may fail in exceptional cases.	none

Exploitation

Attempt exploitation of a vulnerability to confirm that it can be exploited by NodeZero.

Name	Description	Risk
Bluekeep (CVE-2019-0708)	Tests exploitability of the Bluekeep vulnerability (CVE-2019-0708). There is a moderate-level risk this exploit may crash the target host, and it is not recommended for use against production systems.	high
Cisco Smart Install Vulnerability (CVE-2018-0171)	Tests exploitability of the Cisco Smart Install vulnerability (CVE-2018-0171). The test attempts to pull router config from the vulnerable router via the TFTP protocol. Against a few older models of Cisco routers, running this exploit may cause the router to reload or go down.	moderate
EternalBlue (MS17-010)	Tests exploitability of the Windows SMB remote code execution vulnerability EternalBlue. This is a kernel buffer overflow exploit and carries a moderate risk of crashing the target. It is not recommended for use against production systems. This exploit is only attempted if NodeZero is able to reliably determine the target operating system and NodeZero is not able to first exploit EternalChampion/EternalSynergy/ EternalRomance.	moderate
EternalChampion/ EternalSynergy/ EternalRomance (MS17-010)	Tests exploitability of the Windows SMB remote code execution vulnerabilities EternalChampion, EternalSynergy, and EternalRomance.	low
Exploding Can (CVE-2017-7269)	Tests exploitability of the IIS 6.0 WebDAV vulnerability CVE-2017-7269, aka Exploding Can.	low
HP iLO Web API Remote Code Execution (CVE-2017-12542)	Tests exploitability of the HP iLO Web API Remote Code Execution vulnerability (CVE-2017-12542). The test attempts to retrieve users and their credentials by exploiting a heap-based buffer overflow.	low
Heartbleed (CVE-2014-0160)	Tests exploitability of the Heartbleed vulnerability (CVE-2014-0160), if discovered by NodeZero. This test dumps memory from the vulnerable server.	low
Server Service Vulnerability (MS08-067)	Tests exploitability of the Windows SMB remote code execution vulnerability CVE-2008-4250, aka MS08-067. There is a high likelihood that this exploit will crash the SMB service on the target after successful exploitation.	high

Hash Cracking

Properties related to cracking hashes found in the environment.

Name	Description	Risk
Automatic Hash Cracking	Automatically attempt to crack hashes found in the environment.	none

Man in the Middle Attacks

Properties related to modules that conduct man-in-the-middle (MITM) attacks.

Name	Description	Risk
Expanded LLMNR and NetBIOS poisoning	Enables sniffing of cleartext passwords and hashes sent over insecure protocols such as LLMNR, NetBIOS, SMB v1, and HTTP. This will sniff all available traffic regardless of scope.	none
Limited LLMNR and NetBIOS poisoning	Enables sniffing of cleartext passwords and hashes sent over insecure protocols such as LLMNR, NetBIOS, SMB v1, and HTTP. This is limited to the scope provided during the configuration of the pentest. If selected, this option overrides the 'Expanded LLMNR and NetBIOS poisoning' option.	none
Net-NTLM Authentication Coercion	Enables Net-NTLM Authentication coercion techniques. This allows attackers to capture Net-NTLM (NTLMv2) hashes by coercing machines to authenticate to an attacker controller server.	none
Net-NTLM Hash Relaying	Enables SMB relay attacks. This allows attackers to gain unauthorized access to machines by capturing Net-NTLM (NTLMv2) hashes over the network and relaying them to target SMB servers.	none

Post-Exploitation

Properties related to actions taken after compromising a host

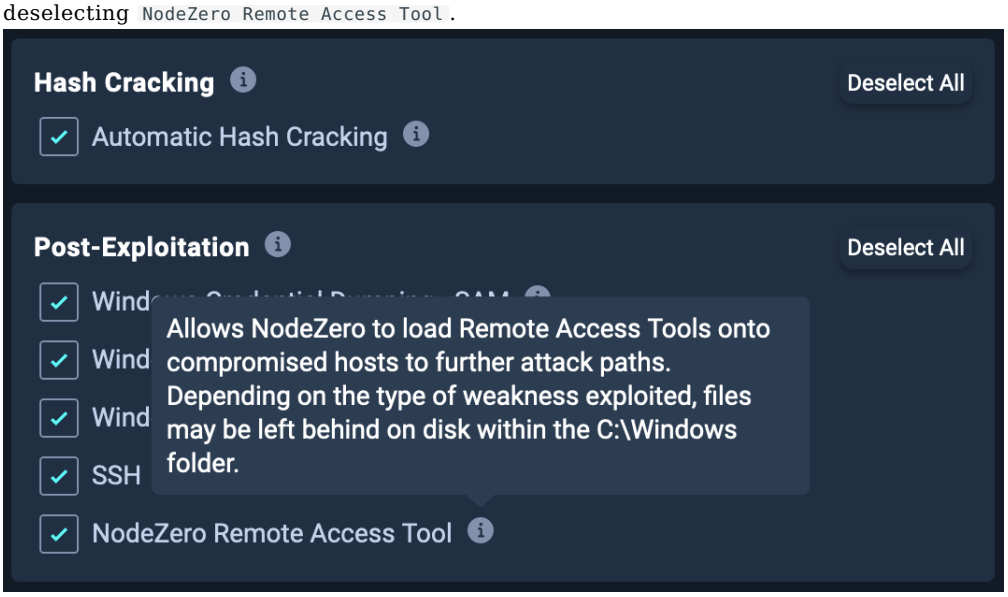
Name	Description	Risk
SSH	Enables post-exploit actions such as system enumeration and privilege escalation on hosts for which SSH access was gained. In exceptional circumstances, files may be left on disk in the /tmp folder.	none
Windows Credential Dumping - LSA Secrets	Enables dumping of credentials from the Local Security Authority (LSA) after gaining administrative access to a Windows machine. In exceptional circumstances, cleanup may fail, leaving files on disk.	none
Windows Credential Dumping - LSASS	Enables dumping of credentials stored in the Local Security Authority Subsystem Service (LSASS) process, after gaining administrative access to a Windows machine. In exceptional circumstances, cleanup may fail, leaving files on disk.	low
Windows Credential Dumping - SAM	Enables dumping of credentials from the Security Account Manager (SAM) database after gaining administrative access. In exceptional circumstances, cleanup may fail, leaving files on disk.	none

3.4.2 Remote Access Tool

Paid Feature

Limited Access Feature Advanced post-exploitation and lateral movement are only available to paid clients

NodeZero can leverage detected weaknesses and vulnerabilities to deploy Remote Access Tools (RATs). These tools provide additional access that NodeZero uses to further explore attack paths during operations. This functionality is enabled by default but can be disabled under the `Post-Exploitation` section in the advanced configuration settings by deselecting `NodeZero Remote Access Tool`.



NodeZero coordinates these RATs to conduct more extensive post-exploitation on hosts, including common MITRE ATT&CK® techniques such as system information gathering and credential dumping.

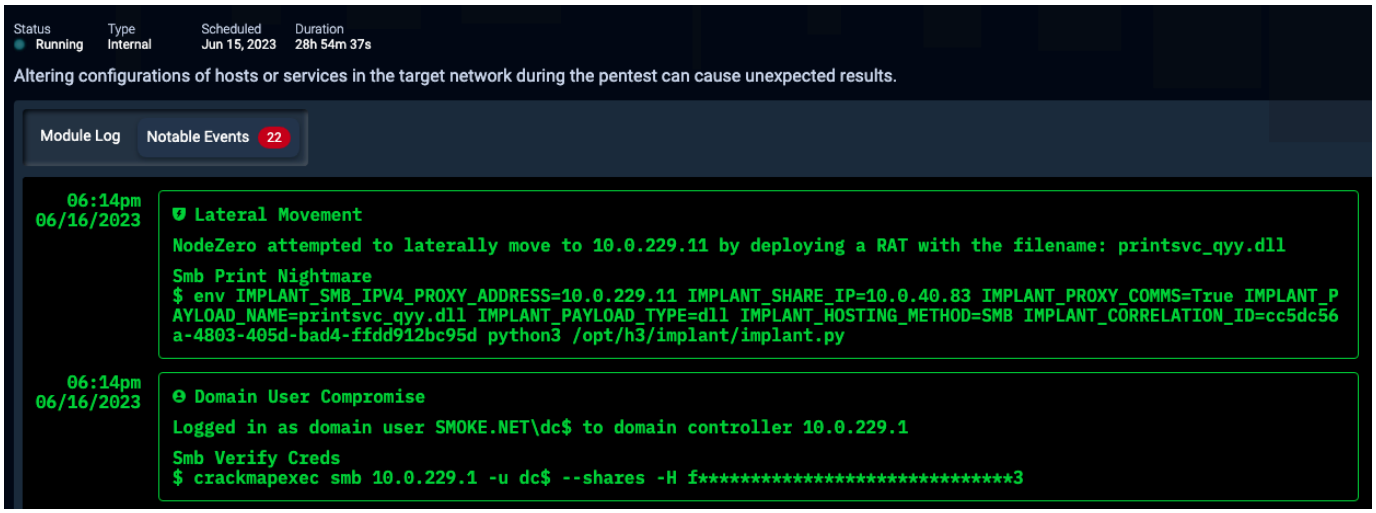
What is a Remote Access Tool?

Remote Access Tools are applications that attackers use to remotely administer an endpoint. These tools fall into different categories, including legitimate administrative tools like AnyDesk, as well as specially designed tools meant to evade antivirus and EDR products, such as Cobalt Strike, Meterpreter, and Sliver.

NodeZero utilizes a specially developed internal tool that offers many of the same capabilities as the aforementioned adversary emulation tools.

How will I know if NodeZero deployed a RAT during an operation?

During an operation, NodeZero will generate an event in the "Real-Time View" that includes the host to which it attempted to deploy the RAT, along with any associated artifacts that may be present, such as a filename.



After an operation, the host's "Activity Logs" will contain the same type of information.

Are there risks associated with enabling this feature?

No, the tooling that NodeZero employs is a User Mode RAT, which means it will not introduce system instability when used. Any post-exploitation activities conducted, such as credential dumping, use safe tactics to avoid causing any system disruptions.

Does the RAT leave any artifacts after an operation?

The RAT makes significant efforts to clean up itself and any artifacts during and after an operation. When the operation ends, all deployed RATs are uninstalled from all systems. However, in rare circumstances, some artifacts may be left on disk. If any artifacts are left, they can most commonly be found in:

- `C:\Windows\Temp\` - Some weaknesses exploitation and credential dumping activities may momentarily create a file with a random name in this directory.
- `C:\Windows\System32\spool\drivers\x64\` - A file may be created here when exploiting common print service issues.

3.5 BloodHound

BloodHound is an open source toolset to collect and analyze relational data within an Active Directory and/or Azure environment. BloodHound uses graph theory to reveal hidden and often unintended relationships within these environments to allow Attackers and Defenders to identify highly complex attack paths that would otherwise be impossible to quickly identify. BloodHound has become an industry standard tool for both Red Teams and Blue Teams to attack and defend Active Directory environments.

For additional details on how to use the BloodHound tool see BloodHound's Official Documentation.

> **Compatibility Note**
>
> On August 8, 2023 SpectreOps released the latest version of BloodHound as BloodHound: Community Edition. **Currently, NodeZero only supports BloodHound v4.2-v4.3.1**.

3.5.1 Installing and Setting up BloodHound

Please see BloodHound's Installation documentation:

- Windows
- Mac
- Linux

Using Neo4j in a Docker Container

If you wish to run the Neo4j DBMS for BloodHound in a Docker container, there is an available image on dockerhub.

1. Pull the docker image:

```
docker pull neo4j/neo4j:4.4.13
```

2. Create a local `data` directory to which neo4j has write permissions.
3. Create and run the docker container:

- Volume mount the created `data` directory to `/data` within the container
- Publish container ports 7687 (the bolt protocol port) and optionally 7474 (the neo4j browser interface).
- Setting the `NEO4J_AUTH` environment variable to a username and password combination

```
docker run --name neo --rm -v $PWD/data/:/data -e NEO4J_AUTH=neo4j/password -p 7687:7687 -p 7474:7474 neo4j:4.4.13
```

Reference: Docker run Command

Once the container is running, you should be able to connect via the BloodHound GUI.

3.5.2 How Does NodeZero Use BloodHound?

After NodeZero discovers and verifies a domain user credential, it will utilize a BloodHound data collector to gather information on the Active Directory or Azure environment. NodeZero stores this data in a neo4j 4.4.x graph database in our ephemeral architecture during the life of the pentest, and will utilize it to identify complex attack paths that may lead to compromising the domain. After the pentest finishes, the BloodHound data is backed up and stored for a limited time. H3 customers who wish to utilize NodeZero's BloodHound collections to inform their own Red/Blue/Purple team operations can request the data from a pentest for a **limited time**.

3.5.3 Obtaining NodeZero's BloodHound Data

> **Note**
>
> The ability to download NodeZero's collected BloodHound data is a paid feature, unavailable to free trials. If you would like to request access to this feature, please contact H3 Customer Success.

3.5.4 Using a NodeZero Pentest's BloodHound Data

NodeZero provides BloodHound data in the form of a neo4j backup dump file. Users can use this file to directly load the data into the neo4j database they connect the BloodHound GUI to.

Using `neo4j-admin` to Import NodeZero's BloodHound Dump to Neo4j

Neo4j provides an administrative command-line tool called `neo4j-admin` to manage/administer its Database Management System (DBMS). This tool is typically located in the neo4j `bin` directory. `neo4j-admin`'s `load` command loads the archive file that NodeZero produces. The command can be run from an online or an offline neo4j DBMS. Typically, the `neo4j-admin load` command should be run as the `neo4j` user to ensure appropriate file permissions.

```
$neo4j-admin load --database=neo4j --from=<DUMP_FILE_PATH>
```

Reference: neo4j 4.4 - Restore a database dump

Importing NodeZero's BloodHound Dump When Running Neo4j as a Docker Container

The neo4j docker images from dockerhub do not contain the `neo4j-admin` utility. Instead, you will need to pull the image for `neo4j-admin` itself:

```
docker pull neo4j/neo4j-admin:4.4.13
```

Once complete, users can use the following command to extract the BloodHound data dump from NodeZero into the data folder they will mount when running the `neo4j` container:

```
docker run --interactive --tty --rm  --volume=$PWD/data:/data --volume=<ABSOLUTE_PATH_TO_BLOODHOUND_DATA_DUMP.dump>:/backups.dump neo4j/neo4j-
admin:4.4.13 neo4j-admin load --database=neo4j --from=/backups.dump
```

Once the dump is finished extracting, you can run your neo4j container:

```
docker run --name neo --rm -v $PWD/data/:/data -e NEO4J_AUTH=neo4j/password -p 7687:7687 -p 7474:7474 neo4j:4.4.13
```

3.6 Glossary

Welcome to our product glossary, a quick reference for essential cybersecurity terms unique to our solution. Whether you're new to cybersecurity or a seasoned professional, this resource will help you understand the specific language used in our product. Organized alphabetically, if you find something is missing, please feel free to provide feedback at the bottom of the page!

3.6.1 Attack Path

An attack path refers to the sequence of steps or actions an attacker may take to compromise a system or network. It involves identifying vulnerabilities and other weaknesses, exploiting them, and navigating through the network to access valuable information or resources.

3.6.2 BloodHound Data

BloodHound data is the reconnaissance information collected and analyzed by the BloodHound tool within an Active Directory and/or Azure environment. NodeZero users have the option to obtain the BloodHound data collected during a pentest.

3.6.3 Impact

Impacts summarize, in business terms, the effects NodeZero was able to achieve as a result of exploiting weaknesses in your environment.

3.6.4 N-Day

An N-day is a software or hardware vulnerability that is already publicly known, (n days since disclosure) but there may or may not be a security update available to remediate the vulnerability.

3.6.5 NodeZero Runner

The NodeZero runner enables the automated deployment of a NodeZero Docker container. This allows you to provision and deploy pentests from the portal, without having to manually run a NodeZero launch script.

3.6.6 Notable Event

A feature of Real-Time View (RTV). These events signify that during the pentest, NodeZero performed actions that would likely lead to a critical Impact.

3.6.7 RAT

RAT stands for remote access tool, software that gives a person full control of a tech device remotely. They have legitimate uses, such as technical support, but can also be controlled by attackers with malicious intent. In the context of NodeZero, a RAT is used to provide NodeZero with additional access to further explore attack paths during operations.

3.6.8 Real-Time View (RTV)

Real-Time View in NodeZero provides you with real-time information and updates on the progress of your running pentest, including status updates for injected credentials.

3.6.9 Sensitive Data Exposure

An Impact that indicates NodeZero was able to potentially access sensitive information given the filetype or service that is compromised. Examples include, but are not limited to:

- Business documents in file shares (.docx, .pdf, .xlsx)
- Outlook PST files
- Confluence RCE
- Exchange RCE

3.6.10 States (Pentest)

The pentest lifecycle encompasses several stages, each with its unique purpose and characteristics. These stages surface in portal with words like: 'Preparing', 'Action Needed', 'Running', 'Processing', and 'Done'. Below, you will find a detailed breakdown of these stages and their respective descriptions.

State Name	Description
Action Needed	Copy and run the one-time command on your Docker Host to launch NodeZero.
Action Needed: Paused	Start your pentest after adding NodeZero's IP to your allowlist.
Done	The pentest is fully complete and results are available in the portal.
Ended	The pentest was ended early by the user. Results are available in the portal.
Error	An error was encountered during this pentest.
Paused	The pentest is paused.
Pausing	The pentest is in the process of pausing.
Preparing	The pentest is setting up the resources it needs.
Preparing (Start Paused)	Resources are being provisioned for the pentest. Once provisioning is complete, the pentest will start in a paused state.
Processing	The pentest results are being processed.
Resuming	The pentest is resuming from a paused state.
Running	The pentest is live and running.
Scheduled	The pentest has been scheduled and will begin provisioning resources shortly.
Unknown	The pentest encountered an issue. Contact us for further assistance if this issue persists.

3.6.11 Weakness

A weakness refers to a vulnerability or security flaw that can be exploited by an attacker to compromise a system or network. Weaknesses can include misconfigurations, outdated software, default credentials, or other vulnerabilities that can be leveraged to gain unauthorized access or perform malicious actions.

3.7 Horizon 3 Weaknesses

3.7.1 Weaknesses

NodeZero's identifies and surfaces many weaknesses that it finds during a pentest. These weaknesses are identified by a Common Vulnerabilities and Exposures (CVE) identifier (e.g. `CVE-2021-44228`), or a Horizon3.ai weakness identifier (e.g. `H3-2022-0001`).

This page provides a reference for Horizon3.ai Weaknesses identified by NodeZero. For information on CVEs identified by NodeZero, please reference the official CVE website maintained by MITRE.

Weakness ID	Name
H3-2020-0002	Anonymous Access to ZooKeeper API
H3-2020-0003	Anonymous Access to Printer using PJL or PS
H3-2020-0004	Zone Transfer Allowed to Any Server
H3-2020-0005	Anonymous FTP Enabled
H3-2020-0007	SMB Null Session Allowed
H3-2020-0008	Guest Account Enabled
H3-2020-0009	Weak NFS Export Permissions
H3-2020-0010	NFS UID/GID Manipulation Possible
H3-2020-0016	Insecure IPMI Implementation
H3-2020-0017	IPMI Cipher Zero Vulnerability
H3-2020-0021	Unauthenticated Access to the Jenkins Script Console
H3-2020-0022	Insecure Java JMX Configuration
H3-2020-0023	Apache Hadoop YARN ResourceManager Unauthenticated Command Execution
H3-2020-0030	Android Debug Bridge (ADB) over TCP Enabled
H3-2021-0001	Public Access to Amazon S3 Bucket
H3-2021-0002	Subdomain Takeover
H3-2021-0003	Unauthenticated Access to Sensitive Kubelet API Endpoints
H3-2021-0004	Kubernetes Privileged Container Exposure
H3-2021-0005	Unauthenticated Kubelet API Remote Code Execution Vulnerability
H3-2021-0006	Unauthenticated Kubernetes API Server Access
H3-2021-0007	Kubernetes Service Account Token Exposure
H3-2021-0008	Unauthenticated Etcd Access
H3-2021-0009	Unauthenticated Docker Registry API Access
H3-2021-0010	Unauthenticated Docker Engine API Access
H3-2021-0011	Kerberos Pre-Authentication Disabled
H3-2021-0012	Weak or Default Credentials - FTP
H3-2021-0013	Weak or Default Credentials - Telnet
H3-2021-0014	Weak or Default Credentials - SSH
H3-2021-0015	Weak or Default Credentials - SNMP
H3-2021-0016	Weak or Default Credentials - Microsoft SQL Server
H3-2021-0017	Weak or Default Credentials - MySQL
H3-2021-0018	Weak or Default Credentials - Postgres
H3-2021-0019	Weak or Default Credentials - Password Spray
H3-2021-0020	Weak or Default Credentials - Cracked Credentials
H3-2021-0021	Weak or Default Credentials - Web Applications

Weakness ID	Name
H3-2021-0024	Dangling DNS Record
H3-2021-0029	AWS Unrestricted Assume Role Access
H3-2021-0030	SMB Signing Not Required
H3-2021-0031	Public Access to Git Repository
H3-2021-0032	Credential Reuse
H3-2021-0033	mDNS Poisoning Possible
H3-2021-0034	LLMNR Poisoning Possible
H3-2021-0035	NBT-NS Poisoning Possible
H3-2021-0036	Unauthenticated Access to Elasticsearch
H3-2021-0037	Werkzeug Debug Console Enabled
H3-2021-0038	Kerberoasting
H3-2021-0039	Unrestricted Sudo Privileges
H3-2021-0040	AWS Instance Metadata Service v1 Exposed
H3-2021-0041	Apache Druid Server-Side Request Forgery Vulnerability
H3-2021-0042	Credential Dumping - Security Account Manager (SAM) Database
H3-2021-0043	Credential Dumping - Local Security Authority (LSA) Secrets
H3-2021-0044	Credential Dumping - Local Security Authority Subsystem Service (LSASS) Memory
H3-2021-0045	Credential Dumping - /etc/shadow File
H3-2021-0046	Credential Dumping - Active Directory Services Database (NTDS)
H3-2021-0047	JBoss Application Server HTTP Invoker Remote Code Execution Vulnerability
H3-2022-0001	Web Application Cross Site Scripting Vulnerability
H3-2022-0002	Azure Multi-Factor Authentication Disabled
H3-2022-0003	Remote Desktop Protocol (RDP) Port Exposed to the Internet
H3-2022-0004	Server Message Block (SMB) Port Exposed to the Internet
H3-2022-0005	Secure Socket Shell (SSH) Port Exposed to the Internet
H3-2022-0006	Database Port Exposed to the Internet
H3-2022-0007	Telnet Port Exposed to the Internet
H3-2022-0008	File Transfer Protocol (FTP) Port Exposed to the Internet
H3-2022-0009	Simple Network Management Protocol (SNMP) Port Exposed to the Internet
H3-2022-0010	Risky Port Exposed to the Internet
H3-2022-0012	Unauthenticated Access to Jira Dashboards
H3-2022-0015	Web Application Path Traversal Vulnerability
H3-2022-0016	Active Directory Certificate Services Misconfiguration Privilege Escalation - Subject Alternative Name
H3-2022-0017	Active Directory Certificate Services Misconfiguration Privilege Escalation - Any Purpose or No (aka SubCA) EKU Misconfiguration

Weakness ID	Name
H3-2022-0018	Active Directory Certificate Services Misconfigured Enrollment Agent Template
H3-2022-0019	Active Directory Certificate Services Misconfigured Template Requires Enrollment Agent Signature
H3-2022-0020	Active Directory Certificate Services Misconfigured Template Access Controls
H3-2022-0021	Active Directory Certificate Services Domain Escalation via Vulnerable PKI AD Object Access Controls
H3-2022-0022	Active Directory Certificate Services - EDITF_ATTRIBUTESUBJECTALTNAME2 flag set
H3-2022-0023	Active Directory Certificate Services: Vulnerable Certificate Authority Access Control
H3-2022-0024	Active Directory Certificate Services Misconfiguration: NTLM Relay to AD CS HTTP Endpoint
H3-2022-0025	Unauthenticated Access to Kibana
H3-2022-0026	Unauthenticated Access to Kubeflow
H3-2022-0027	Unauthenticated Access to Jupyter
H3-2022-0028	Unauthenticated Access to Apache Solr
H3-2022-0029	Unauthenticated Access to ThoughtWorks GoCD
H3-2022-0030	Unauthenticated Access to Paessler PRTG Network Monitor
H3-2022-0031	Unauthenticated Access to Mongo Express
H3-2022-0032	Unauthenticated Access to Prometheus Alertmanager
H3-2022-0033	Unauthenticated Access to Jenkins People Directory
H3-2022-0034	Anonymous Access to Zoho ManageEngine ADManager Plus Employee Search
H3-2022-0035	Unauthenticated Access to JavaMelody Monitoring Console
H3-2022-0036	Guest Access to Zabbix Dashboards
H3-2022-0037	Laravel Debug Mode Enabled
H3-2022-0038	Ruby on Rails Debug Mode Enabled
H3-2022-0039	Golang pprof Debugging Endpoint Enabled
H3-2022-0040	Symfony Debug Mode Enabled
H3-2022-0041	Symfony Profiler Enabled
H3-2022-0042	Django Debug Mode Enabled
H3-2022-0043	Backup File Exposure
H3-2022-0044	Shell History File Exposure
H3-2022-0045	PHPinfo Page Exposed
H3-2022-0046	Rails Database Configuration File Exposure
H3-2022-0047	Apache Tomcat Example Scripts Exposed
H3-2022-0048	Apache Web Server Configuration File Exposure
H3-2022-0049	IIS web.config File Exposure
H3-2022-0050	PHP-FPM Configuration File Exposure
H3-2022-0051	Symfony Configuration File Exposure

Weakness ID	Name
H3-2022-0052	Ansible Configuration File Exposure
H3-2022-0054	CGI Test Script Exposed
H3-2022-0055	phpMyAdmin Setup Page Exposed
H3-2022-0056	Anonymous Deployment Privileges in JFrog Artifactory
H3-2022-0057	jQuery File Upload Widget Exposed
H3-2022-0058	Jolokia Local File Inclusion Misconfiguration
H3-2022-0059	Spring Boot Configuration Properties Actuator Exposed
H3-2022-0060	Spring Boot Env Actuator Exposed
H3-2022-0061	Apache Web Server htpasswd File Exposure
H3-2022-0062	Microsoft FrontPage service.pwd File Exposure
H3-2022-0063	Private Keys Exposed on Web Server
H3-2022-0064	Rails Secret Token Exposure
H3-2022-0065	Unauthenticated Access to Apache Airflow
H3-2022-0066	Git Repo Exposed on a Web Server
H3-2022-0067	Weak or Default Credentials - MongoDB
H3-2022-0068	Airflow Configuration Exposure
H3-2022-0069	Web Directory Listing
H3-2022-0070	Anonymous MongoDB Access
H3-2022-0071	Jenkins Self-Signup Enabled
H3-2022-0072	Apache Airflow Debug Mode Enabled
H3-2022-0073	Microsoft Windows Machine Account NTLM Coercion via Authenticated LSARPC Spoofing
H3-2022-0074	AWS Assume Role Access
H3-2022-0075	Public-Facing Application Exposed with HTTP Basic Authentication
H3-2022-0076	Unauthenticated AWS Cognito Role Has Non-Standard Permissions
H3-2022-0078	Unauthenticated Gitlab User Enumeration
H3-2022-0079	Credential Dumping - AWS Instance Metadata Service v2
H3-2022-0080	WordPress Unauthenticated User Enumeration
H3-2022-0081	Atlassian Jira Unauthenticated User Enumeration via the User Picker Browser
H3-2022-0082	Exposed Kubernetes Version
H3-2022-0083	Anonymous Access to the Kubernetes Dashboard
H3-2022-0084	Credential Reuse - Windows Local Administrator Accounts
H3-2022-0085	Credential Reuse - Shared Windows Local User and Domain User Accounts
H3-2022-0086	Domain User with Local Administrator Privileges
H3-2022-0087	Password Reuse
H3-2022-0088	Public Access to Amazon EC2 AMI

Weakness ID	Name
H3-2022-0089	Public Access to Amazon EBS Snapshot
H3-2022-0090	Public Access to Amazon RDS Snapshot
H3-2022-0093	Weak or Default Credentials - Cracked Credentials from Active Directory Services Database (NTDS)
H3-2022-0095	Password Reuse Found in Active Directory Services Database (NTDS)
H3-2023-0002	Flask Authentication Bypass Misconfiguration
H3-2023-0003	Pre-Windows 2000 Computer Set
H3-2023-0008	AWS Multi-Factor Authentication Disabled
H3-2023-0009	Kerberos Unconstrained Delegation
H3-2023-0010	Kerberos Constrained Delegation
H3-2023-0011	Microsoft Windows Machine Account NTLM Coercion via EventLog Remoting Protocol Manipulation
H3-2023-0012	Microsoft Windows Machine Account NTLM Coercion via Print Spooler Protocol Manipulation
H3-2023-0013	Authenticated Microsoft Windows Machine Account NTLM Coercion via File Server Remote VSS Protocol Manipulation
H3-2023-0014	Authenticated Microsoft Windows Machine Account NTLM Coercion via Distributed File System Namespace Management Protocol Manipulation
H3-2023-0015	Authenticated Microsoft Windows Machine Account NTLM Coercion via EventLog Remoting Protocol Manipulation
H3-2023-0016	Authenticated Microsoft Windows Machine Account NTLM Coercion via Print Spooler Protocol Manipulation
H3-2023-0017	Microsoft Windows Machine Account NTLM Coercion via File Server Remote VSS Protocol Manipulation
H3-2023-0018	Microsoft Windows Machine Account NTLM Coercion via Distributed File System Namespace Management Protocol Manipulation
H3-2023-0019	Credential Dumping - Data Protection API (DPAPI) Secrets
H3-2023-0020	PaperCut File Upload Remote Code Execution Vulnerability
H3-2023-0021	Phished Credential
H3-2023-0022	PaperCut Arbitrary File Read and Deletion Vulnerability

3.8 Injecting Credentials

3.8.1 Injecting Credentials

NodeZero can run pentests from a compromised user perspective. This type of perspective shows the impact an attacker would have if leveraging a specific set of assumed compromised credentials.

Users can run an authenticated pentest by injecting credentials into the pentest via the Real-Time View, as described below.

NodeZero uses injected credentials in ways that emulate how an attacker may use credentials they compromise. This feature allows users to execute "what if" scenarios to see what impacts may result from compromised credentials.

Why Inject Credentials?

WHAT IF EMPLOYEE X WAS PHISHED?

No matter how advanced our network's technological defenses, humans have been and will continue to be a popular attack vector through which attackers can gain initial access. Whether through phishing attacks or other forms of social engineering, we should expect that user credentials may fall into an attacker's hands.

If your organization performs phishing exercises you may identify a set of credentials that are prone to being phished. By injecting their credentials into a pentest NodeZero can generate a complete picture of the potential impacts of a successful phishing attack.

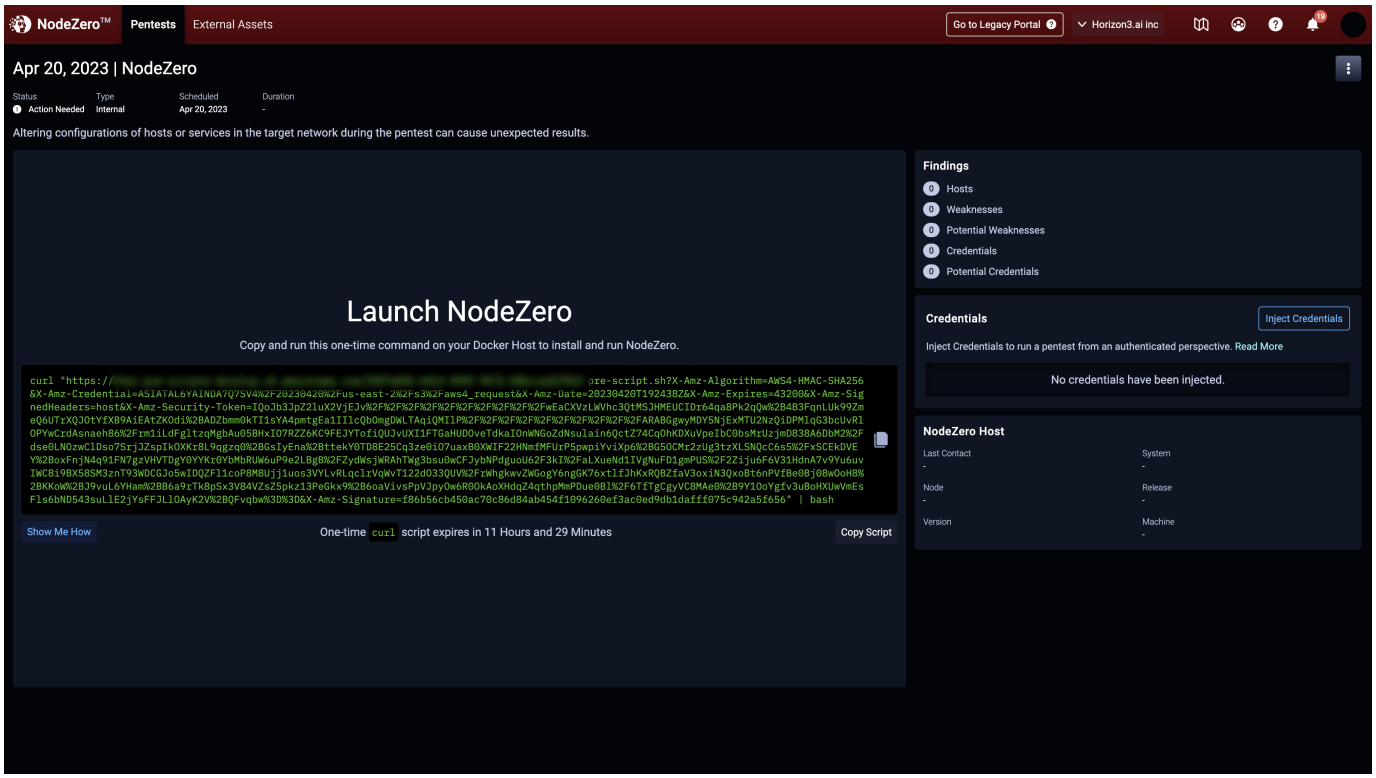
WHAT IF EMPLOYEE X GOES ROGUE?

We all like to believe we can trust our employees and co-workers, but at Horizon3 we encourage our users to ~~Trust but~~ Verify. It is important to implement and verify access policies that use the concept of least privilege: users and service accounts should only have access to the resources to which they need access.

By injecting a credential for a user or service account into a pentest, NodeZero can generate a complete picture of what resources that account has access to.

How to Inject Credentials

Users can inject credentials through the `Real-Time View` immediately after launching a pentest.

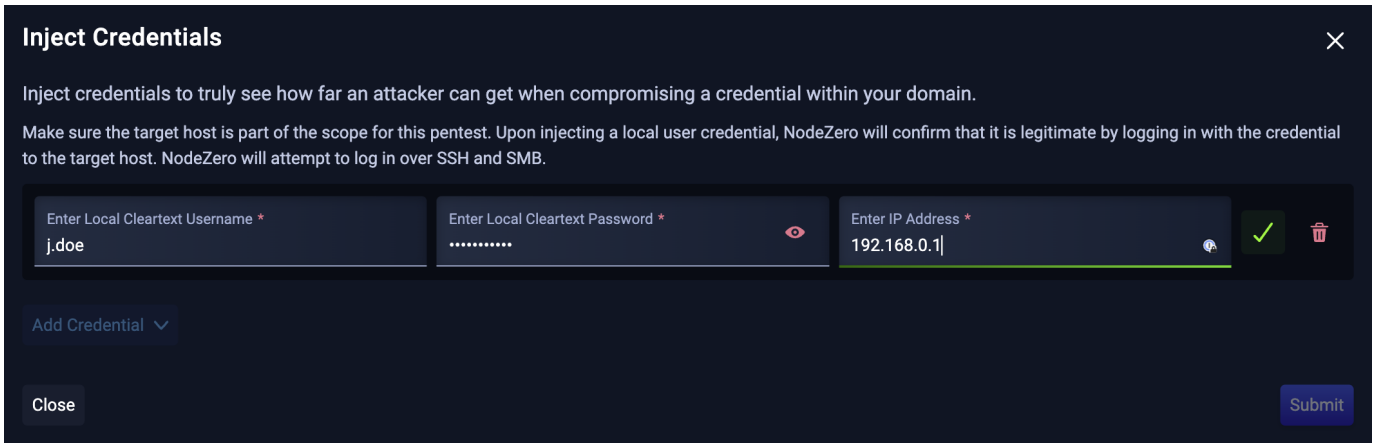

NodeZero™
Pentests
External Assets
Go to Legacy Portal
Horizon3.ai inc
Apr 20, 2023 | NodeZero
Status
Action Needed
Type
Internal
Scheduled
Apr 20, 2023
Duration
-
Altering configurations of hosts or services in the target network during the pentest can cause unexpected results.
Launch NodeZero
Copy and run this one-time command on your Docker Host to install and run NodeZero.
curl "https://
pre-script.sh?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=ASIATAL6YAINDA7Q7SV4%2F20230420%2Fus-east-2%2Fs3%2Faws4_request&X-Amz-Date=20230420T192438Z&X-Amz-Expires=43200&X-Amz-Sig
nedHeaders=host&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEJv%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJHMEUCIDr64qa8Pk2qQw%2B4B3FqnLUk99Zm
eQ6UTrXQJOtYfXB9AiEAtZKOdi%2BADZbmm0kTI1sYA4pmtgEaIIllcQbOmgDWLTAqiQMIlP%2F%2F%2F%2F%2F%2F%2F%2F%2FARABGgwyMDY5NjExMTU2NzQiDPMlqG3bcUvRl
OPYwCrdAsnaeh86%2Frm1ildFgltzqMgbAu05BHxIO7RZZ6KC9FEJYTofiQUJvUXI1FTGaHUDOveTdkaIOnWNGoZdNsulain6QctZ74CqOhKDXuVpeIbC0bsMrUzjmD838A6DbM2%2F
dse0LNOzwClDso7SrjJZspIkOXKr8l9qgzq0%2BGsIyEna%2BttekYQTD8E25Cq3ze0iO7uaxB0XWIF22HNmfMFUrP5pwpiYviXp6%2BG5OCMr2zUg3tzXLSNQcC6s5%2FxSCEkDVE
Y%2BoxFnjN4q91FN7gzVHVTDgYOYYKr0YbMbRUW6uP9e2LBgB%2FZydWsjWRAhTWg3bsu0wCFJybNPdguoU62F3kI%2FaLXueNd1IVgNuFD1gmPUS%2F2Ziju6F6V31HdnA7v9Yu6uv
IWC8i9BXS8SM3znT93WDCGJo5wIDQZFI1coP0M8Ujj1uos3VYLwRlqclrVqWvTI22dO33QUV%2FrWhgKwvZWGogY6ngGK76xtlfJhKxRQBZfaV3oxiN3QxoBt6oPVfBe08ji08wDoH8%
2BKKoW%2BJ9vuL6YHam%2B86a9rTk8pSx3V84VZsZ5pkz13PeGkx9%2B6oaVivsPpVJpyOw6ROOkAoXHdqZ4qthpMmPDue08l%2F6TfTgCgyVC8MAe0%2B9Y1OoYgfv3uBoHXUwVmEs
Fls6bND543suLlE2jYsFFJllOAyK2V%2BQFvqbw%3D%3D&X-Amz-Signature=f86b56cb450ac70c86d84ab454f1096260ef3ac0ed9db1daffd075c942a5f656" | bash
Show Me How
One-time curl script expires in 11 Hours and 29 Minutes
Copy Script
Findings
0 Hosts
0 Weaknesses
0 Potential Weaknesses
0 Credentials
0 Potential Credentials
Credentials
Inject Credentials
Inject Credentials to run a pentest from an authenticated perspective. Read More
No credentials have been injected.
NodeZero Host
Last Contact
-
System
-
Node
-
Release
-
Version
Machine
-

Click the `Inject Credentials` button to open the Inject Credentials modal. In the modal, choose a credential type from the `Add Credential` drop-down to add a new credential. The supported credential types are shown in the table below.

Type	Description	Example
Domain User: Cleartext	Cleartext credentials for an Active Directory domain user. If there is not a domain controller in scope, NodeZero will not attempt to use this credential. An attacker may compromise this type of credential through various means including phishing, social engineering, key logging, or password guessing.	Username: `john.doe` Password: `MyPassword123`
Domain User: NTLM Hash	The NTLM Hash for an Active Directory domain user. If there is not a domain controller in scope, NodeZero will not attempt to use this credential. An attacker may compromise this type of credential if they were able to dump the SAM or NTDS database on a domain controller.	Username: `jdoe` Hash: `31d6cfe0d16ae931b73c59d7e0c089c0`
Local User: Cleartext	Cleartext credentials for a local Windows or Linux user. These credentials include the IP address of the local machine and will be used to attempt login over SSH and SMB. An attacker may compromise this type of credential through various means including phishing, social engineering, key logging, or password guessing.	Username: `jdoe2` Password: `MyPassword123` IP Address: `10.0.0.1`
Local User: NTLM Hash	The NTLM Hash for a local Windows user. These credentials include the IP address of the local machine and will be used to attempt logins over SMB. An attacker may compromise this type of credential if they are able to dump the SAM database on a local Windows machine.	Username: `Administrator` Hash: `31d6cfe0d16ae931b73c59d7e0c089c0` IP Address: `192.168.0.1`
AWS User: Access Keys	An AWS access key and secret access key. By injecting an AWS credential, all cloud resources belonging to the associated AWS ID will be considered in scope. An attacker may compromise this type of credential by finding it on a compromised machine or file share, as they are commonly stored in files in the user's directory.	Access Key ID: `AKIASP2TPHJSVM75TWVN` Secret Access Key: `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`

Tip

While you cannot inject an AWS Role in the `Real-Time View`, there is another way this can be accomplished. See: Injecting an AWS Role.



After entering the credential details, click the green checkmark. Add more credentials as desired, then click `Submit` to send the credentials to NodeZero.

You can view the status of credentials in the Real-Time View using the icons next to each credential. Credentials states are shown below:

Icon	State	Description
	Pending	The credential has been submitted but has not yet been received by NodeZero. This state may occur when a pentest first begins until the ephemeral infrastructure is fully deployed.
	Received	The credential has been received by NodeZero. If NodeZero has attempted to use the credential but authentication was unsuccessful, it will remain in the Received state.
	Confirmed	The credential has been used by NodeZero to successfully authenticate within the target environment.

Tip

You may continue to inject credentials in the Real-Time View until the pentest completes and enters the Processing state. Injecting credentials may extend the duration of a pentest, but the best way to minimize the pentest operation runtime is to inject credentials early in the pentest.

FAQ

Is it safe to inject credentials in NodeZero?

Yes. Horizon3 takes the security of injected credentials seriously.
Once injected, credentials are securely transferred to NodeZero's ephemeral environment, which is dedicated to a single pentest and is destroyed once the pentest is completed. Sensitive parts of credentials (e.g. plaintext passwords, hashes, private keys, etc.) are never stored in persistent databases.

How many credentials can I inject?

NodeZero supports injecting up to twenty total credentials per pentest.

Are injected credentials used when re-running a pentest?

No. Since injected credentials are destroyed at the end of a pentest, NodeZero cannot use them when re-running a pentest.
However, in the Real-Time View you can see descriptions of the credentials that were used in the previous pentest. We encourage users to re-enter these credentials to get a comparable pentest experience.

Does injecting credentials increase the duration of a pentest?

There are many factors that affect pentest duration, the most significant of which are number of live hosts, services, and web applications. When credentials are injected, NodeZero will attempt to authenticate with the credentials, and if successful will perform various post-exploitation tasks such as enumerating shares and dumping credentials on compromised hosts, which can lead to further discoveries that may extend the duration of the pentest.

The best way to minimize the pentest runtime is to inject credentials early in the pentest.

3.8.2 Injecting an AWS Role

AWS roles can be used to assign permissions to various AWS services. Injecting an AWS role can help understand the impact of one of these roles being compromised.

Example

What would happen if someone hacked into our web application running on that EC2 instance?

Users can inject an AWS role into a pentest by running NodeZero on an EC2 instance and assigning that EC2 instance an IAM instance profile.

When the pentest starts, NodeZero will query the IMDS and treat the EC2 instance role as an injected credential. By injecting an AWS credential, all cloud resources belonging to the associated AWS ID will be considered in scope.

Since an EC2 instance can only have one IAM instance profile assigned, and because an IAM instance profile can only contain one role, it is only possible to inject one AWS role per pentest.

Creating an IAM Instance Profile

An instance profile is what attaches an AWS role to an EC2 instance.

If you create an AWS Role in the AWS Management Console for an EC2 instance (assign the EC2 service as the trusted entity) an IAM instance profile with the same name will also be created. This means that often you do not need to manually create an IAM instance profile.

If you created the AWS Role with the AWS CLI or if the role was not created for an EC2 instance you will need to manually create an IAM instance profile.

To manually create an IAM instance profile you will need to use the AWS CLI. The commands below show how you would create an IAM instance for a role named `ExampleRole`.

```
aws iam create-instance-profile --instance-profile-name ExampleRole
aws iam add-role-to-instance-profile --instance-profile-name ExampleRole --role-name ExampleRole
```

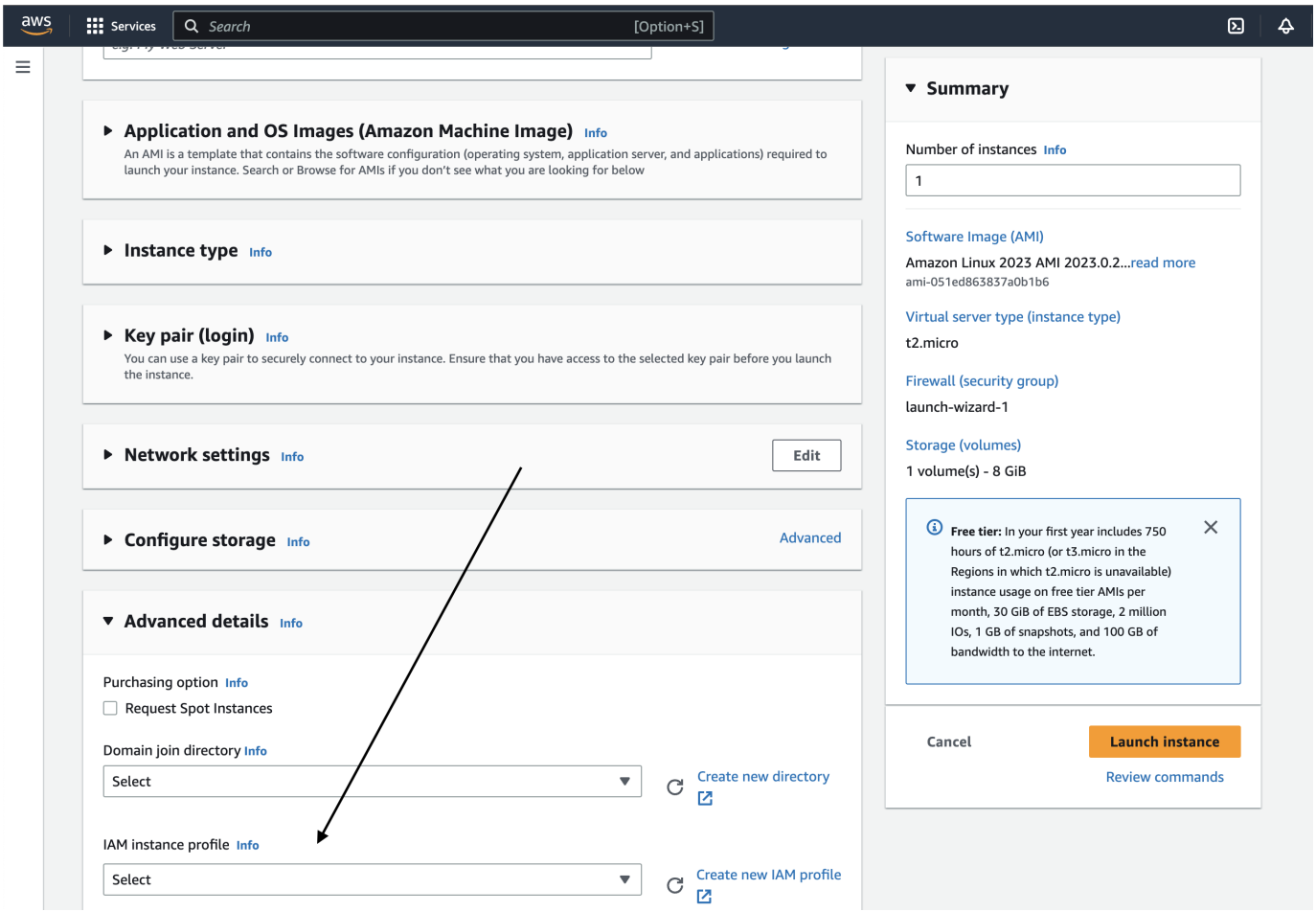
Tip

The role assigned to the instance profile must have the EC2 service specified in the trust relationship policy.

Assigning an IAM Instance Profile to an EC2 Instance

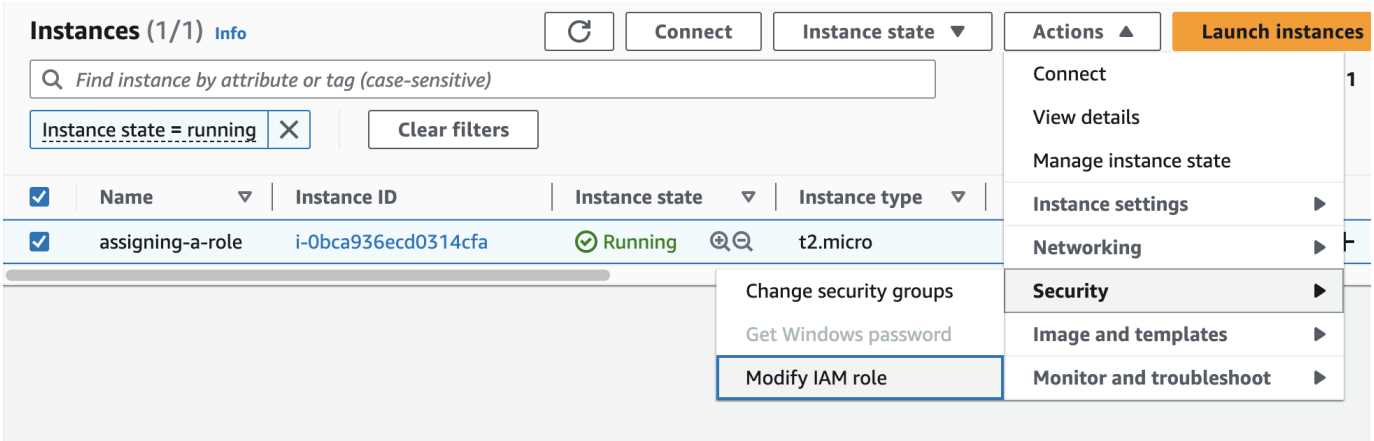
Method 1 - AWS Management Console - Assign Instance Profile when Launching a New Instance

Launch a new EC2 instance using the `Launch an instance` flow, find the `Advanced details` section and select an IAM instance profile from the drop-down.

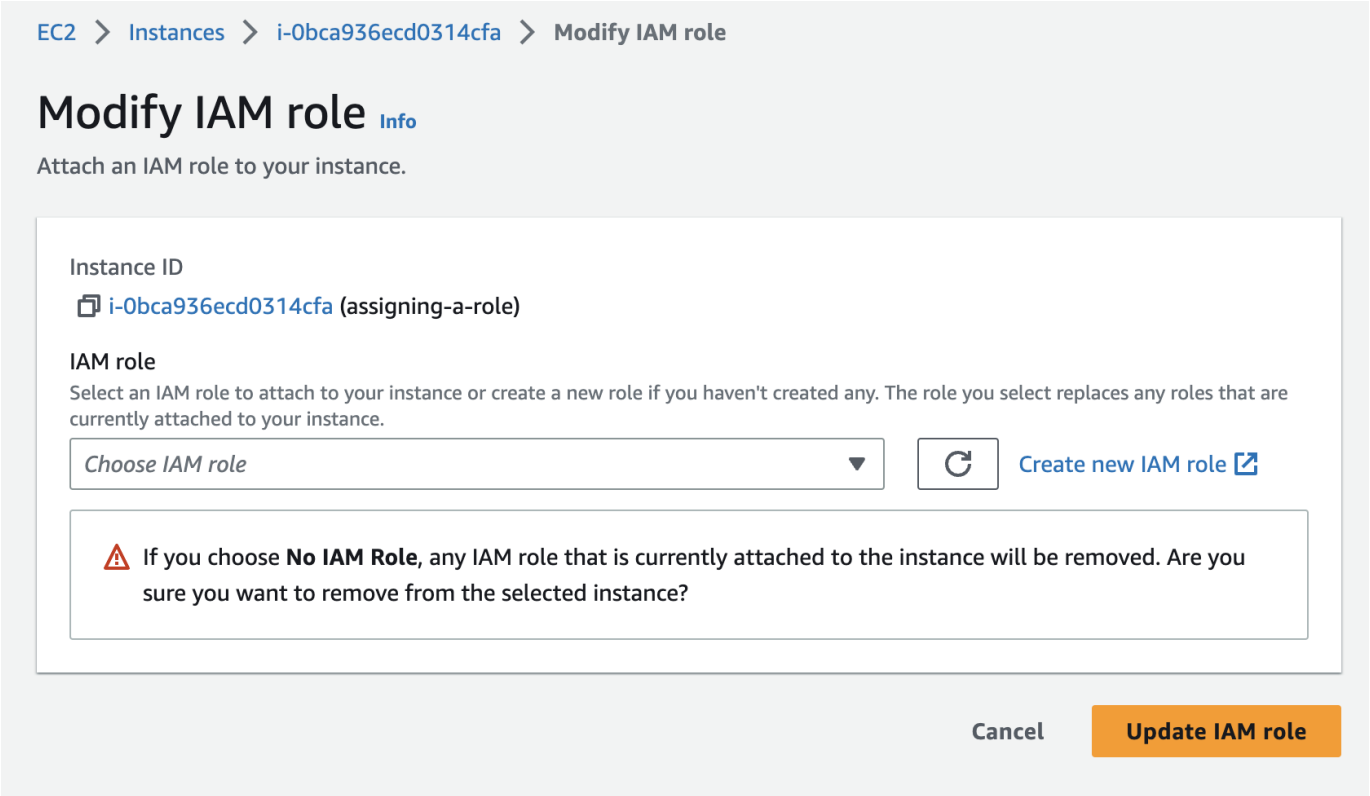


Method 2a - Assign Instance Profile to an Existing EC2 Instance - AWS Management Console

From the `EC2 Dashboard` or from the `Instance Details` page select an EC2 instance, click the `Actions` drop-down menu, click the `Security` sub-menu, and finally select `Modify IAM Role`.



From the `Modify IAM role` page, select the IAM role you want to inject from the drop-down and click `Update IAM role.`



Method 2b - Assign Instance Profile to an Existing EC2 Instance - AWS CLI

AWS CLI Command:

```
aws ec2 associate-iam-instance-profile --instance-id i-123456789abcde123 --iam-instance-profile Name=admin-role
```

3.9 Phishing Impact Test

3.9.1 Phishing Impact Test

The NodeZero Phishing Impact Test is designed to help you measure and understand the impact of successful phishing attack within your organization, starting with your most phish-prone employees.

Why should I run a NodeZero Phishing Impact Test?

No matter how secure we make our network perimeter, attackers will find ways to exploit the humans that use our networks to gain initial access. Hopefully, your organization provides education and training to employees to reduce the likelihood of successful phishing attacks, but inevitably some slip through the cracks.

So the question is not "what happens if someone gets phished?", but "what happens when someone is phished?".

This is where NodeZero's Phishing Impact Tests come in!

This test is intended to supplement your organization's simulated phishing tools to not just identify who my be susceptible to phishing, but to also understand what happens when they get phished.

NodeZero will automatically capture the credentials of simulated phishing attack victims and use them to pentest your internal network. The report from this test can be use to assess the business risk of a successful phishing attack, and identify security controls that can be put in place to mitigate this risk.

How does a NodeZero Phishing Impact Test work?

The Phishing Impact Test works a lot like an Internal Pentest, but enables NodeZero to automatically capture and use credentials captured through your organization's Phishing Simulation Tool.

At a high-level:

- NodeZero is deployed within your organization's network
- NodeZero generates a JavaScript payload to be copied into the landing page for your simulated phishing campaign
- You deploy your simulated phishing campaign, sit back, and wait for a bite
- When a user is phished, the credentials are captured and directly injected into NodeZero's ephemeral infrastructure
- NodeZero uses the phished credentials for perform an authenticated pentest within your network
- NodeZero generates a report to convey the impacts of the phishing attack

How to Run a NodeZero Phishing Impact Test

0. BEFORE YOU BEGIN

- You will need access to both the NodeZero Portal and your organization's Phishing Simulation Tool (e.g. KnowBe4, ProofPoint, etc.)
- Plan out the details of your phishing campaign
- How long will the campaign last?
- What is the target group for this phishing campaign?
- What type of phishing email(s) am I sending?
- What landing page(s) will the phishing emails link to?

Campaign Duration

NodeZero currently supports phishing campaigns up to 28 days.

1. CREATE A NEW PENTEST

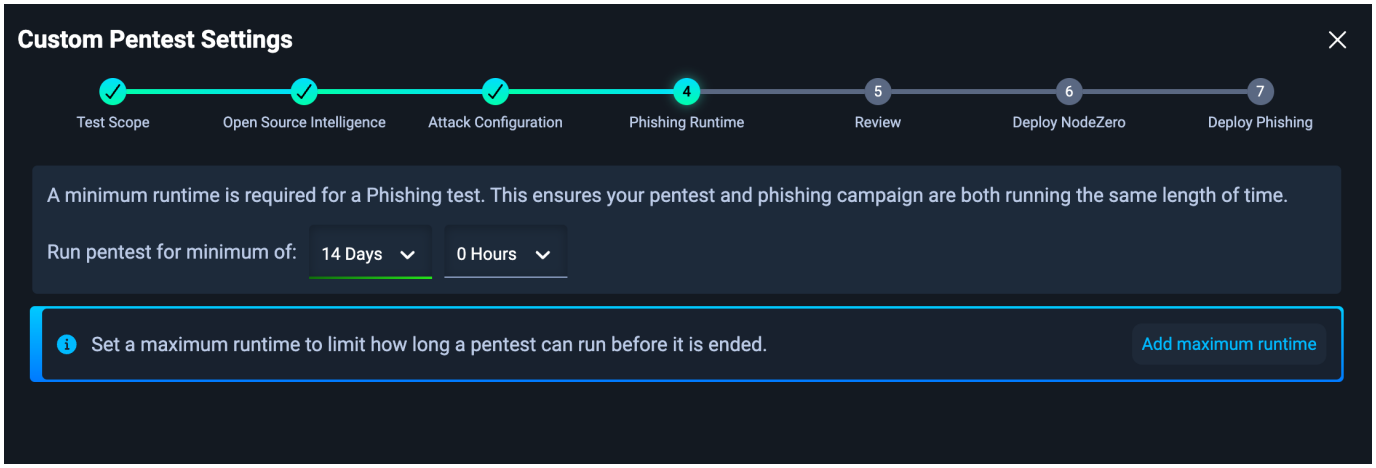
Open the NodeZero Portal and navigate to the Pentests tab.

Click `+ RUN PENTEST` to open the Pentest Configuration and select `Phishing Campaign`.

2. CONFIGURE THE PHISHING IMPACT TEST

The pentest configuration for a Phishing Impact Test is very similar to an Internal Pentest, so it should be familiar.

A key difference from the Internal Pentest is that you must set a phishing campaign runtime. This should be **at least as long** as your phishing campaign within your Phishing Simulation Tool.



Tip

Consider your phishing targets when configuring your pentest scope. Include networks in which your phishing targets likely have access. Also consider including networks to which they definitely should NOT have access. NodeZero can help validate if your assumptions and security controls are correct.

3. DEPLOY NODEZERO

Run the curl script on your NodeZero host to kick off the Phishing Impact Test. Or, if you are using a NodeZero runner, the test will begin automatically.

Don't have a NodeZero host?

See our documentation in Setup NodeZero Host.

This script will validate the Docker installation, download the most up-to-date NodeZero Docker image, and begin the test.

4. CONFIGURE THE PHISHING LANDING PAGE

4.1 Copy NodeZero Phishing Script and embed at the bottom of the HTML

The NodeZero Phishing Script should be copied to the bottom of the HTML file right above the closing `</body>` element. In some cases phishing templates are missing the `<body>` tag. In such cases putting the script at the bottom on the HTML will suffice.

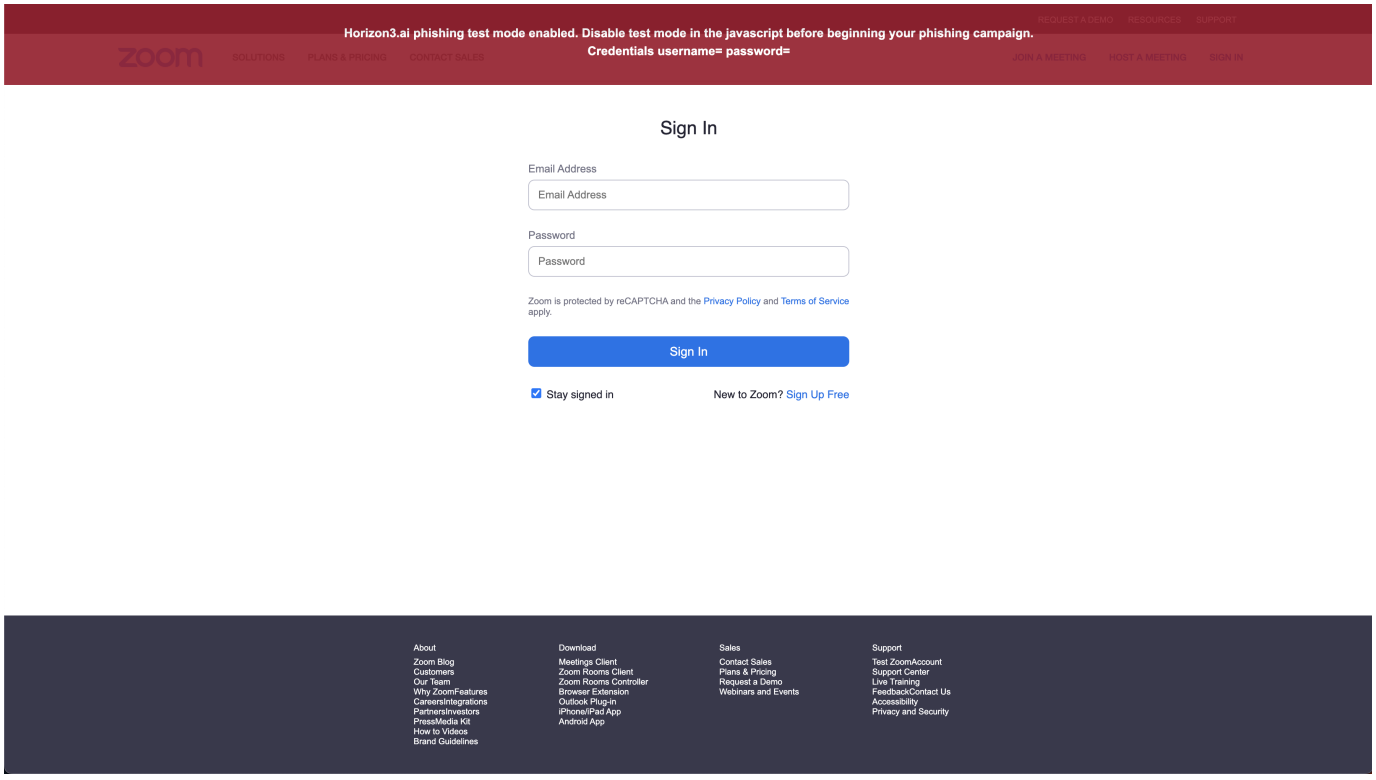
The example HTML page below shows where to paste the script, and can be used to try it out yourself.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Horizon3.ai Phishing Impact Test Page</title>
</head>
<body>
<style type="text/css">
  * { box-sizing: border-box; }
  body { font-family: "Helvetica Neue"; font-size: 15px; color: #000; }
  h1 { font-size: 34px; color: #000; font-weight: 200; }
  form { width: 100%; max-width: 356px; margin: 0 auto; display: block; }
  input[type="text"],input[type="password"] { border: 2px solid rgba(0, 0, 0, 0.4); background-color: rgba(255, 255, 255, 0.4); color: #000; font-
size: 15px; padding: 6px 8px; margin-bottom: 12px; }
  input[type="text"]:focus, input[type="password"]:focus { border-color: #0078d7; background-color: #fff; }
  input[type="text"], input[type="password"], input[type="submit"] { width: 100%; }
  #logon { background-color: #0078d7; border: 0; color: #fff; padding: 8px 12px; text-align: center; width: 100%; font-size: 15px; }
  #logon:hover { background-color: #004e8c; cursor: pointer; }
</style>
<center>
  <img alt="Horizon3.ai logo." height="25" src="https://p7i3u3x3.rocketcdn.me/wp-content/uploads/2022/06/Horizon3ai_Logo_Tagline_Horizontal_RGB-
WhiteTxt.png.webp" style="width: 200px; height: 50px; margin: 2% auto 0 auto; background-color: #000;" width="100" />
  <h1>Sign in</h1>
  <p>Use your Horizon3.ai account.<br /></p>
  <form action="/pages/19906e20e8a60c71badf8f94151c19a2" id="loginform" method="post">
    <input id="username" name="username" placeholder="Email" required="" type="text" />
    <input id="password" name="password" placeholder="Password" required="" type="password" />
    <input id="logon" name="logon" type="submit" value="Sign in" /> 
  </form>
</center>

<!-- PASTE NODEZERO PHISHING SCRIPT HERE -->

</body>
</html>
```

Some phishing templates may already contain JavaScript and there could be interference between the scripts. Carefully review any existing scripts to identify conflicts. Remove any unnecessary `<script>` tags in the template. In most cases, the only `<script>` tag should be the NodeZero Phishing Script.

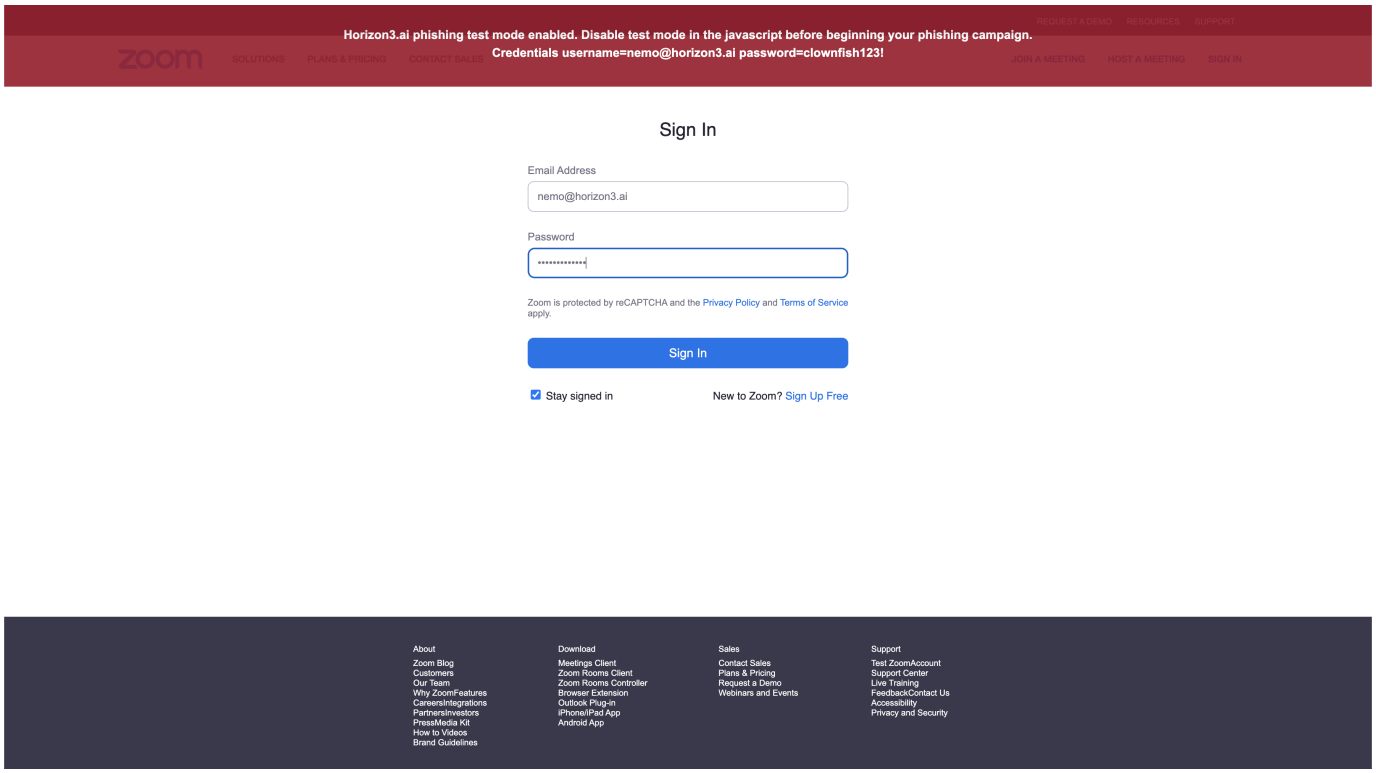
4.2 Preview the landing page

Open the landing page to preview and test it. Notice that there is a red banner explaining Horizon3.ai Phishing Impact test mode enabled. If you do not see a red banner check the script for `const testModeEnabled = true`.



4.3 Test the landing page credential detection

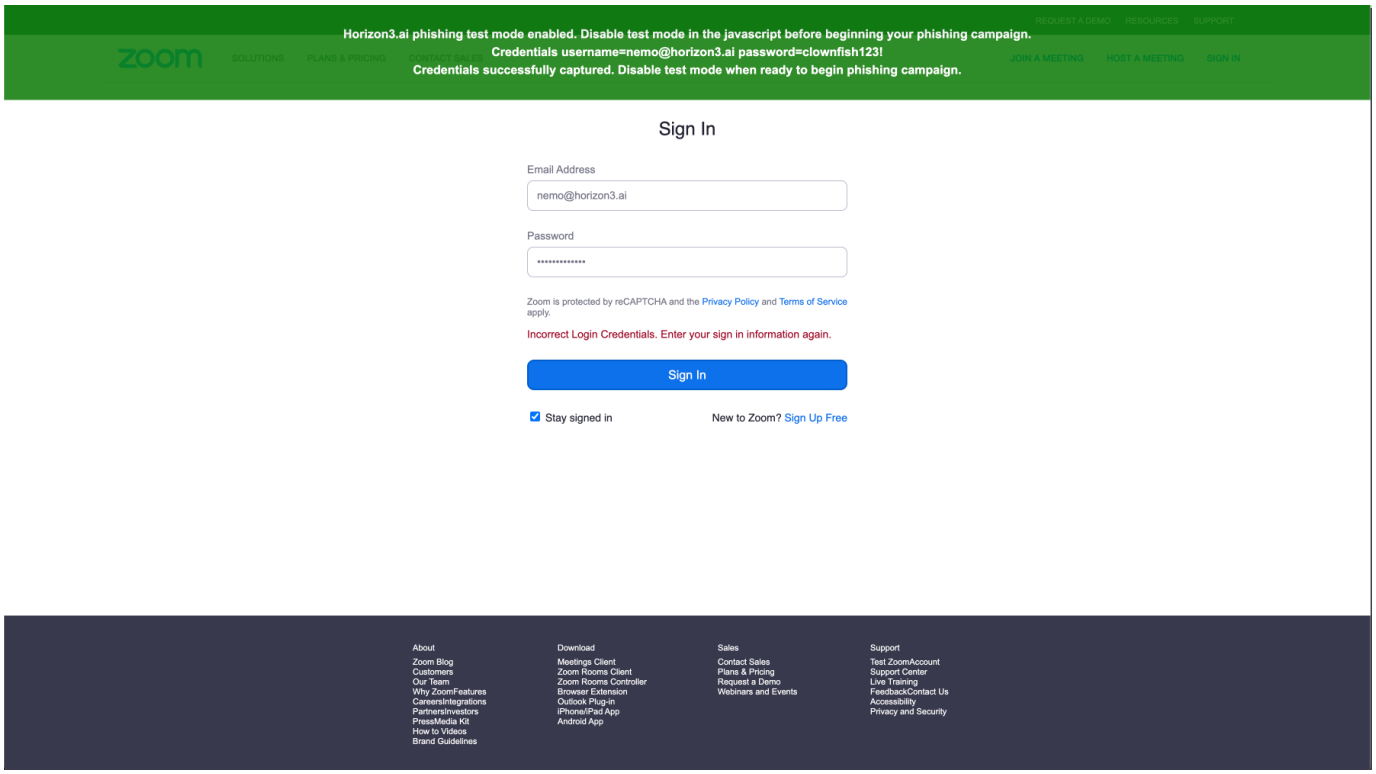
From the landing page preview, enter a username and password. The credentials will be reflected to you in the red test banner as you type.



If you do not see credentials when typing into the input fields, then there is an issue with the script identifying the right inputs.

4.4 Test the landing page credential submission

From the landing page preview, submit a set of credentials. If the script is working properly the banner will turn green. At this point, you have validated that the script is correctly capturing and submitting credentials to NodeZero.



Additionally, you may notice a red form validation error that reads `Incorrect Login Credentials. Enter your sign in information again.` This is added by the Horizon3.ai phishing script in an attempt to trick the user into submitting another variation of credentials that can also be used during the Pentest. You can disable this feature if you like by changing the following variable to false `const showIncorrectCredentialsError = false`.

4.5 Disable test mode

Revisit the landing page source code and change the line

```
const testModeEnabled = true
```

with

```
const testModeEnabled = false
```

Refresh the landing page preview and ensure that the banner is no longer present. If you do not change this variable, then the red banner will show to phished users.

Your landing page is now armed and ready for your phishing campaign.

Having issues with the NodeZero Phishing Script?

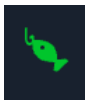
See our documentation in How to use the NodeZero Phishing Script.

5. LAUNCH YOUR PHISHING CAMPAIGN

Launch the Phishing Campaign in your Phishing Simulation Tool.

As phished credentials are received, they will appear in Real-Time View and automatically be used by NodeZero.

You can view the status of credentials in the Real-Time View using the icons next to each credential. Credentials states are shown below:

Icon	State	Description
	Pending	The phished credential has been submitted but has not yet been received by NodeZero. This state may occur when the pentest first receives the credential.
	Received	The phished credential has been received by NodeZero. If NodeZero has attempted to use the credential but authentication was unsuccessful, it will remain in the Received state.
	Confirmed	The phished credential has been used by NodeZero to successfully authenticate within the target environment.

FAQ

Is it safe to send phished credentials to NodeZero?

Yes. Horizon3 takes the security of phished credentials seriously.
Once phished, credentials are securely transferred to NodeZero's ephemeral environment, which is dedicated to a single pentest and is destroyed once the pentest is completed. Sensitive parts of credentials (e.g. plaintext passwords, hashes, private keys, etc.) are never stored in persistent databases.

How many credentials can I phish in a single Phishing Impact test?

NodeZero supports phishing up to 100 total credentials per Phishing Impact test. Credentials that exceed this limit may show up in the user interface but will not be used by NodeZero. If you find that your Phishing Impact tests are exceeding this limit, consider reducing the number of emails in your phishing campaign or reach out to support to discuss a limit increase.

3.9.2 The NodeZero Phishing Script

Setting up your first phishing pentest requires understanding of the NodeZero Phishing Script, how it works, and what it can do. This page helps answer those questions.

How it works

The NodeZero Phishing Script works by identifying the username and passwords inputs on the login page and submitting them to NodeZero.

The phishing script is currently setup to work on login pages that have a form containing a username/email, password, and a submit button. It does not capture other fields.

Since login pages vary we need a flexible way to capture what input is the username, password, and submit. The script leverages an allowlist of words within the script to automatically find the username, password, and submit inputs. If the script cannot detect these fields then it will require light updates and troubleshooting to the script.

Troubleshooting the NodeZero Phishing Script

This section provides tips for troubleshooting problems with the NodeZero Phishing Script. If you have additional issues or questions please reach out to Horizon3.ai support.

THE RED TEST BANNER IS NOT GOING AWAY

Open the landing page source code and ensure it contains the following line.

```
const testModeEnabled = false
```

If the value is set to `true`, change it to `false`.

If you cannot find the line on the page, regenerate and replace the phishing script into the login page. Then, find the line and change the value from `true` to `false`.

After updating the source code, be sure to press the Refresh button on your browser to reload the page.

CREDENTIALS ARE NOT SHOWING UP ON THE RED BANNER IN TEST MODE

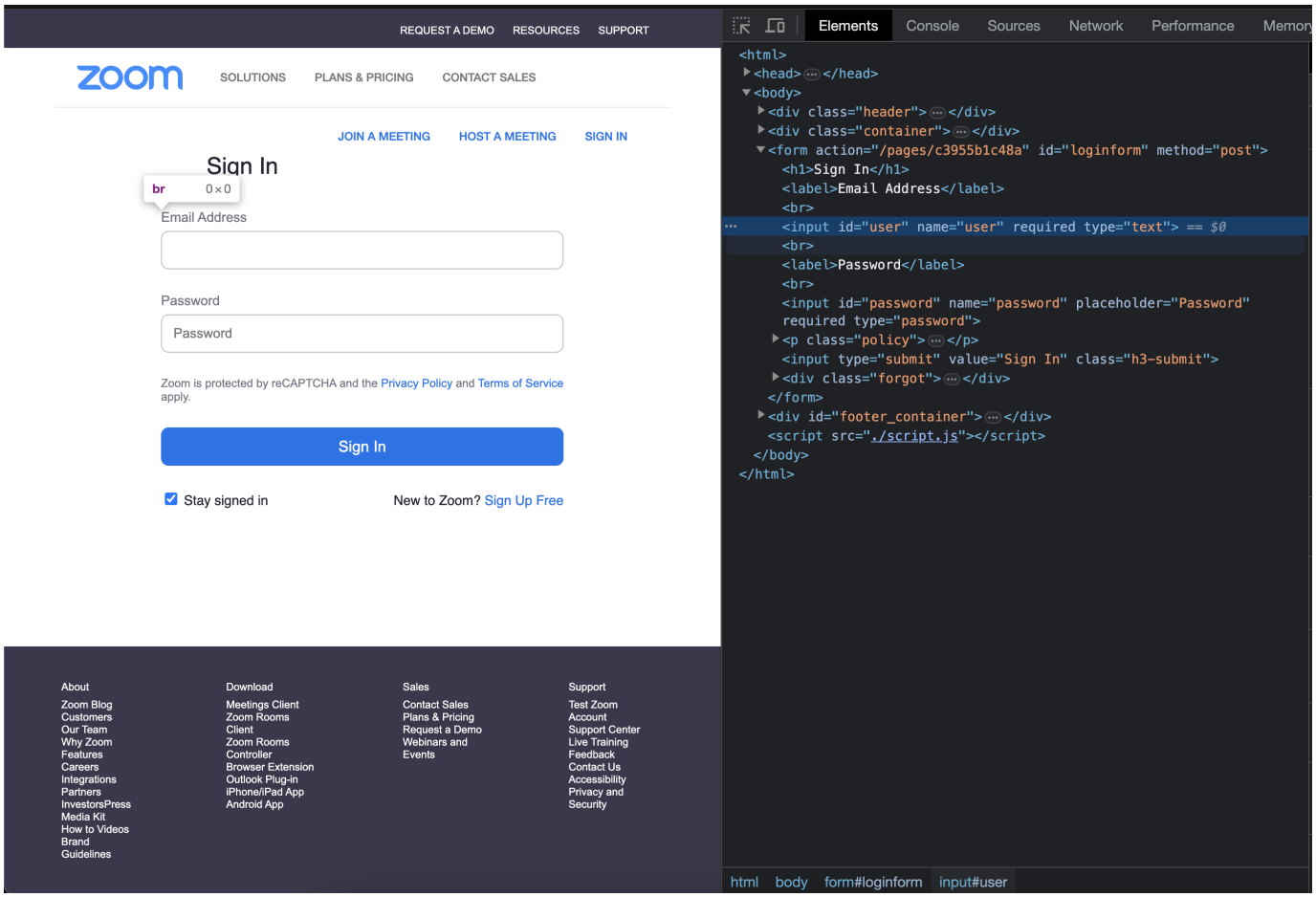
If you notice that the script is not reflecting credentials as you type, then the script is not detecting the form inputs on the landing page. Try the following approach.

The way the script finds the username, password and submit inputs is with the following wordlists within the script.

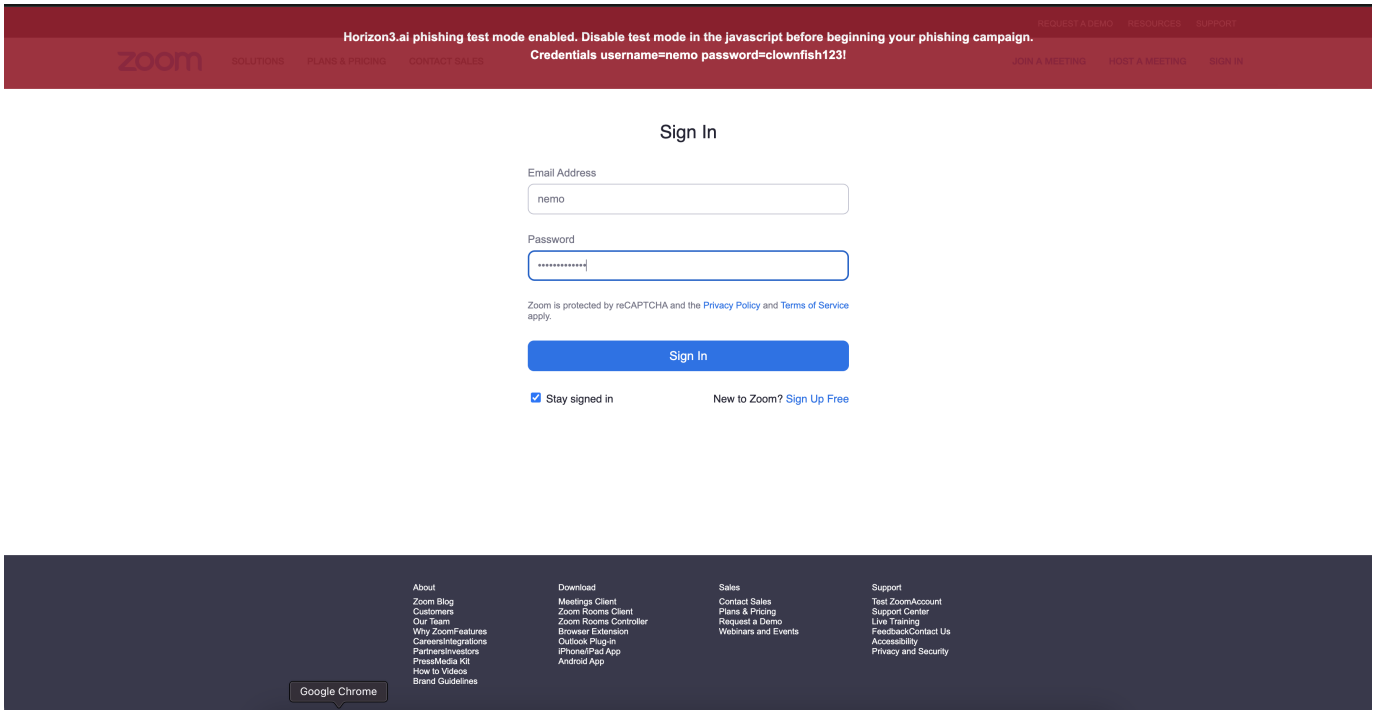
```
const lowercaseUsernameTerms = ['email', 'username', 'uname']
const lowercasePasswordTerms = ['password']
const lowercaseSubmitTerms = ['submit', 'sign in', 'log in']
```

Let's imagine that the username field is not being correctly captured. In order to troubleshoot you can open the developer tools in your browser and inspect the input element. In this case the HTML attributes on the input are `user` and our wordlist for `lowercaseUsernameTerms` does not include `user` we can add this item to the wordlist and refresh the page and test again.

```
const lowercaseUsernameTerms = ['email', 'username', 'uname', 'user']
```



Type into the username field and see if the credential reflects back. If so, you're good to proceed on to the next steps.



THE BANNER DOES NOT TURN GREEN WHEN SUBMITTING CREDENTIALS

If you notice the test mode banner does not turn green and indicate success after submitting credentials, there may be an issue with the API Key used by the script. The generated phishing script is specific to a single phishing test and cannot be reused.

Try regenerating and replacing the phishing script in the login page.

I DON'T WANT TO SHOW THE `INCORRECT LOGIN CREDENTIALS` MESSAGE

Open the landing page source code and ensure it contains the following line.

```
const showIncorrectCredentialsError = false
```

If the value is set to `true`, change it to `false`.

If you cannot find the line on the page, regenerate and replace the phishing script into the login page. Then, find the line and change the value from `true` to `false`.

After updating the source code, be sure to press the Refresh button on your browser to reload the page.

3.10 NodeZero Modules

3.10.1 Cyanide

Cyanide is H3's tool to facilitate and correlate man-in-the-middle (MITM) attacks and credential relays.

Cyanide utilizes opportunistic network protocol poisoning techniques and active coercion techniques to solicit a device to connect to NodeZero with authentication material. If MITM relay has been enabled for the pentest, Cyanide will attempt to relay authentication material to vulnerable target services and applications within the scope of the pentest.

Terminology

MITM Credential Relay attacks typically involve two hosts within the target network. H3 uses the terms "Source" and "Target" to differentiate these two hosts and their associated weaknesses and misconfigurations that enable a successful MITM attack.

- Source - The host/user that initiates a connection and authentication session to NodeZero's relay server. Typically, the source authentication material will provide a username and possibly the path to a requested resource (e.g SMB share, SQL database, etc.)
- Source Weakness - the poisoning or coercion weakness that enabled or caused the source host to connect to NodeZero's relay server.
- Target - The host to which NodeZero will relay the authentication material for exploitation.
- Target Weakness - the Weakness on the target host that enables cyanide to relay credentials successfully and gain unauthorized access.

Purpose

During a pentest, NodeZero has to be able to accurately correlate the source of discovered credentials and track where they are being utilized to access network resources. Additionally, NodeZero needs to accurately determine which

weaknesses were utilized in an attack chain. Cyanide's primary purpose is to make these correlations for MITM attacks, and capture authentication material for use or hash-cracking. Cyanide answers the questions:

- Who
- The user, machine or service account the captured authentication material represents.
- What
- The attack method used to cause the source host/user to connect to NodeZero:
- Poisoning (LLMNR, NBT-NS, MDNS)
- Coercion (PetitPotam, ShadowCoerce, PrinterBug, etc.)
- The resource requested by the source user (e.g SMB share, SQL database, etc.)
- Where
- The source host of the authentication material (i.e. Where the connection came from).
- Where were the credentials utilized (i.e. the target service/host of the relay attack).
- When
- The date/timestamp of each event will be available:
- When the host was poisoned
- When the hash was captured
- When a relay attack happened
- Why
- With the combined data, Cyanide can tell why this attack happened and why it was successful.

System Breakdown

Currently the Cyanide system consists of 4 distinct parts:

1. Cyanide Message Pump
   - Responsible for correlating source and target information and providing collected data to NodeZero
2. Responder
   - Responsible for broadcast protocol poisoning
3. Intimidator
   - Responsible for coercion attacks
4. Impacket's ntlmrelayx
   - Responsible for handling inbound SMB and HTTP connections and relaying authentication material to vulnerable targets

THE CYANIDE MESSAGE PUMP

The main cyanide process, or message pump, processes incoming messages from the other 3 components of the Cyanide system and takes appropriate action correlating source and target information and populating a database that NodeZero can utilize to understand what MITM interactions are occurring and what new authentication material is available for the pentest.

RESPONDER

Responder is an LLMNR, NBT-NS, and MDNS poisoner. It will answer specific NBT-NS (NetBIOS Name Service) queries based on their name suffix (see: Archived Microsoft KB). By default, the tool will only answer File Server Service requests, which is for SMB.

If Responder poisons a source host via one of these broadcast protocols, it will reach back via whatever protocol the broadcast was for (e.g. SMB, RDP, MSSQL, etc.).

- If the protocol is NOT SMB or HTTP, Responder will simply capture the credential and inform Cyanide of the results. Cyanide will then report this data back to NodeZero's ephemeral cloud architecture and attempt to re-use or crack the captured credential material.
- If the protocol IS SMB or HTTP, ntlmrelayx SMB and HTTP servers will handle it appropriately - either relaying it to a vulnerable service or dumping the credential for cracking.

- Poisoners
- LLMNR
- LLMNR stands for Link-Local Multicast Name Resolution. LLMNR is based on the DNS format and enables computers on the same local network to conduct name resolution of other hosts. LLMNR is unicast, so only the device that sent the request will see the reply.
- Server port UDP/5355
- NBTNS
- NBT-NS stands for Network Basic Input/Output System Name Service. NBT-NS is often referred to as its base application programming interface, NetBIOS, for short. The NBT-NS protocol is used similarly to LLMNR, except it utilizes hosts on the network by their NetBIOS name and will ask the receiving machine to disclose and return its current set of NetBIOS names. NBT-NS can utilize broadcast, unicast, or multicast.
- UDP/137
- UDP/138
- MDNS
- mDNS stands for Multicast Domain Naming System (mDNS). mDNS replies are sent over multicast so that everyone can see them and keep their local mDNS cache up to date.
- UDP/5353
- Servers
- MSSQL
- TCP/1433
- UDP/1434
- RDP
- TCP/3389
- Kerberos
- TCP/88
- FTP
- TCP/21
- POP
- TCP/110
- SMTP
- TCP/25
- TCP/587
- IMAP
- TCP/143
- HTTPS
- TCP/443
- LDAP
- TCP/389
- UDP/389
- DCERPC
- TCP/135
- WINRM
- TCP/5895

Expanded vs. Limited Poisoning

NodeZero's Attack Configuration options have 2 options that control the behavior of Responder:

- Expanded LLMNR and NetBIOS poisoning
- Responder will sniff all available traffic regardless of scope.
- In "Expanded" mode, relay sources from outside the pentest's configured scope will NOT be targeted for any other attacks; they are only used to capture/relay credential material.
- Limited LLMNR and NetBIOS poisoning
- Responder is limited to the scope provided during the configuration of the pentest. If both options are selected, the Limited option will override the Expanded option.

Where does it work?

Since Responder works by capturing broadcast and multicast packets, capturing requests in different networks is not possible and therefore, Cyanide will only work within NodeZero's subnet.

INTIMIDATOR

Intimidator is H3's framework for integrating NTLM coercion techniques with Cyanide. Intimidator provides a quick plug-and-play capability to facilitate the inclusion of new coercion techniques and open source tools quickly into NodeZero. Cyanide communicates with Intimidator over a duplexed IPC -- allowing the two processes to coordinate coercion and relay attacks effectively.

IMPACKET'S NTLMRELAYX

Cyanide utilize's a modified version of Impacket's NTLMRelayx as the base for our relay server.

When a source host connects and provides authentication material to ntlmrelayx's SMB or HTTP server, it will save the NTLMv2 hash for cracking and relay the authentication session to high-value service vulnerable to NTLM relay within the scope of the pentest. Possible targets include: - SMB servers with SMB-signing disabled: If cyanide is able to successfully log into the server, it will attempt to dump local credentials. - ADCS Server with the ESC8 Misconfiguration - LDAP servers with LDAP Signing disabled.

Scoping Scenarios

The below Scenarios and Examples review cyanide's behavior when the "Limited LLMNR and NetBIOS poisoning option is configured for the pentest.

SCENARIO 1

No scope is specified OR if the scope of the NodeZero host subnet is specified Scope defaults to the full subnet of the NodeZero host to poison

Example 1

```
NodeZero host subnet: 192.168.0.0/24
Scope: Auto-Expand
```

Result: Cyanide will get a scope of 192.168.0.0/24 because no scope was specified and poisoning can only happen within the network of NodeZero. Relaying will occur against high-value targets that are discovered. Coercion attempts will be made against high-value targets within the pentest scope.

Example 2

```
NodeZero host subnet: 192.168.0.0/24
Scope: 172.16.100.0/24, 10.0.0.0/16, 192.168.0.0/24
```

Result: Cyanide will get a scope of 192.168.0.0/24 because the specified scope contains the subnet of the NodeZero host. Relaying and coercion will occur against high-value targets that are within the scope specified.

SCENARIO 2

The scope of the NodeZero host is within the whitelist, Cyanide will get that as its scope

Example

```
NodeZero host subnet: 192.168.0.0/24
Scope: 172.16.100.0/24, 10.0.0.0/16, **192.168.0.0/30**
```

Result: Cyanide will only poison hosts within the 192.168.0.0**/30** subnet because it falls within the NodeZero hosts subnet. Relaying and coercion will occur against high-value targets that are within the scope specified.

3.11 Notifications

3.11.1 Notifications

NodeZero will send email notifications for the following events that occur during the operation lifecycle.

State types:

ℹ️ - informational
⚠️ - action required

Event / State	Type	Description	Sent To
Pentest started	ℹ️	NodeZero has launched and successfully connected back to the one-time-use ephemeral architecture to start the pentest	The user who created the pentest
Pentest completed	ℹ️	The pentest completed and results are available in the Portal	The user who created the pentest
Pentest paused	⚠️	The pentest paused due to lost connectivity with NodeZero	The user who created the pentest
Pentest resumed	ℹ️	The pentest resumed after being paused	The user who created the pentest
Pentest expired	ℹ️	The pentest never launched and was auto-canceled after 12 hours	The user who created the pentest
Paused pentest canceled	ℹ️	The pentest was stuck in a paused state for over a week and was auto-canceled	The user who created the pentest
Pentest waiting for NodeZero	⚠️	The pentest was created, but has not yet launched, because NodeZero has not yet connected back the H3 cloud	The user who created the pentest
Pentest ready to start	⚠️	The pentest launched in a paused state, typically for whitelisting NodeZero's IP, and is waiting to be started by the user	The user who created the pentest
Scheduled action initiated	ℹ️	The scheduled action was successfully initiated at its scheduled time, e.g. creating a scheduled pentest via the h3-cli or Portal	The user who created the scheduled action
Scheduled action failed	⚠️	The scheduled action failed to run	The user who created the scheduled action
NodeZero Runner hit an error	⚠️	The NodeZero Runner failed to launch NodeZero	The user who created the Runner, along with the user who created the pentest that failed to launch

3.12 Media

3.12.1 Media

The only place to get your NodeZero knowledge on. Learn tips and tricks for making efficient use of NodeZero

Videos

Introduction to NodeZero

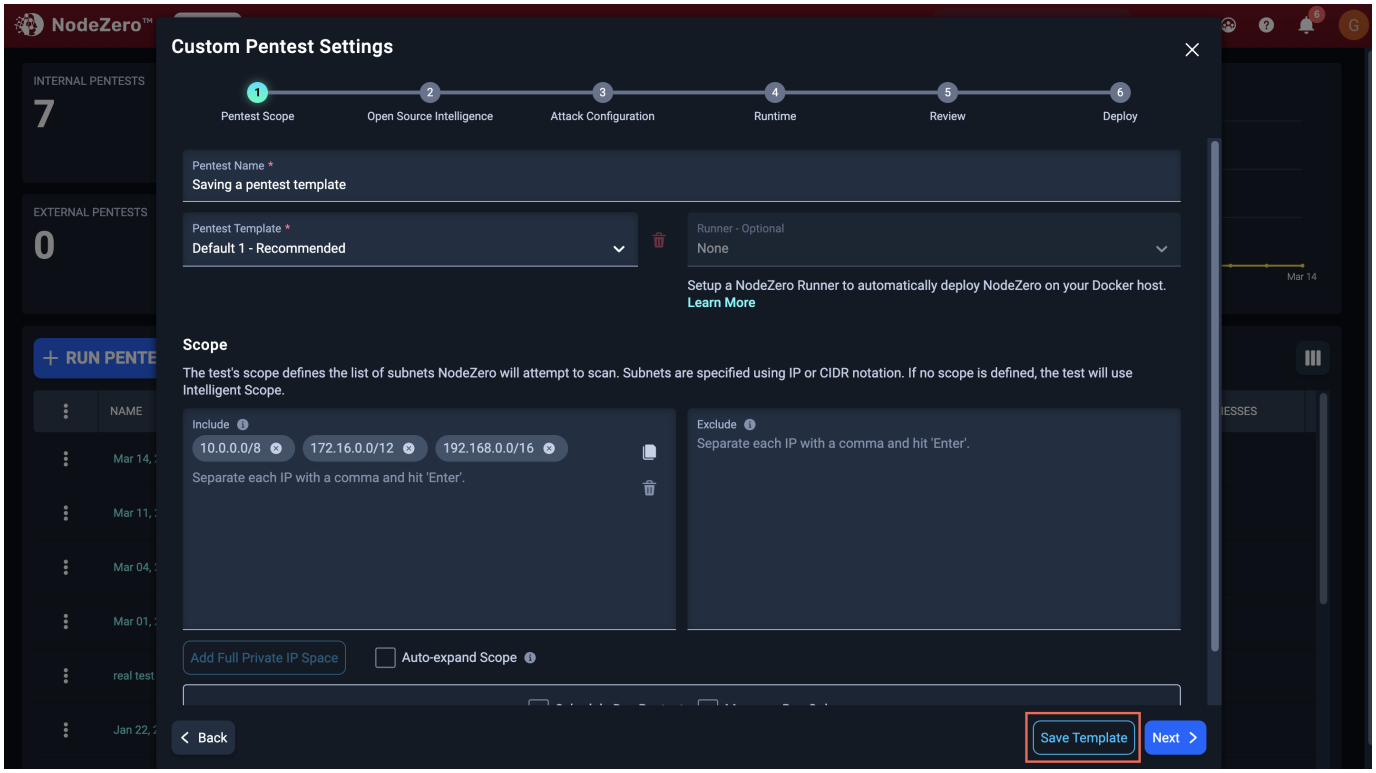
3.13 Templates

3.13.1 Templates

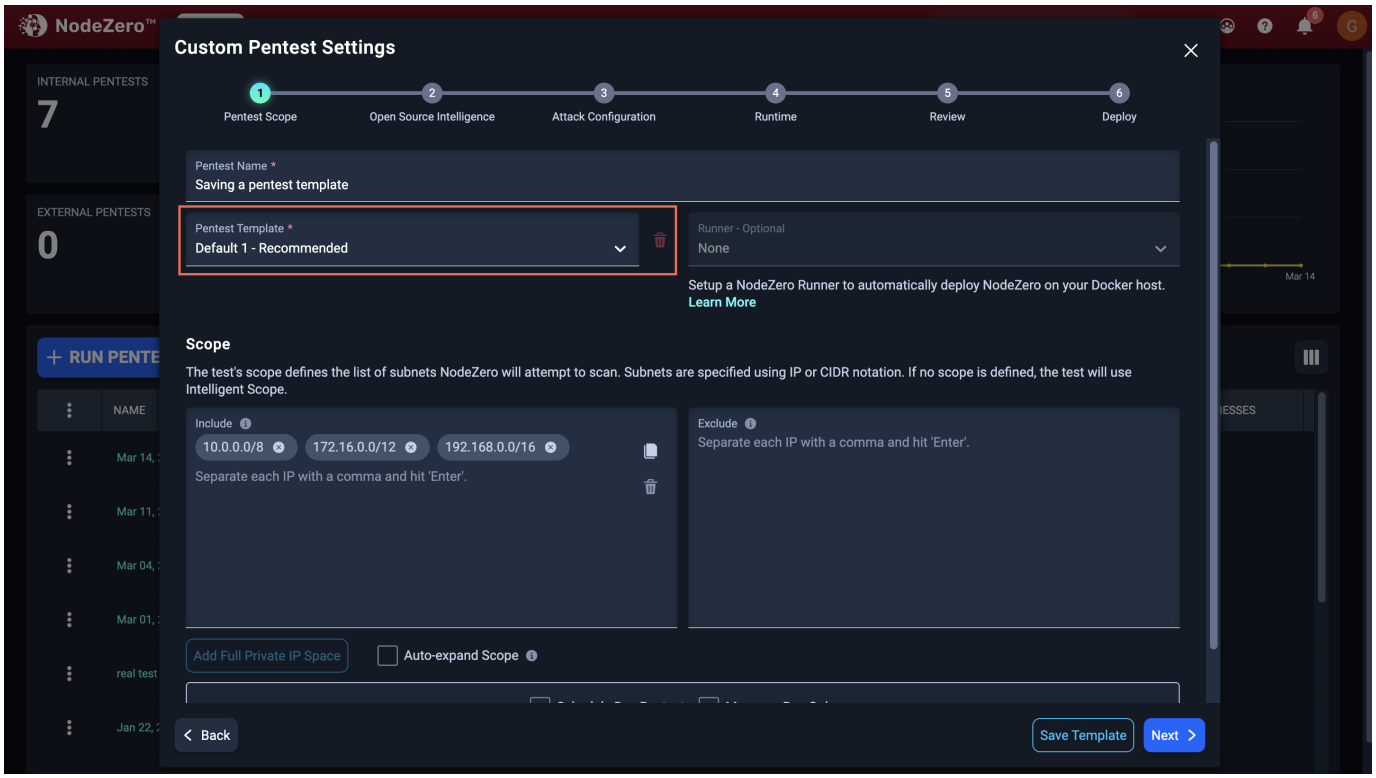
Templates allow you to save a pentest configuration and then use that saved configuration when running a one-off pentest or when scheduling repeated pentest runs. You can save or load templates from within the Run Pentest wizard, or using the template management page in the Horizon3 portal.

Templates in the Run Pentest wizard

When you define a one-off pentest using the Run Pentest wizard, you can click the `Save Template` button at the bottom of the wizard to store your test configuration in a template for future use:



If you want to use a previously-saved template when defining a test in the wizard, use the `Pentest Template` dropdown on the first page of the Run Pentest wizard:

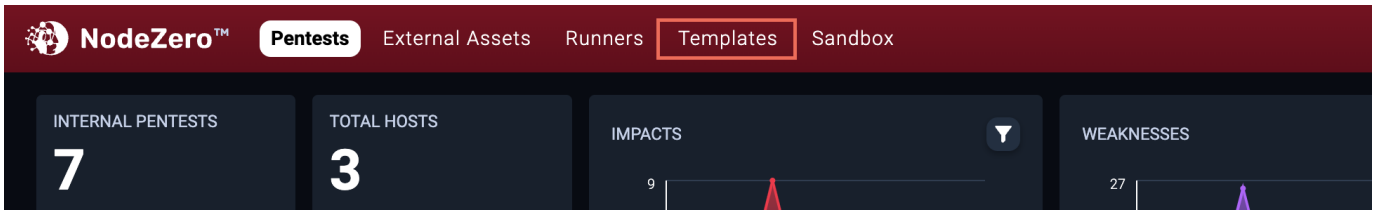


You can also delete a template by clicking the trash icon near the `Pentest Template` dropdown.

Template Management page

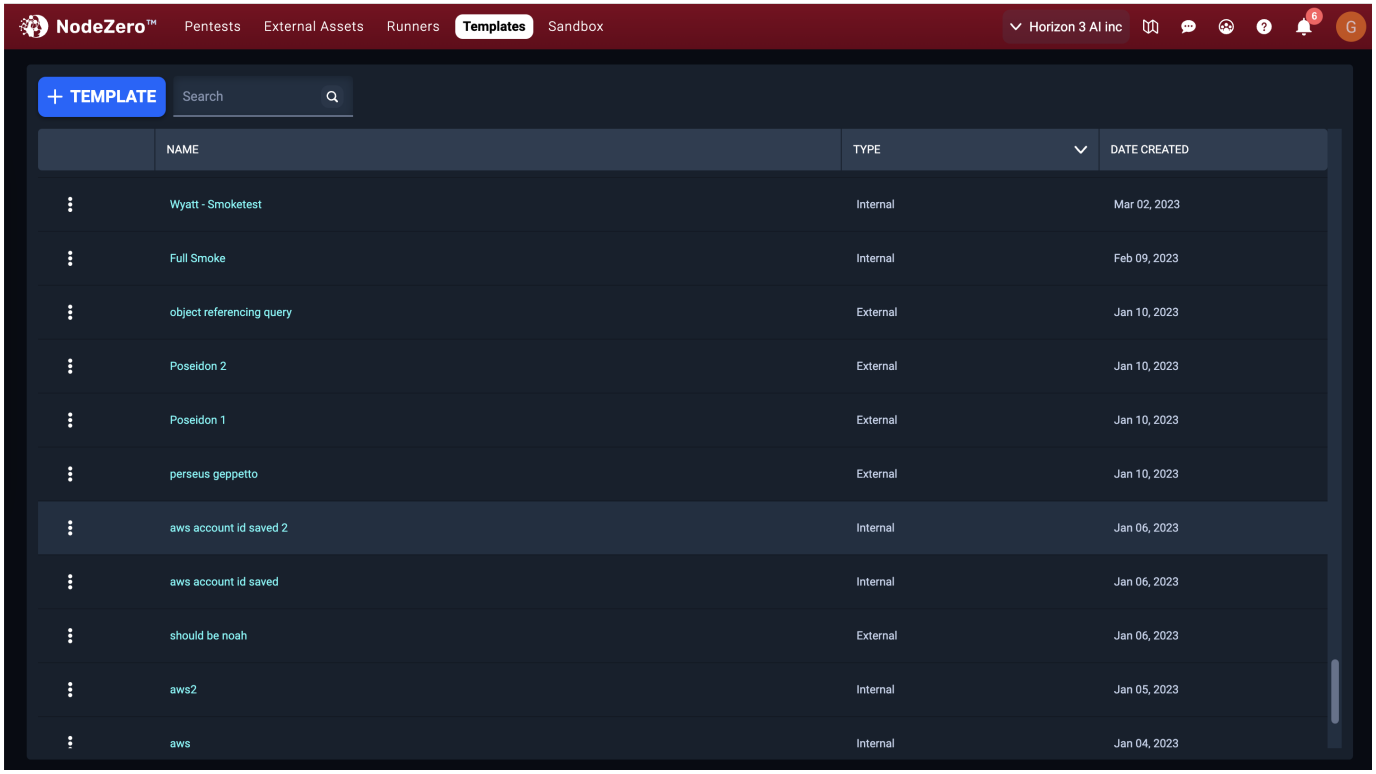
Although you can access or create templates from within the Run Pentest wizard, you may want to see all of your templates and manage them in one place. For this, you can use the template management page. This page lists all of your templates and lets you duplicate or delete them. You can edit templates you saved earlier, and create new ones.

To access the template management page, click the `Templates` link in the navigation area at the top of any Horizon3 portal page:



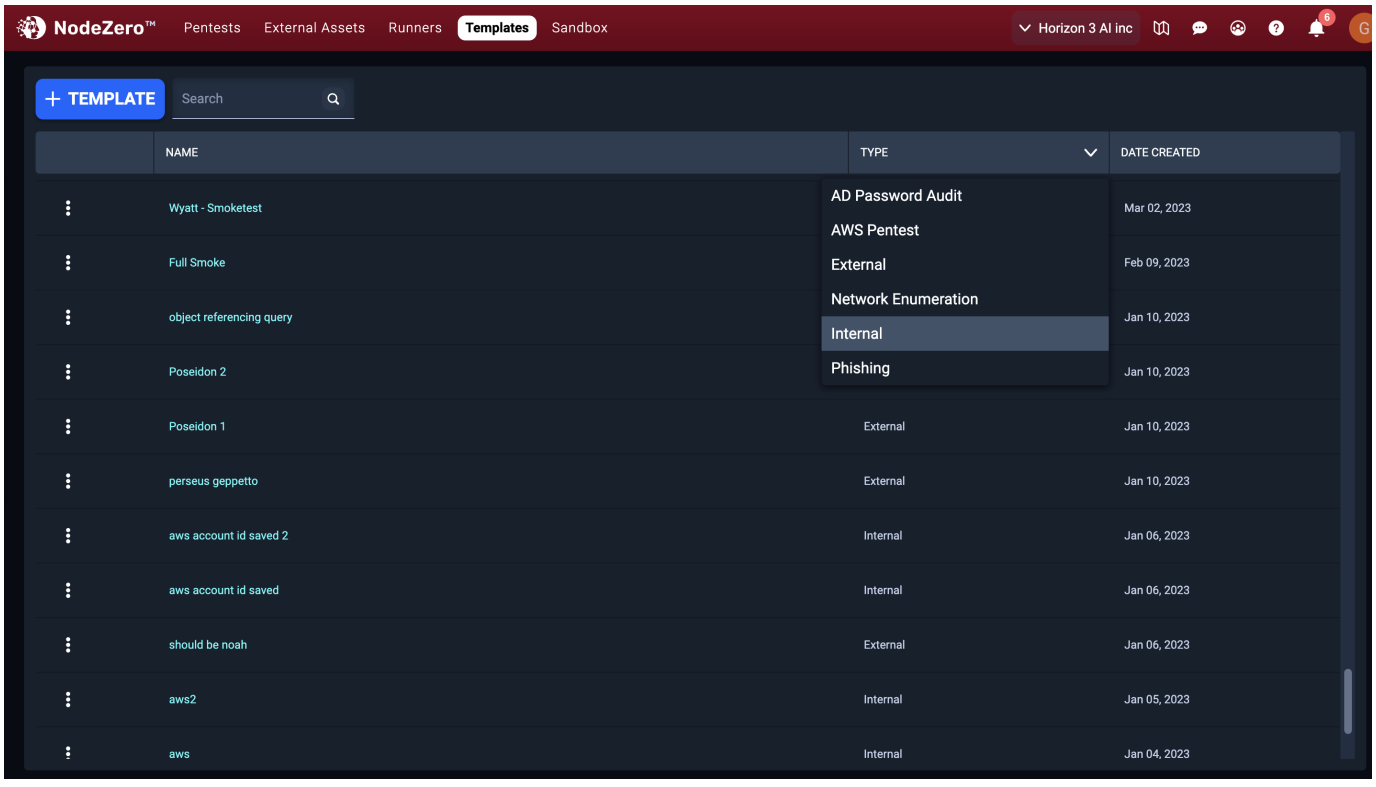
THE TEMPLATE LIST

The template list shows you all of your templates in one place:

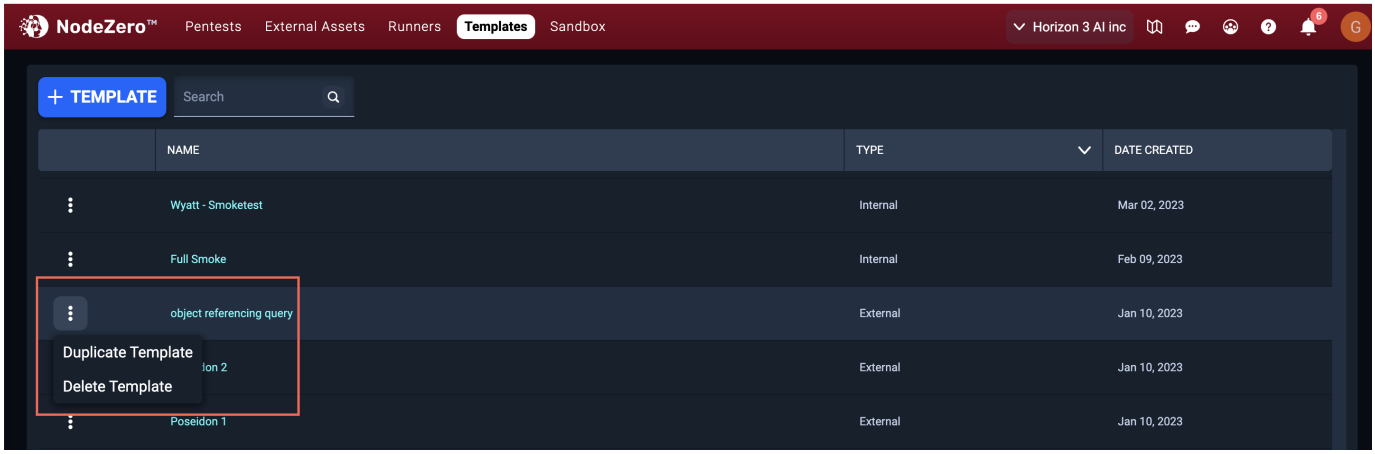


You can click on each column header to adjust the sort order for the table. You can also search template titles by entering some text in the search box at the top of the table.

By default, the template table shows templates belonging to any test type. You can filter this list to only show a specific test type by clicking the filter arrow in the `Type` column:



Within the table, you can duplicate or delete any template row by opening the action menu for that row and selecting the corresponding action:

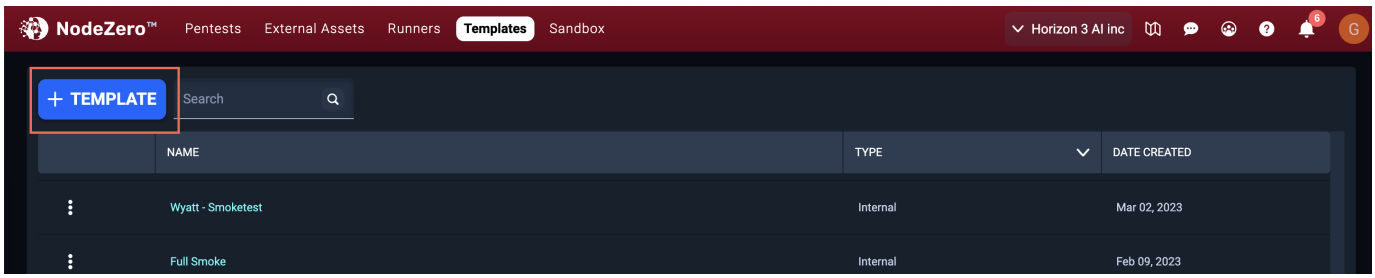


Template names are unique

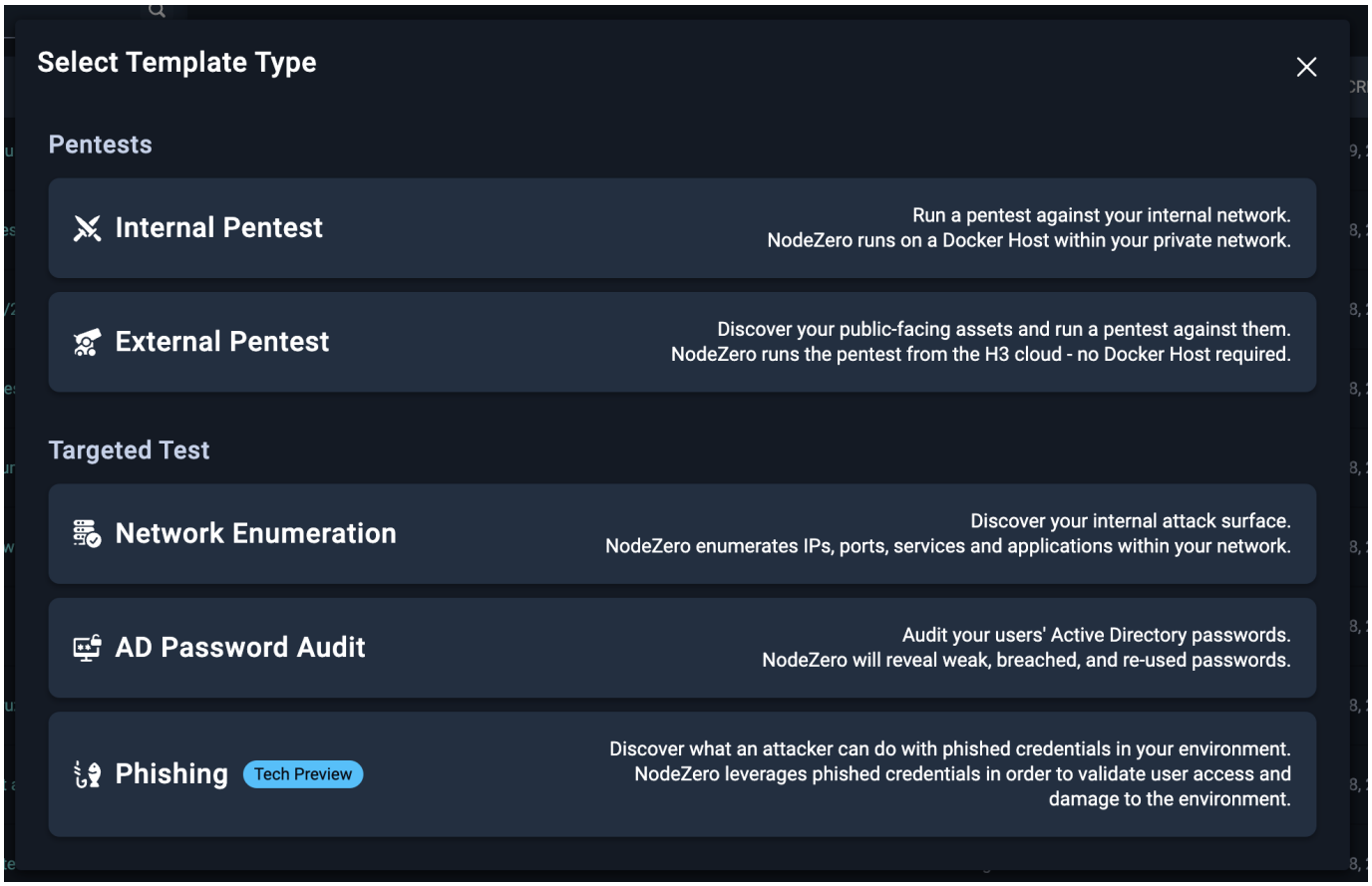
Template names must be unique within your organization. If you attempt to create or duplicate a template with a name that is already used by an existing template, you'll receive a warning. You can choose to overwrite the original template, or cancel.

CREATING TEMPLATES

You can create a new template from the template management page by clicking on the `+ Template` button near the top:



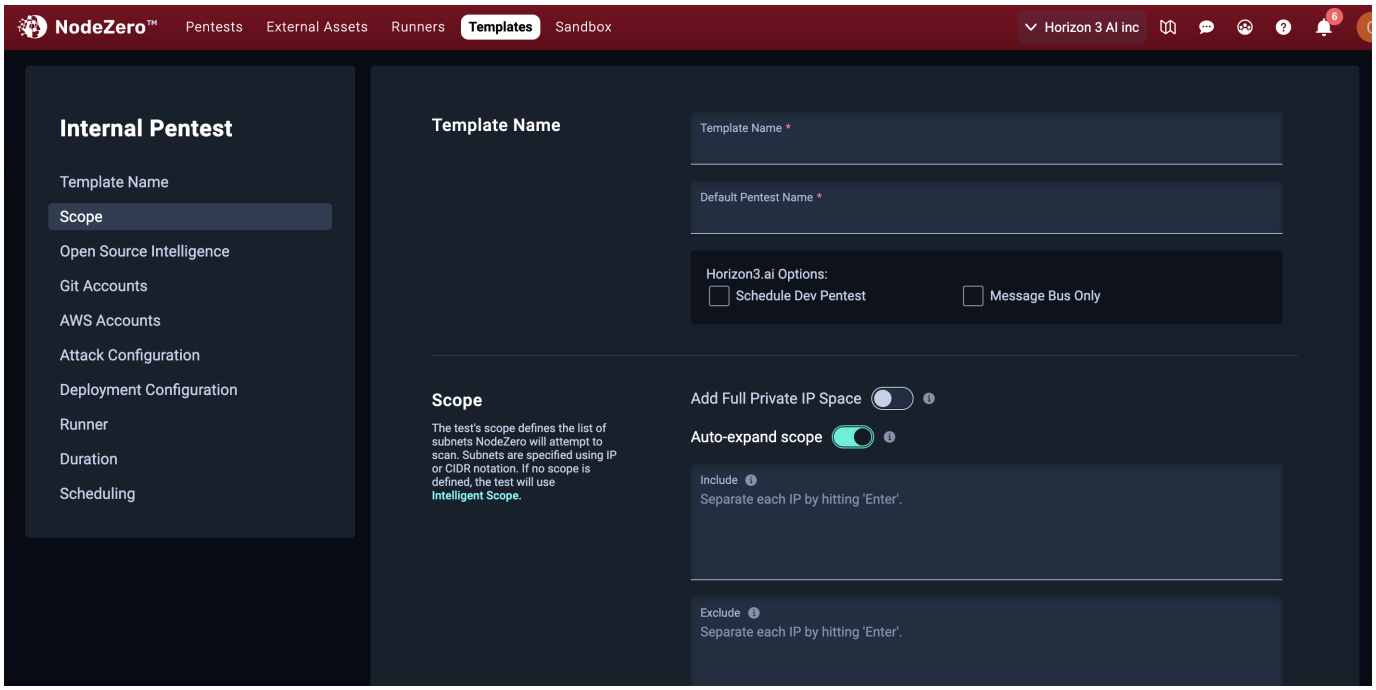
This will open the template type selection dialog, where you can select which type of template you want to create:



After selecting a type, you will see a form where you can input all of the configuration for this template. This configuration will be used when starting pentests based on this template. For more about running pentests from templates, see Running pentests using templates.

You can use the navigation links on the left to jump to different parts of the form. Every template must have a name and a default pentest name (which is used as a default name when running pentests from this template).

Depending on the test type you selected for this template, the form will have different fields. For example, here's the form for an Internal template:



Fill out the form as needed for your testing requirements, then click `Create` at the bottom of the form.

If there were any issues with the form values, you'll see validation errors on the form. Otherwise, you will be taken back to the template list. Your new template will be ready to use from the pentest wizard.

> **Template names are unique**
>
> Template names must be unique within your organization. If you attempt to create or duplicate a template with a name that is already used by an existing template, you'll receive a warning. You can choose to overwrite the original template, or go back and change the name.

For more information about the actual test configuration parameters and what they do, see the test documentation.

EDITING TEMPLATES

You can edit the configuration for an existing template. To do this, find the template you want to edit in the list and click on it. This will take you to the edit page for that template, which is similar to the page for creating templates. The main difference is that the template's type and name cannot be changed after they were first saved.

Make any required changes to this template and then click `Update` at the bottom of the page. You'll be returned to the template list once this has completed.

Running pentests using templates

The purpose of having a template is so that you can easily start pentests using a template's predefined configuration. There are two ways to do this: manually, or on a schedule.

To manually start a pentest using a template, use the Run Pentest wizard, as described earlier in this document.

You can also run pentests regularly by setting up a schedule. For more information, see this page.

3.14 Release Notes

3.14.1 Release Notes

Our Release Notes provide a consolidated monthly summary of all the enhancements and updates delivered. These notes capture the culmination of continuous improvements we've made throughout the month, offering you a cohesive overview.

For real-time updates and the latest release information, please check the notifications directly within the portal.

2024.04

April Updates ☔ April showers bring not just flowers, but also a deluge of enhancements to fortify your digital defenses! This month, we're rolling out new features designed to proactively empower your organization against the evolving threat landscape.

- **Rapid Response Alert Center**: Stay ahead of cyber threats with our new Rapid Response Alert Center. It provides early alerts and actionable intelligence, allowing you to proactively address vulnerabilities before they're exploited widely. This strategic addition is designed for real-time defense adjustments, ensuring you're always prepared.
- **New Attack Content**: Enhance your defense with our latest attack modules targeting:
- Progress Kemp LoadMaster for remote command execution.
- Entra Seamless SSO to forge Entra ID credentials.
- Azure Instance Metadata Service for querying sensitive data.
- AWS RDS databases to check and exploit default database credentials.
- GlobalProtect for critical infrastructure protection.
- MLFlow for targeting specific machine learning workflows.
- And more!
- **Auto-injected Azure Credentials**: Boost your Azure operations with auto-injected credentials now available for NodeZero scheduled pentests, enhancing both efficiency and security posture.

Check out the 2024.04 Details Page for all the details.

2024.03

March Updates ☘ March marches in with the promise of new growth and our commitment to continuous innovation. This month, we've cultivated a crop of robust features and enhancements aimed at strengthening your security landscape. Spring into action with our latest update highlights:

- **Rapid Response Tests**: Spring into action with our new Rapid Response tests, tailored to help you surgically test and verify the most critical and emerging vulnerabilities within your environment.
- **New Attack Content**: New content like the Fortinet FortiClient EMS SQL injection vulnerability that leads to remote code execution, and other high-profile CVEs.
- **Template Management Page**: Organize and streamline your attack templates with our newly designed template management UI.
- **Active Directory Password Audit**: Enhanced performance, now dumping NTDS secrets 10 times faster.

Check out the 2024.03 Details Page for all the details.

2024.02

February Updates ❤️ Love is in the air, and so is the promise of enhancing cybersecurity with our February updates. This month, we're delivering a bouquet of new features and improvements, all designed to sweeten your security strategy. From user interface enhancements to testing your defenses from the attacker's perspective, let our latest offerings be your Valentine's gift from NodeZero.

- **Sticky Table Headers**: Navigate large data tables with ease, thanks to sticky headers that stay in view as you scroll.
- **Dashboard Views**: Customize your dashboard experience with new vertical and horizontal layout options, ensuring the most critical information is always where you need it.
- **Cookie Consent for GDPR**: Enhance user privacy with our updated cookie consent feature, now in compliance with GDPR.
- **New Attack Content**: Stay ahead of attackers with new attack modules, including CVEs targeting Ivanti Connect Secure, GitLab, ConnectWise SecureConnect, and more.
- **Azure and AWS Enhancements**: Gain deeper insights and control in cloud environments with our latest Azure user creation and AWS metadata service credential harvesting capabilities.

Check out the 2024.02 Details Page for all the details.

2024.01

January Updates ❄️ As the new year begins amidst the quiet chill of winter, our team has ignited a beacon of innovation to heat up your security strategy. This January, we're rolling out powerful updates aimed at enhancing your cybersecurity posture. Embrace new beginnings and make a resolution to harden your infrastructure with our latest developments from the attacker's perspective!

- **Phishing Impact Test**: A significant leap forward with the launch of the Phishing Impact Test in NodeZero, enabling organizations to measure the potential impact of phishing with precision. See our phishing page here
- **Attack Path Enhancements**: Major improvements to Attack Paths, introducing a Vertical Display and Concise/Detailed views, for a clearer narrative on your security landscape.
- **External Asset Discovery Updates**: A series of updates to enhance the identification and management of external assets.
- **New Attack Content**: Expanding our arsenal with critical vulnerabilities targeting Ivanti Connect Secure VPN, Fortra GoAnywhere MFT, Apache OFBiz, Jenkins, and more, alongside inclusion of 21 vulnerabilities from the CISA KEV list.

Check out the 2024.01 Details Page for all the details.

2023.11

November Updates 🦃 As November ushers in the crispness of late autumn, our team has been busy harvesting a rich array of updates and enhancements. This month, we present a bountiful selection of new features and refinements, each designed to fortify and streamline your security landscape. Step into November's technological cornucopia and explore what we've cultivated for you!

- **New Attack Content**: Attack content targeting Cisco IOS XE, Citrix NetScaler, Apache ActiveMQ, and Confluence.
- **Cyanide Activity Identification**: Enhanced NodeZero activity identification in logs with updated Cyanide, now including a static suffix 'H3N0' for simplified tracking.
- **Azure Attack Flow Improvements**: Strengthened Azure integration with the ability to use Azure Refresh and Access Tokens, streamlining the authentication process.
- **Advanced Pentest Management**: Introducing new functionalities like moving pentests between accounts and downloading key pentest data such as External IPs and AD Password Audit results in CSV format.

Check out the 2023.11 Details Page for all the details.

2023.10

October Updates 🎃 As the nights grow longer and Halloween shadows creep in, we've conjured up a spellbinding set of updates for you this month. Like a cauldron brimming with potions, our platform brews with enhancements to bewitch and bolster your security endeavors. Dive in, if you dare!

- **Credential Injection with Node Zero Runners**: Node Zero Runners now support automatic credential injection for scheduled operations, requiring zero manual input post-setup. Especially useful for monthly Active Directory Password Audits, ensuring process adherence and catching overlooked policy errors.
- **Revamped Fix Actions Report**: The newly refreshed fix action report offers an intuitive table of contents and detailed insights, pinpointing affected hosts for each identified weakness. It's a consolidated resource for action-based insights.
- **Enhanced Exposure Score Visibility**: The pentest summary now displays an Overall Exposure Score, derived from a meticulous assessment of critical impacts, weaknesses, and data exposure. Improve your security by addressing these highlighted vulnerabilities.

Check out the 2023.10 Details Page for all the details.

2023.09

September Updates 🍂 As the leaves turn golden and begin their descent, we're thrilled to unveil a flurry of fresh features this autumn. Just as trees are shedding layers, we added layers of innovation in September!

- **Expanded Attack Content**: New content for Citrix devices, Azure VM access, Adobe Coldfusion, advanced password spray, and more!
- **Phishing Integration**: Dive into NodeZero's brand-new test type and seamlessly integrate it with your Phishing campaigns.
- **NodeZero Runner Resilience**: Use `h3-cli` for effortless registration of your NodeZero Runner as a system service.
- **Enhanced Data Discovery**: See "Protected Data" results during pentests for more insightful findings.
- **Revamped Executive Summary**: Discover our refreshed, intuitive design

Check out the 2023.09 Details Page for all the details.

2023.08

August Updates As the summer sun continues to shine bright, so do our platform enhancements! We've brought in a fresh wave of updates this month, aiming to make your experience more seamless and engaging.

- **Enhanced Proxy Support:** Easier and more streamlined proxy configurations.
- **Expanded Coercion Methods:** New methods added to exploit PetitPotam vulnerabilities.
- **Improved Single Sign-On (SSO) Experience:** Open beta for paid accounts.
- **Portal UI Updates:** Introducing new color themes ("Modern" and "Light") and redesigned navigation bar for enhanced user interaction.
- **Phishing Impact Test (Beta):** Introducing new Phishing Impact Test to measure the impact of phishing attacks.
- **Feature Additions:** Added attack content for Juniper, cPanel, H2 Database, Adobe ColdFusion, and Metabase.

Check out the 2023.08 Details Page for all the details.

2023.07

The only thing hotter than July is all the new features. Here are some highlights:

- **New/Updated Vulnerability Detections**: Added several new detections and exploits for weaknesses.
- **Password Spraying**: Improved dynamic generation of weak passwords.
- **External Host Discovery**: Expanded NodeZero's coverage and accuracy for identifying hosts during external enumeration.
- **Domain Controller Identification**: Added better domain controller identification in adverse networks.

Check out the 2023.07 Details Page for all the details.

2023.06

Summer is here, along with a release packed with great new features! Here are a few highlights:

- **Single Sign-On (SSO) Integration**: Added support for Single Sign-On using OpenID Connect (OIDC).
- **Password Audit Operations**: Easily audit the strength and similarity of user passwords in your Active Directory environment.
- **Remote Access Tool (RAT)**: NodeZero can now leverage detected weaknesses and vulnerabilities to deploy Remote Access Tools (RATs).
- **Bulk Authorize External Assets**: The External Assets page has improved ability to sort, filter, and bulk-authorize assets.
- **NodeZero Runner Management**: New Runner Management page improves visibility and control over your Runners.
- **BloodHound Data Collection**: NodeZero now collects BloodHound data during Pentest operations, which can be downloaded post-op.

Check out the 2023.06 Details Page for all the details.

2023.05

After the abundance of amazing attacks in April, we're thrilled to share even more May blooms with you! Here are some of the key highlights:

- **Pentest Scheduling in the Portal**: Say goodbye to manual configurations! You can now easily schedule future pentests and series of pentests directly in the Portal, streamlining your workflows. See the scheduling page for more information.
- **VirtualHost Support for Kubernetes**: NodeZero now supports VirtualHosts in Kubernetes modules, providing enhanced testing capabilities for containerized environments.
- **Real-Time View Enhancements**: Gain deeper insights with Real-Time View updates for External Pentests, including status updates for injected credentials. Stay on top of the progress with real-time information.
- **Portal Login Enhancements**: Experience enhanced authentication capabilities with a new Social Sign-In button for Microsoft/Azure.
- **Attack Content Updates**: As always, we're continually keeping NodeZero up-to-date with important exploits and attack techniques.

Check out the 2023.05 Details Page for all the details.

2023.04

This release packs in some great new features, including NodeZero Runners and the Network Enumeration operation type. A few highlights for this release include:

- Introduction of `NodeZero Runners`, which enable automated deployment of NodeZero without needed to copy-paste the curl script.
- The `Network Enumeration` operation, the first of several Targeted Tests, which enables you to discover the attack surface of your internal network without identifying or exploiting vulnerabilities.
- Added ability for users to inject credentials immediately after scheduling a pentest and while a pentest is paused
- Added new visualizations and filtering to the Hosts Page
- Enhanced password spray and password cracking routines to utilize usernames from breach data
- As always, we're continually keeping NodeZero up-to-date with important exploits and attack techniques.

Check out the 2023.04 Details Page for all the details.

2023.03

Spring is in the air, and with it comes the latest updates to NodeZero! March brings a fresh breeze of features and improvements to help your cybersecurity program bloom. Check out the highlights below, or view in detail on the March 2023 page.

- New user experience enabled by default!
- The new user experience is now enabled by default. Customers that had access to the old experience can still switch back for a limited time.
- As always, we're continually keeping NodeZero up-to-date with important exploits and attack techniques.

Check out the 2023.03 Details Page for all the details.

2023.02

This release is filled with amazing new capabilities and we are excited for you to use them

View the 2023.02 Details Page for detailed explanations, but here are some highlights:

- New User Experience: This is a whole new look and feel to the portal. We've revamped the executive summary and made it much easier to navigate through the results of your pentest
- Externally pentest IP Addresses: Available in the new user experience, you can now add IPs to the scope for an External pentest
- Pause and Resume pentest operations: Available in the new user experience, you can now pause and resume ops from the portal
- 1-Click-Verify multiple weaknesses at a time: Available in the new user experience
- H3 CLI: You can now schedule a pentest to run automatically on a recurring basis using the H3 CLI tool

And much more. Check out the 2023.02 Details Page for all the details.

2023.01

Happy New Year!

This month's release improves functionality in user interface and additional attack content including:

- VMware vRealize Log Insight VMSA-2023-0001
- Active Directory Certificate Services (ADCS) ESC8
- Additional Cloud Attack Content
- Multiple CISA KEVs

View the 2023.01 Details Page for detailed explanations, enhancements, and bugfixes!

3.14.2 2024.04

Features/Enhancements

RAPID RESPONSE

- **Rapid Response Alert Center**: This new feature provides NodeZero platform license holders with early alerts and actionable intelligence, enabling them to counteract emerging cyber threats before they are widely exploited.
- **Rapid Response Tests for External Pentests**: Enhancements now allow users to run N-Day tests from an external perspective, focusing on identifying exposed and vulnerable assets.

AZURE SUPPORT FOR INJECTED CREDENTIALS

- **Injected Credentials for Azure**: Now visible on the Real Time View (RTV) page.
- **Auto-Injected Azure Credentials**: Azure credentials can now be automatically injected into scheduled pentests using a NodeZero runner.

TEMPLATE MANAGEMENT IMPROVEMENTS

- **Schedule UI**: Users can now optionally attach a schedule to any template. For internal-type templates, a runner is required if a schedule is used.
- **Auto-Injected Credentials**: Now available in the Template Tab UI.
- **Documentation**: Comprehensive user documentation for templates is now available on the Templates page.

New Attack Content

- **CVE-2024-1212 Progress Kemp LoadMaster RCE**: This vulnerability allows unauthenticated attackers to execute commands remotely due to an authentication bypass.
- **Entra Seamless SSO (Silver Ticket Attack)**: If NodeZero compromises the `AZUREADSSOACC$` account in Active Directory, it can forge fraudulent credentials to log into cloud resources.
- **Azure Instance Metadata Service (IMDS) Queries**: Enhanced NodeZero RAT capabilities include querying the Azure Instance MetaData Service (IMDS).
- **AWS RDS Database Enumeration**: NodeZero can now enumerate AWS RDS databases, attempt default credentials, and perform database enumeration if valid credentials are found.
- **CVE-2024-3400**: This critical, unauthenticated command injection vulnerability affects GlobalProtect in specific configurations and can be tested via targeted N-Day Test or standard internal/external pentest.
- **Detection for CISA KEV CVE-2024-3273** and support for **CVE-2023-6975** and **CVE-2023-6977** targeting MLFlow version 2.11 and below.

OTHER UPDATES & IMPROVEMENTS

- **Fortinet Server SQLi RCE Exploit**: Extended to work against targets in version 7.2.X.

Fixed Bugs

- Resolved an issue where NodeZero incorrectly labeled IP addresses as belonging to a cloud service.

3.14.3 2024.03

Features/Enhancements

NEW FEATURE

We are thrilled to unveil the **Rapid Response tests**, a pioneering set of capabilities tailor-made to augment Horizon3.ai's Rapid Response Program.

Rapid Response is a dedicated service from Horizon3 that proactively informs organizations of emerging, exploitable vulnerabilities relevant to assets previously scanned by NodeZero within their environments. A central page for Rapid Response alerts is slated for launch in the upcoming months.

Horizon3.ai's Rapid Response tests enable swift assessment and verification of specific, high-impact vulnerabilities within your environment, ensuring they are effectively mitigated. This curated list focuses on urgent, exploitable vulnerabilities demanding immediate attention and action.

Rapid Response tests will be executed through the "Run a Pentest" screen, concentrating solely on the selected vulnerabilities for the test.

Currently, Rapid Response tests are restricted to internal pentests. The expansion to include public-facing assets in these tests is planned for the near future. For now, to test public-facing assets, select "Run an External Pentest" to run a full External Pentest that includes this content.

NEW ATTACK CONTENT

- **Fortinet FortiClient EMS Vulnerability (CVE-2023-48788)**: NodeZero now tests for a recent SQL injection vulnerability leading to remote code execution and full server compromise.
- **JetBrains TeamCity Authentication Bypass (CVE-2024-27198)**: Added to our attack suite.
- **FortiClient EMS Application Fingerprinting**: Improved detection of the FortiClient EMS application.

OTHER UPDATES & IMPROVEMENTS

- **New Template Management Page**: A fresh user interface for managing templates within your organization, facilitating the creation, editing, or deletion of templates.
- **Active Directory Password Audit**: Enhanced capability allowing NodeZero to extract NTDS secrets 10 times faster, along with bolstering the stability of the feature.
- **AWS Attack Capabilities**: NodeZero now enumerates lambda functions for sensitive data, such as AWS keys utilized as environment variables.
- **Settings Page Redesign**: The settings page has undergone a redesign for a more uniform appearance, aligning with other sub-navigated pages.
- **Real-Time View Updates**: Now includes minimum and maximum runtime information for ongoing pentests, provided they have been specified in the pentest configuration.
- **External Assets**: Introduced the functionality to sort by status type.

Fixed Bugs

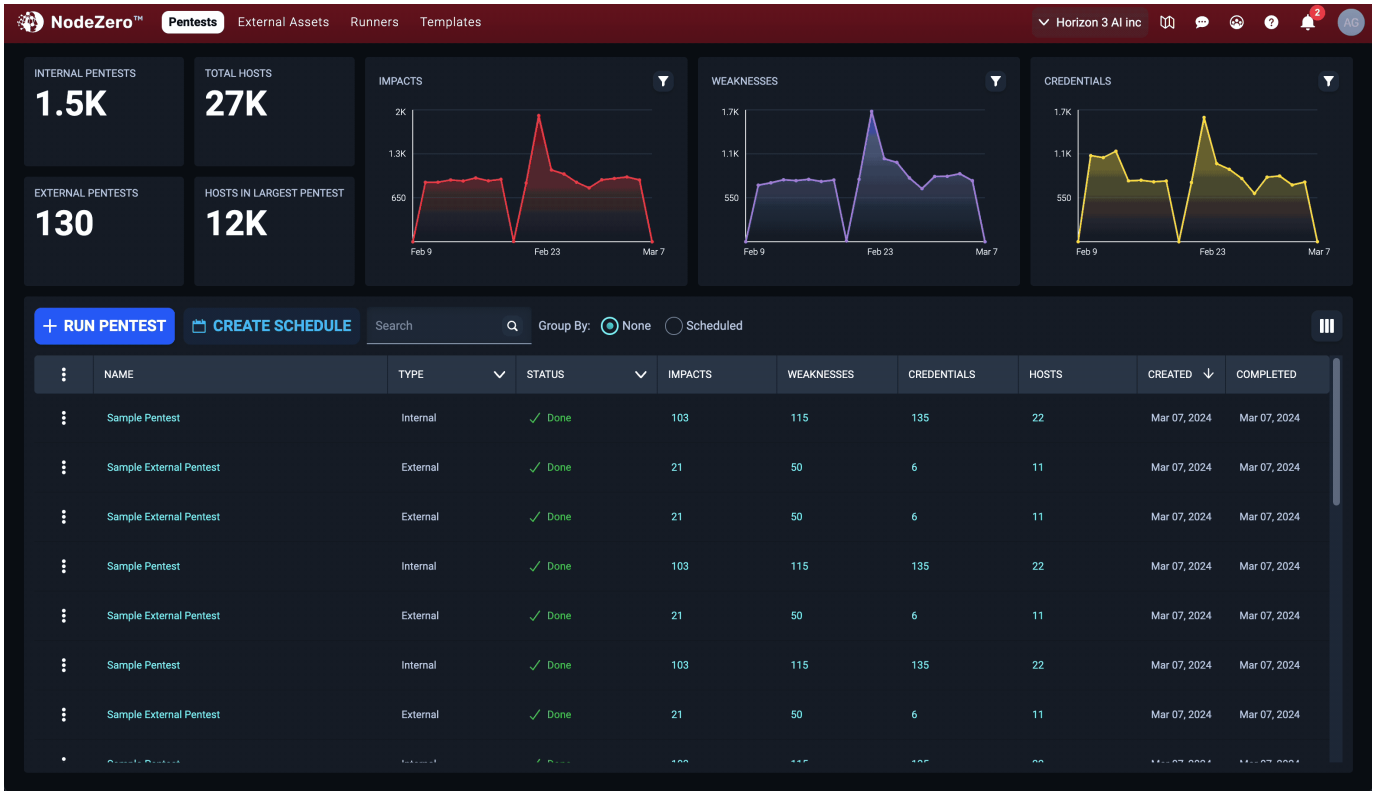
- **Screenshot Functionality**: Resolved issues when taking screenshots related to invalid or weak SSL certificates.

3.14.4 2024.02

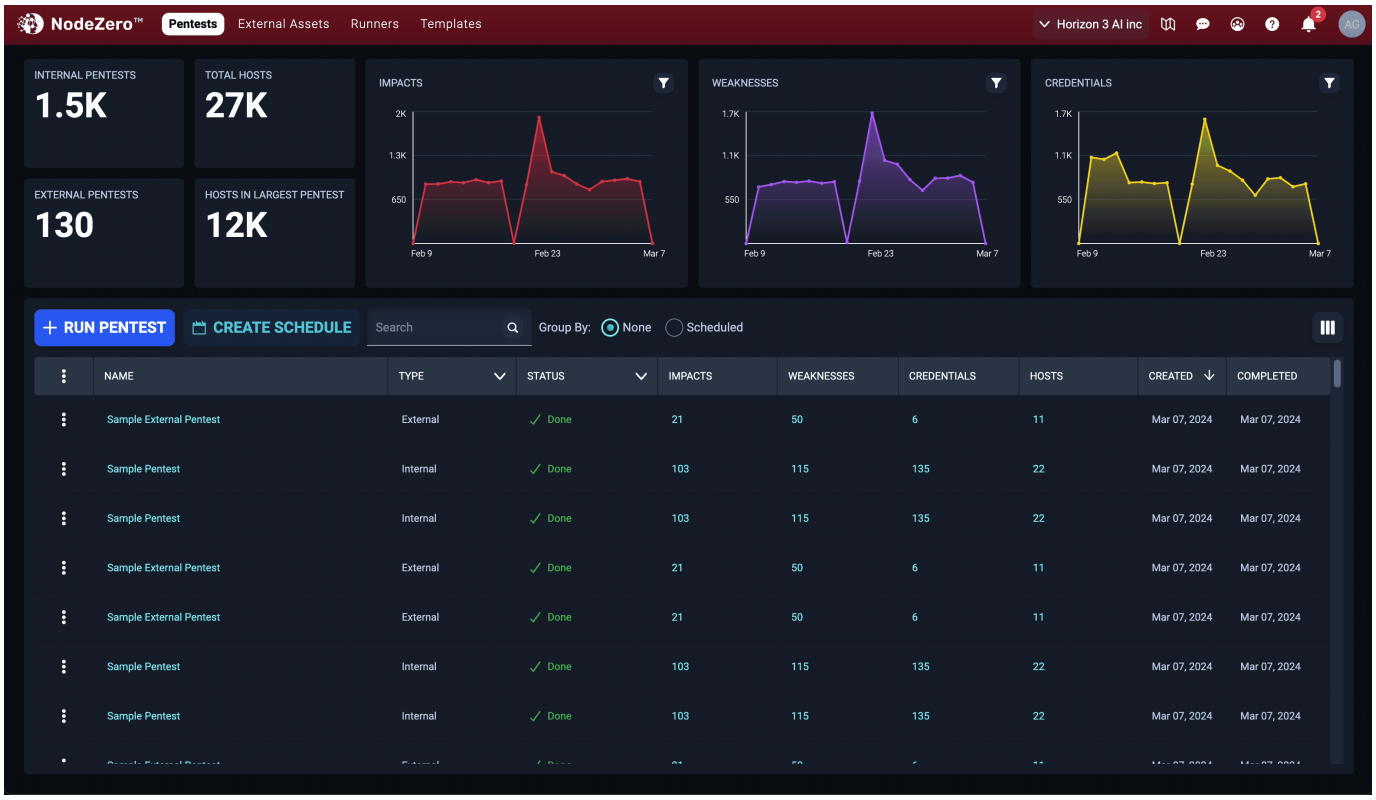
Features/Enhancements

NEW PORTAL FEATURES

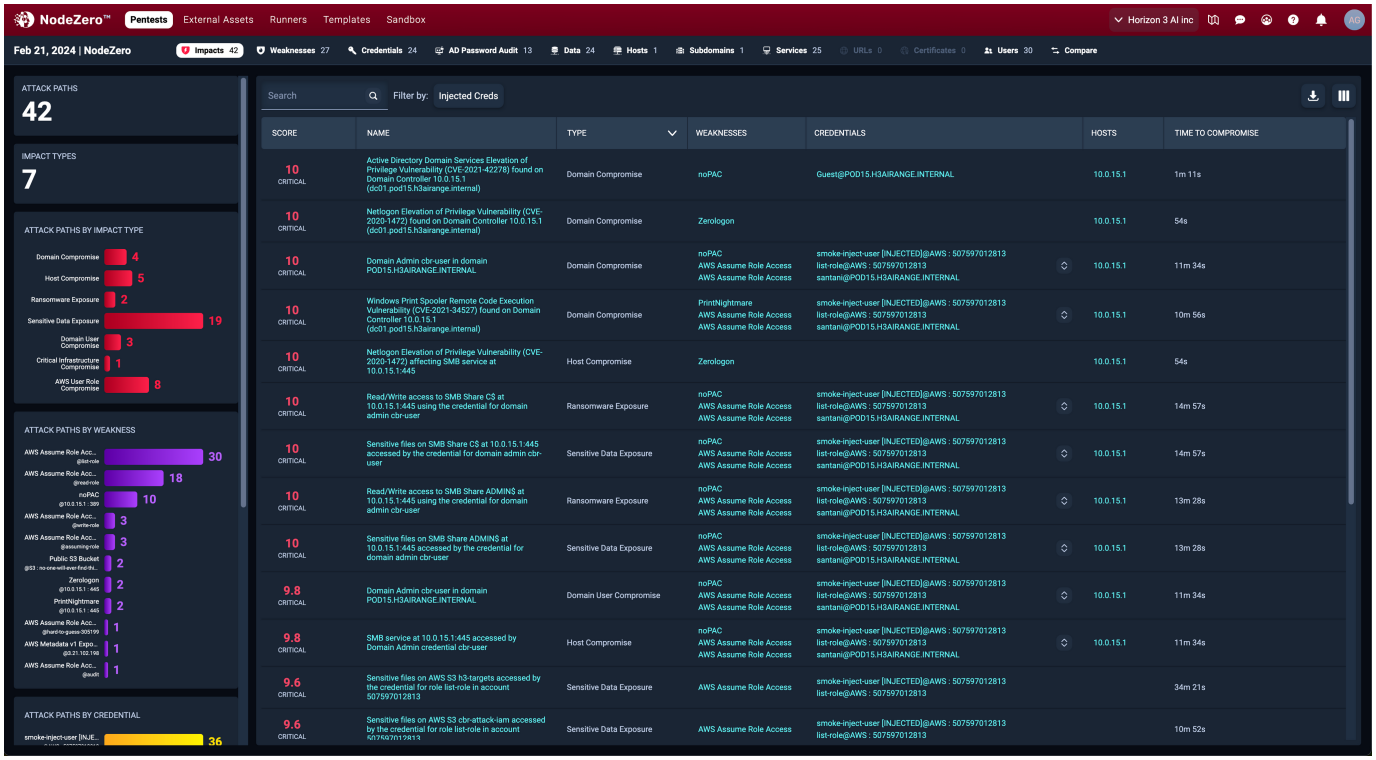
Sticky Table Headers: To enhance data readability, especially in large tables, sticky headers have been implemented across the portal. Now, as users scroll down a table, headers remain visible at the top, eliminating the need to scroll back up to recall column meanings.



Vertical and Horizontal Dashboard: A new toggle under user settings allows for switching the dashboard view between vertical and horizontal layouts:



In the **Vertical View**, dashboards and charts are displayed side-by-side, with charts on the left and tables occupying a broader screen space for easier scrolling:



In vertical mode, all bars in charts are displayed without the need for scrolling. Additionally, the fixed height on the bar container is removed to improve visibility on the side.

Cookie Consent for GDPR: The EU instance of the Portal now features a cookie consent prompt, ensuring GDPR compliance while respecting user privacy. Visitors can accept or manage cookies for a customized browsing experience.

NEW ATTACK CONTENT

- **CVE-2024-21893**: This vulnerability in Ivanti Connect Secure and Ivanti Policy Secure allows server-side request forgery in the SAML component, enabling unauthorized access to restricted resources. Combined with CVE-2024-21887, attackers can fully compromise the system. CISA advises disconnecting and rebuilding affected Ivanti appliances. For further details, refer to our blog post.
- **CVE-2023-7028 GitLab Account Takeover**: A critical flaw in GitLab enabling attackers to reset user passwords and potentially take over accounts.
- **ConnectWise SecureConnect Advisory**: ConnectWise issued an advisory for a critical vulnerability in their SecureConnect software, affecting authentication and path traversal. This vulnerability could lead to administrative control over the ConnectWise server and code execution on connected endpoints. Patching instructions can be found here.
- **Azure and AWS Enhancements**: NodeZero now can create Azure users and elevate them to Global Admins.
- **RAT Enhancements**: The RAT has been extended for broader implantation capabilities, including Linux hosts and harvesting AWS Metadata Service credentials.
- **Attack Path Chaining**: NodeZero also now includes functionality to pilfer through S3 buckets for sensitive information.

OTHER UPDATES & IMPROVEMENTS

- Increased the maximum number of phished credentials in a pentest from 100 to 10,000.

Fixed Bugs

- Resolved an issue in the s3 subdomain takeover process, allowing for direct takeover of subdomains pointed to by DNS records or CloudFront distributions.
- Corrected the ordering of credential insights in the Injected/Phished Credentials Summary.

3.14.5 2024.01

Features/Enhancements

NEW FEATURES

- **Phishing Impact Test** is now live in NodeZero! This feature is crafted to help you gauge and comprehend the impact of successful phishing campaigns within your organization, starting with the employees most susceptible to phishing.
- **Attack Path Enhancements** now include a Vertical Display option, along with Concise/Detailed views. These improvements aim to provide a clearer narrative of your security posture, emphasizing critical impacts and weaknesses.
- **External Asset Discovery** has been updated to assist in identifying the status and warnings for discovered hosts that may not be authorized for pentesting.

NEW ATTACK CONTENT

- **Ivanti Connect Secure VPN**: Authentication Bypass (CVE-2023-46805) and Remote Code Execution (CVE-2024-21887) vulnerabilities have been added.
- **Fortra GoAnywhere MFT Authentication Bypass (CVE-2024-0204)**. For more details, see our blog post.
- **Apache OFBiz Remote Code Execution Vulnerability (CVE-2023-51467)**.
- **Jenkins CLI Vulnerability**: An arbitrary file read through the CLI can lead to RCE (CVE-2024-23897).
- **Confluence Data Center and Server RCE (CVE-2023-22527)**. For additional information, see our blog post.
- Added checks for **21 vulnerabilities from the CISA KEV list**.
- A suite of new **Azure, Azure AD, and MS Entra AD Connect enumerations** enhance NodeZero's capabilities in cloud and hybrid-cloud environments.

UPDATES & IMPROVEMENTS

- **Attack Path Improvements**: New toggle buttons for attack graphs, vertical attack path display, and options for detailed or concise attack path narratives.
- **1-Click Verify Documentation**: Now available to streamline verification processes.
- **New Filters in Tables**: Added "Filter by Injected Creds" & "Filter by Phished Creds" in the Impacts, Weaknesses, and Credentials tables.
- **Summary Page Enhancements**: Now displays injected and phished credentials for a comprehensive view.

FIXED BUGS

- **Zmap Upgrade**: Moved to version 3.0.0 to diminish errors that could disrupt the scope_discovery module in certain operations.
- **Nuclei Template for CVE-2020-10770**: Enhanced to reduce false positives.
- **Azure ADConnect**: Excluded Azure ADConnect AD Service Account from weakness H3-2023-0030 consideration.
- **Azure Refresh Tokens Verification**: Rectified the module responsible for verifying Azure Refresh Tokens.
- **EDR Interference with RAT**: Addressed an issue where EDRs blocking RAT's process list retrieval resulted in Data nodes without resource IDs.
- **Implant RCE Module**: Fixed retry mechanism in some failure scenarios.
- **Payload Echoing by Printers/Servers**: Adjustments made to mitigate false positives based on header checks.
- **AWS boto3 Commands**: Updated commands for creating public S3 buckets.
- **Httpx Scanning on Port 9103**: Resolved an issue causing printers to print gibberish.
- **Attack Path Renderings**: Corrections made for weakness H3-2022-0086.
- **Host Discovery**: Implemented fingerprint-based deduplication to refine host discovery accuracy.

4. Downloads

4.1 Downloads

These pages contain a set of tools and resources that can help you set up and use Horizon3 services.

- NodeZero Host VM (OVA/VHD)
- Host Check Script (checkenv)
- NodeZero App for Splunk (Splunkbase link)
- Horizon3.ai CLI

4.1.1 Validating Checksums

You'll find that tools distributed here come with a `download` and a `checksum` file. The `download` file is the actual file you'll use, and the `checksum` file is a `SHA256` checksum used to verify the first download completed correctly and came from Horizon3.ai.

Given the nature of our industry, we encourage anyone downloading files that claim to come from Horizon3.ai to verify their checksums against the ones posted in this site.

Following are some simple steps to do that.

Linux

Most Linux distributions come with a `sha256sum` command line utility that reads a checksum file, computes the checksum against the file it represents, and returns OK if they match. Just run the command below with both files in the same directory:

```
sha256sum -c some_h3.utility.sha256.checksum
some_h3.utility: OK
```

MacOS

Just like with Linux, MacOS comes with a utility called shasum that works in a similar way. Run the following command with both files in the same directory:

```
shasum -c some_h3.utility.sha256.checksum
some_h3.utility: OK
```

Windows

Windows Powershell has their own utility, but you have to manually compare the output of the following command with the contents of the checksum file yourself:

```
get-filehash -algorithm sha256 some_h3.utility
```

4.2 NodeZero Host Virtual Machine (OVA/VHD)

The NodeZero Host virtual appliance is a small virtual machine based on a pre-configured Ubuntu 20.04 installation. It's designed to execute NodeZero pentests and bundles tools that facilitate pentest execution, as well as debug and maintenance.

4.2.1 Downloads

> **Tip**
>
> Always verify the files download come from Horizon3.

VMWare/Virtualbox importable OVA

Download | SHA256

Windows Hyper-V importable VHD

Download | SHA256

4.2.2 Specifications

The NodeZero host virtual machine comes pre-configured to use these resources:

- 2 x CPUs
- 8GB of RAM
- 40GB of disk
- Bridged network adapter

4.2.3 Before Setting Up

If your environment restricts access to external sites through an outbound proxy or similar mechanism, please make sure that it allows connections to the following sites:

- `*.ubuntu.com`
- `downloads.horizon3ai.com`
- `github.com` (optional for `h3-cli` updates)

4.2.4 Installation

Installing the virtual machine is a matter of importing the OVA file from the download link above into the virtualization environment. We provide the following set of steps as an example to use with VMWare's vSphere client or with VirtualBox.

VMWare vSphere

vSphere client is one of VMWare's virtual environment management solutions. You can find more information on the client itself in VMWare's documentation.

> **Note**
>
> The following steps are for vSphere client version 7.0.3.00500.

After downloading and verifying the most recent NodeZero-####.ova file from the downloads section above, follow these steps to import and launch the NodeZero host virtual machine.

1. Log into your VMWare vSphere client.
2. Select Deploy OVF Template from the Actions menu.
3. Select the Local File option
4. Click the `Upload Files` button to locate the OVA file downloaded in step #1.
5. Give your VM a name if you want it to be different from the default, and select a location to deploy to. Click Next.
6. Select the compute resources you'll be using. Click Next.
7. Verify the import settings are correct and that the signature is from Horizon3. Click Next.
8. Select the storage destination. Click Next.
9. Select a network to use. Click Next.
10. Review your selections. Click Finish.
11. To launch the VM, select it from the list on the left and click the `Power On` button.

VirtualBox

After downloading and verifying the most recent NodeZero-####.ova file from the downloads section above, follow these steps to import and launch the NodeZero host virtual machine.

1. Open VirtualBox.
2. Click on `Tools`, then `Import`.
3. Enter the location of the OVA file. Click Continue.
4. Click Import wait for it to complete.
5. Make sure you use a bridged network adapter:
6. Select the newly imported NodeZero virtual machine from the list on the left.
7. Click `Settings`, then `Network`.
8. Confirm that `Attached to` is set to `Bridged Adapter`.
9. Confirm that `Name` is set to the name of the adapter connected to your internal network.
10. Click OK.
11. Select the NodeZero virtual machine from the list on the left.
12. Launch the VM by clicking `Start`.

Windows Hyper-V

After downloading and verifying the most recent NodeZero-####.vhd file from the downloads section above, follow these steps to import and launch the NodeZero host virtual machine.

1. Ensure Hyper-V has both the Management Tools and Platform enabled and installed. See Windows docs for more info
2. Create a virtual machine. From the Hyper-V manager's menu bar, select `Action > New > Virtual Machine...`
3. In that New Virtual Machine Wizard enter a name for the new virtual machine for use in Hyper-V
4. Select the virtual machine as `Generation 1`
5. Select the amount of memory to use for the virtual machine. Set startup memory to `8192` or greater. Note: Dynamic memory can be used or disabled, per customer's environment standards.
6. Select a network to have NodeZero run in. Ensure the network connection is to a production network where a real IP is given (DHCP) or an assigned IP (Static) can connect. **NOTE: Hyper-V's default switch setting will NOT work**
7. Select the virtual hard disk (vhd). Select `Use an existing virtual hard disk` and browse to the NodeZero-xxx.vhd file that you recently downloaded
8. Review the information is correct
9. From the Hyper-V manager select the newly created virtual machine and click Connect
10. A new window will show the virtual machine and select Start to initialize a new NodeZero host virtual machine.

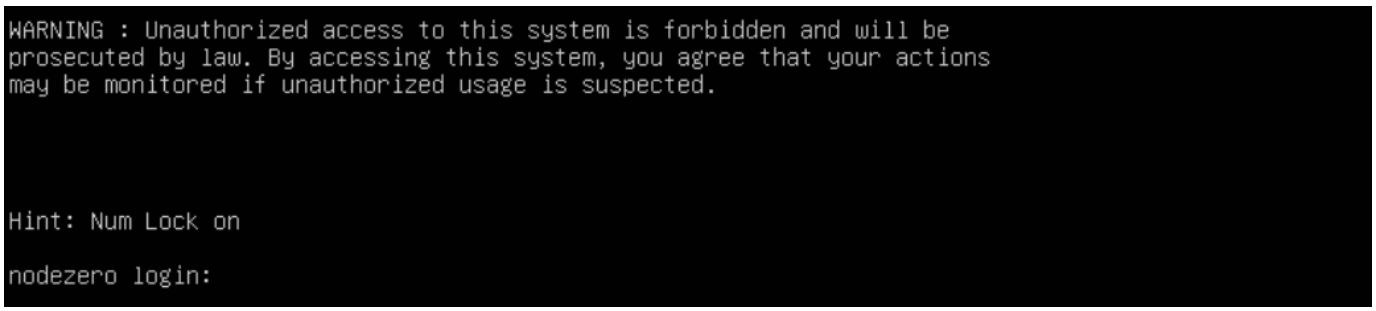
4.2.5 Usage

Connecting

If using vSphere, once the host is powered on, the client interface gives the option of using a web console or a remote console.

If using VirtualBox/Hyper-v, after starting the VM, a new display window appears that shows the operating system load screen.

With either system, once the OS fully loads, there will be a login screen that looks like this:



Username and First Login

When first launching the NodeZero host, SSH password access is disabled until the host is initially logged into and updates the default password.

Login with these credentials:

- Username: `nodezero`
- Password: `nodezero`

When successful, there will be a prompt like the one below:

```
You are required to change your password immediately (administrator enforced)
Changing password for nodezero.

Passwords require min 14 characters and must include at least
1 Digit, 1 Special, 1 Upper and 1 Lower Case
```

```
Current password:
```

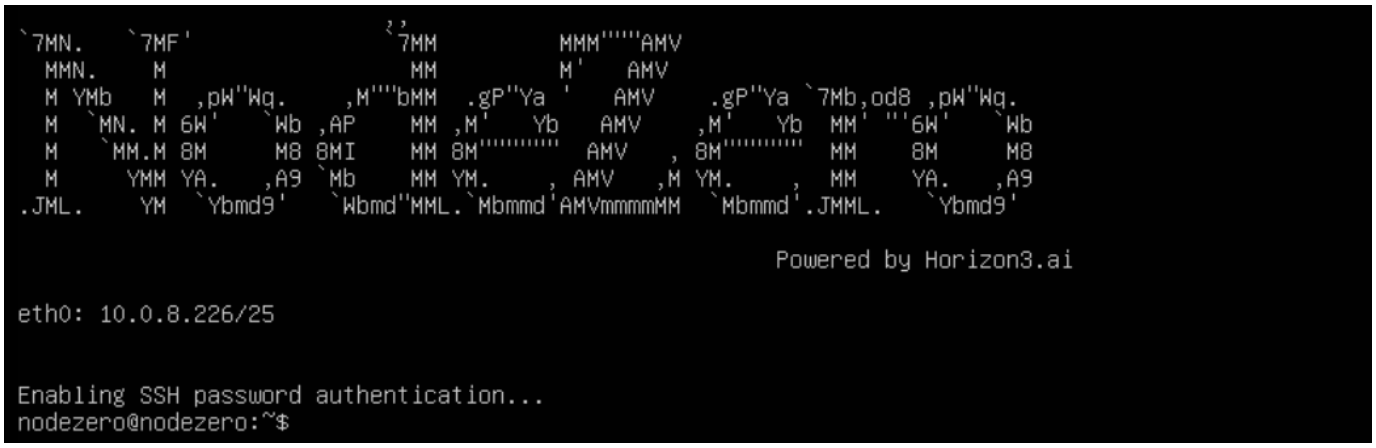
Enter the password from step #1 and hit enter.

Next there will be a prompt for `New password:`, enter a secure password that can be used from now on and hit enter.

Finally confirm the password with `Retype new password:`, enter the same password and hit enter.

Once password has been successfully changed, the user is logged into the host. Make sure to keep that password for use in the future.

Once the login process completes, there will be a message of `Enabling SSH password authentication` displayed. At this point SSH has been enabled on the host and it may be accessed using an SSH client.



Using SSH

To enable SSH on the host, connect to the host over a management console and change the default password for the user `nodezero` as described in the previous step. Once that is done SSH will be enabled on the host.

To connect over SSH with Linux or MacOS, run the command below, replacing `<IP_ADDRESS>` with the one shown in the login screen.

```
ssh nodezero@<IP_ADDRESS>
```

If using Windows, a client like PuTTY will be needed to connect. Fill out the `Host Name (or IP Address)` field with the address shown in the login screen.

n0 utility

The NodeZero host virtual machine comes with a script for setup and maintenancece of the host. To invoke it, type `n0` into the command prompt and a menu like the one below will be presented.

```
n0
```

```
1) Check Environment
2) System Info
3) Configure Static IP
4) Configure Network Proxy
5) Update
6) Set Timezone
7) Version Info
q) Exit

Choose an option: _
```

The following sections provide more information on what these options do.

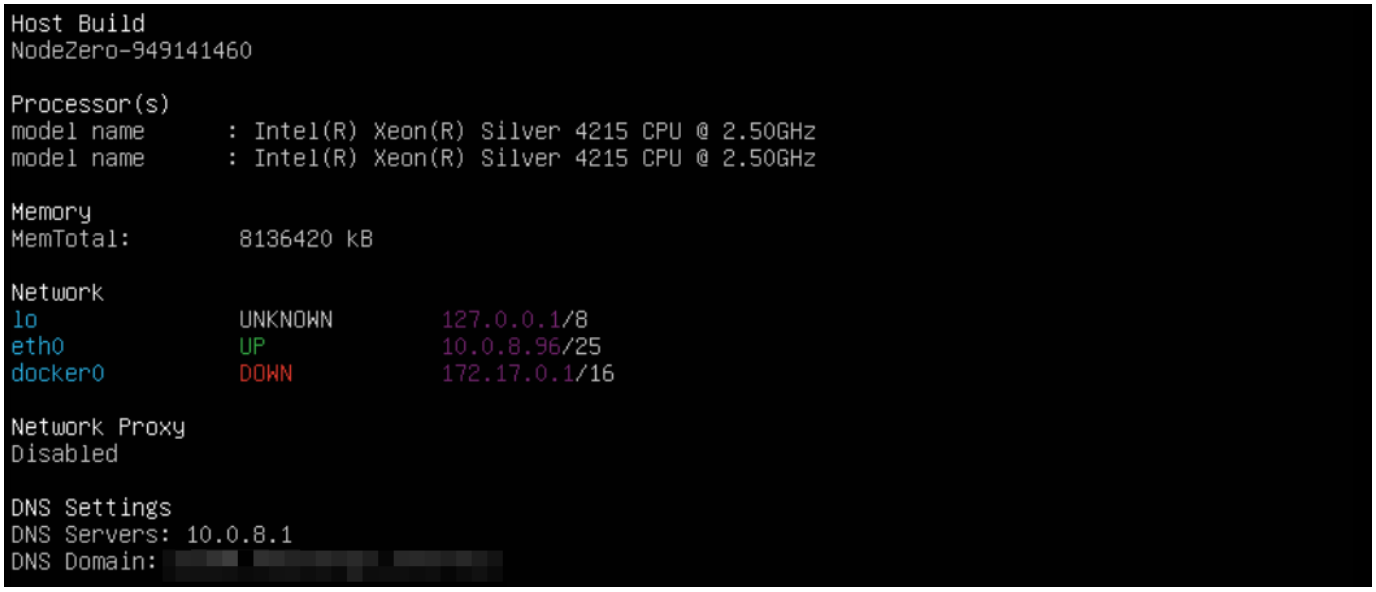
Check Environment

This runs NodeZero's checkenv.sh script to verify the host has the required configuration and settings as well as access externally to the required sites.

System Info

Displays basic system information about the host such as CPUs, Memory and Network settings

Example:



Configure Static IP / Configure DHCP

By default the NodeZero host virtual machine comes with DHCP enabled. The option here will toggle between `Configure Static IP` and `Configure DHCP`.

This defaults to `Configure Static IP` to switch from using DHCP to static ips. Once selected follow the prompts to configure a new IP address, Subnet, Gateway and DNS nameserver

If there is a need to switch back to DHCP, it can be selected with same option number which will show `Configure DHCP`

Configure Network Proxy

To have NodeZero use a proxy for your network provide your proxy setting in the prompt. This command updates values for proxy settings in these files:

/etc/environment

/root/.docker/config.json

/etc/systemd/system/docker.service.d/http-proxy.conf

> **For Changes To Take Effect**
>
> After setting the proxy, the user must logout and back in before the changes takes effect.

Update

This does three things:

1. Updates the underlying OS, mostly for getting the latest security patches
2. Pulls the latest `h3-cli` while preserving the current configuration
3. Updates the `n0` utility itself.

Ensure Latest Updates

It is suggested to run the update option two times back-to-back to ensure the latest updates are retrieved and implemented

Set Timezone

Allows for setting the timezone on the host. Currently supports UTC, GMT, US and EU timezones

Version Info

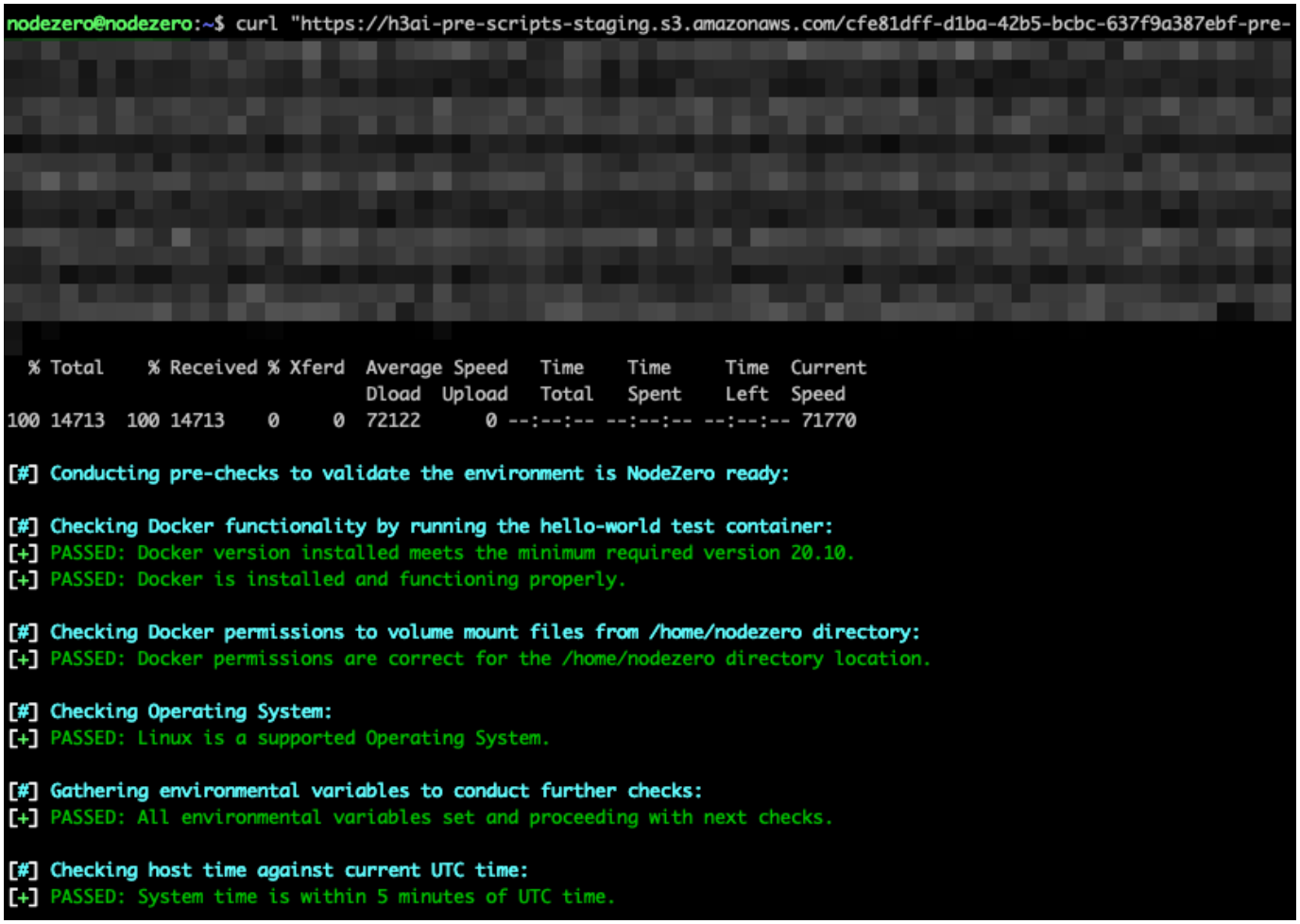
Prints the version of `n0` being used

Exit

Exits the prompt inface of `n0`

Running a NodeZero Pentest

1. Log into the Horizon3 web portal
2. Create a new pentest providing the relevant setup information.
3. Copy and paste the curl command from the portal into the shell of a NodeZero host.
4. The pentest starts executing like the in the screenshot below.



Automating NodeZero

NodeZero supports automating the running of itself through the use of the `h3-cli` and setting up a runner on the host. This can be done by following the steps for setting up a runner

4.3 Host Check Script

This shell script executes inside a Linux environment to validate whether your system is ready to run NodeZero pentests.

> **Note**
>
> This script will NOT launch a pentest.

4.3.1 Downloads

> **Warning**
>
> Always verify the files you download come from Horizon3.

Manual Download (`checkenv.sh`)

Download | SHA256

4.3.2 Download and execute from a shell

```
cd ~
curl https://downloads.horizon3ai.com/utilities/checkenv.sh | bash
```

4.3.3 Troubleshooting

NOEXEC flag on partition

Some users report issues running the health check because they are launching from a partition that denies execution. This is why we recommend first changing to your home directory before executing, as in the example above.

Host's system time is out of sync with NodeZero

If you are seeing this error when running the NodeZero launch script or the `checkenv` script

```
The system time is off by more than 5 minutes of UTC time.
```

To address this you'll need to look at your local time syncing service to make sure it is working properly. Possibly sync your service with a known good source from this list:

- https://tf.nist.gov/tf-cgi/servers.cgi
- https://www.ntppool.org/

If your time syncing service is not something that can be easily repaired, you can still run NodeZero. However, certain cryptographic attacks could be affected. To ensure the successful operation of NodeZero, please deactivate any currently active time synchronization service and synchronize your system time with our servers. The below script will stop `timesyncd`, `ntp` and `chrony`, then sync the system time with NodeZero servers via our `/time` API.

```
# stop timesyncd
sudo timedatectl set-ntp false

# stop ntp
sudo service stop ntp

# stop chrony
sudo service stop chrony

# Sets system time
```

```
UTC=$(curl -k -s -m 3 https://api.horizon3ai.com/v1/time | cut -c17-26)
sudo date -s "@$UTC"
```

NodeZero Runners and `sudo`

Its important to remember that while the Host Check script has the ability to prompt for `sudo` credentials, if running automated operations with the NodeZero runner you might receive a permissions error:

```
[#] Checking Docker functionality by running the hello-world test container:
[+] PASSED: Docker version installed meets the minimum required version 20.10.
[!] FAILED: Failed to validate Docker. Verify this account has permissions to run Docker and retry.
```

If your NodeZero Host requires `sudo` to run `docker` commands, then you may need to start the Runner using `sudo`.

Alternatively, you can try adding the user that invokes `h3 start-runner` to the `docker` group, for example (using `ubuntu` user):

```
sudo usermod -aG docker ubuntu
sudo systemctl restart docker
```

(Note: Make sure to log out and back in after changing groups for the actively logged on user)

4.4 Splunk App for NodeZero

We've built an integration that allows you to ingest NodeZero pentest results directly into Splunk, merging insights gained from the attacker's perspective with existing data and workflows.

You can access the app, along with setup and usage info at Splunkbase

4.4.1 App Contents

The NodeZero App for Splunk comes bundled with the NodeZero Add-on for Splunk built in, which contains the modular input for pulling NodeZero pentests results into the Splunk index of your choosing. There is no need to install both on a standalone instance.

- Sample Dashboard There exists a sample dashboard, `NodeZero Operations`, to demonstrate how one might view summary information across all penetration tests visible in Splunk, then drilldown through specific weaknesses to see exactly what operations NodeZero took against certain hosts during a specific time range.

CIM Compliant

Weakness data has been mapped to the CIM Vulnerability datamodel

4.4.2 Prerequisites

NodeZero API Key

An API key is required to access the API. One can be generated in the Horizon3.ai portal via `Settings > API Keys`. (Note: The Splunk App for NodeZero uses a `ReadOnly` API Key)

4.4.3 Setup

Save API Key to Splunk credential store

On the `Configuration` page, click the `Add` button on the `Accounts` sub-tab

- `Name` is a simple name that gets used by the `Inputs` page
- `Description` is for any notes/details about the API Key/Account
- `API Key` is an encrypted field for saving the NodeZero API Key

Create the Input

On the 'Inputs' tab, click "Create New Input"

- `Name` (required) is a simple name that gets used by the modular input
- `Description` (optional) is for any notes about the modular input
- `API Account` (required) is a dropdown single select to choose the account for which you wish to pull data
- `Index` (required) the index to which you would like to ingest NodeZero data
- `Polling Interval` (optional) Default is 86400 seconds (daily). How frequently you want to poll for new data

Once the input is saved, it immediately begins attempting to pull data from the NodeZero API

Index macro

For the sample dashboard to work, it relies on a macro called: `h3_index`. The default value for this macro is: `()`, and will only work if your events are being sent do the `default` index (usually `main`). Set the macro (via

`Settings > Advanced Search > Search Macros`) to whatever index you selected when defining your modular input (e.g., `(index=my_nodezero_test_index)`

Verify data is being indexed

To make sure the modular input is doing its job, you can run the following search:

```
`h3_index` | stats values(sourcetype) as st
```

You should see three sourcetypes:

```
h3:nodezero:api:action_logs
h3:nodezero:api:host_export_csv
h3:nodezero:api:weakness_export_csv
```

There is also a non-navigable dashboard at `/en-US/app/nodezero/kvstore_state` that will show a table based on the KVstore the modular input uses to store app state. This can be used during troubleshooting to see what data has been pulled for each pentest.

4.4.4 Troubleshooting

For troubleshooting information, view `ExecProcessor` logs and the nodezero logs:

```
    index=_internal (sourcetype=splunkd nodezero component=ExecProcessor) OR sourcetype="nodezero-*"
```

Note: Don't forget the asterisk on the end of `sourcetype=nodezero*`, as they often show up as `nodezero-too-small`.

For more logs: set `LOGGING` to `DEBUG` in the Configuration tab

4.4.5 Support

Email splunk@horizon3.ai with any questions, comments, or concerns. Feedback is greatly appreciated!