The NetRise Platform

User Guide and Documentation

January 20, 2025

Table of Contents

Introduction

The NetRise platform provides an interface for the analysis of firmware images that are typically found in eXtended Internet of Things (XIoT) devices. There are many use cases for analyzing the firmware of devices, ranging from device manufacturers and asset owners to consultants and MSSP organizations. The purpose of this document is not to explore and explain use cases, rather this is the technical documentation that details the full features and functionality of the platform. For more information on specific use cases and operationalization of the product in a variety of settings, please see the NetRise Knowledge Base.

Account and Organization Settings

Login and Account Setup

Access to the NetRise platform is managed through the 'Settings' tab on the left side of the Platform interface. Here, users can be added, deleted, deactivated, assigned roles, and more.

Adding Users

To invite a user to the NetRise Platform, an Owner must enter the user's email address into the invite box, select the desired role for that user, and click 'send'. This will automatically send an invite email to the user to set up their account credentials. Additionally, the user will receive an invite to the NetRise Knowledge Base, where they can access documentation, frequently asked questions, and more.

Managing Roles

Once invited to the platform, a user's role can be changed by any Owner within the platform. In order to change the role of a user, click on the arrow next to the desired account and choose either 'Promote User Role', or 'Demote User Role', as applicable.

The following roles are available. For a detailed table comparing permissions between roles, see Appendix C: Detailed Permissions Matrix

Owner

Owner accounts have read and write access to the organization's data in the platform. They are responsible for administering other accounts including inviting, promoting, demoting, and deleting. Owners can take the following actions on other users in the organization:

- Invite users to the NetRise Platform
- Disable users
- Delete users
- Reset password for users
- Remove users (only available for external accounts. See the Multiple Organization Access section for more details about external accounts.)

Operator

Operator accounts have read and write access to the organization's data in the platform. They can upload assets and view data including downloading assets, extracted assets, individual files, and exporting SBOMs. They can create, modify, and delete Asset Groups as well as add and remove Assets to Asset Groups. They cannot invite, promote, demote, or otherwise administer other accounts.

Member

Member accounts have read-only access to the organization's data in the platform including downloading assets, extracted assets, individual files, and exporting SBOMs. They cannot upload assets, create, modify, or delete Asset Groups, or manage other users.

Account Disabling

To revoke access to a particular user, an Owner can choose either “Disable Account” or “Delete Account” from the drop-down menu next to that account. The “Disable Account” option will prevent the user from logging into the NetRise Platform but can still be re-enabled at a later time. The “Delete Account” option will completely remove the user from the NetRise Platform. They must be re-invited in order to provide access to that email address again.

These options will also prevent the user from logging into any other organizations that they may belong to in the NetRise Platform.

These options are not available for users who have the “external” badge on their account. Use the “Remove Account” option to prevent the user from accessing your organization. See the Multiple Organization Access section for more details about external accounts.

Password Reset

If a user is unable to login due to a forgotten password or their password is believed to have been compromised, an Owner can initiate a password reset through the same menu by choosing ‘Reset Password’.

Using this option will disable the user’s account and then send an email to the user with instructions on how to reset their credentials. The user’s account will automatically be re-enabled when they set a new password.

Users can also manage their own password by navigating to the ‘Security’ tab of the ‘Profile’ section or by using the “Forgot password?” on the login page.

This option is not available for users who have the “external” badge on their account. See the Multiple Organization Access section for more details about external accounts.

Multiple Organization Access

Users can be invited to multiple organizations and access those organizations with the same set of credentials in the NetRise Platform. Each organization has a primary domain, which is typically set to the domain of the company for which the organization was created. Any users who are invited with an email that does not match the primary domain of the organization will be identified as “external users” (indicated by an ‘external’ tag on the user in the User Management page).

Please contact NetRise Support If your organization needs multiple primary domains. This may be the case if your users may have multiple email domains (e.g. @acme.com and @acme.co.uk)

Managing External Users in your organization

When managing your organization, you will only have the following actions available to manage external users:

- Promote User Role
- Demote User Role
- Remove User

External users follow the same permissions as described in the Managing Roles section. Use the “Remove User” option to prevent an external user from logging into your organization.

Managing Users who exist in multiple organizations

When a user is marked as external in an organization, their account cannot be enabled, disabled, or reset by that organization. These actions can only occur from an organization whose primary domain matches that of the user’s (i.e. the user does not have an “external” badge.)

It is important to remember that using the “Disable User” and “Delete User” options will completely prevent the user from logging into the NetRise Platform including your organization and any other organizations that they may be invited to.

Users with access to more than one organization can select the organization to access (after logging in) by using the drop-down menu at the top left of the page, which is always visible.

Appearance

Allows the user to choose how the NetRise Platform looks to you. Select a theme, or sync with your system and automatically switch between light and dark themes.

Assets

The Assets page provides the interface for uploading new firmware or SBOMs to the platform, as well as viewing firmware or SBOMs that have already been uploaded. The Assets table displays the following information about uploaded assets:

Field Name	Description	Provided By	Editable After Upload
Name	The name of the Asset	User	Yes
Vendor	The vendor of the Asset	User	Yes
Product	The product model associated with the Asset	User	Yes
Version	The version of the Asset	User or Platform	Yes
Groups	The number of Asset Groups that the Asset belongs to. (Click the number to show more details.) See Asset Groups for more information.	User	Yes
Type	The type of Asset (either Firmware or SBOM)	Platform	No
SHA256	The SHA256 hash of the Asset	Platform	No
Uploaded On	When the Asset was uploaded	Platform	No
Last Modified	The last time the Asset was analyzed	Platform	No
Analysis Status	The status of the analysis of the Asset	Platform	No
Overall Risk	The risk score of the firmware Asset, after it has been analyzed	Platform	No

Table columns can be toggled on or off by clicking the settings button at the top-right of the table and checking or unchecking the desired column.

In addition to the fields mentioned above, there is an 'Actions' column, which provides a few options:

- Edit Asset Details – Edit any user-provided fields for the asset
- Manage Groups - View and modify the Asset Groups that the Asset belongs to
- View Analysis Log – View standard output logs for the latest analysis of the asset
- Download Asset – Download the raw image that was originally uploaded
- Download Extracted Asset – Download the results of the extraction process (available after the asset has been processed successfully)

Lastly, each row can be expanded by clicking the sideways caret on the left side of the row. This will reveal high level information about the analysis results for that asset broken down by the same categories that are seen in the individual asset reports. These findings bars can be clicked to bring the user to the relevant section of the report for that asset.

Accessing Reports

Reports for each asset can be accessed (after processing has successfully completed) by clicking either the name of the asset or by clicking the risk score.

Uploading Firmware or SBOMs

Clicking the 'Upload' button in the top-right corner of the page brings the user to the upload form which is comprised of the following steps and options:

1.) **Identify the asset**

Select a file for upload by clicking in the dotted box or drag-and-dropping a file into the box. When uploading SBOMs, please ensure that the selected file has one of the file extensions:

- .cdx.json
- .cdx.xml
- .cyclonedx.json
- .cyclonedx.xml
- .spdx
- .spdx.json

The name field will automatically populate with the name of the selected file. NetRise recommends keeping this value; however, users can change it to meet their own specific needs. For example, it can be used for a uniform asset name field to note the physical location and asset identification number associated with the device running the firmware (for enterprise users), or simply the device model and firmware version (for device manufacturers.) The name given to an asset does not need to be unique.

2.) **Add Information about the asset**

This step exposes three optional fields: the Vendor, Product, the Version, and the Type of the Asset. Users should only provide this information if it is known to be accurate, otherwise it is recommended to leave these fields blank.

Note that these fields will automatically populate if the user is uploading an SBOM that has the required metadata fields set; however, the user can change the values to meet their own needs.

3.) Using the “Add to group(s)” field

The “Add to group(s)” dropdown menu allows the Asset to be automatically added to the selected Asset Groups when it is uploaded. The Asset Groups that it belongs to can also be changed after the Asset is uploaded making this field optional during upload. See the Asset Groups section for more information about Asset Groups.

4.) Using the advanced “Asset CPE” field

Expand the “Show advanced” section at the bottom of the Asset Upload modal to access the “Asset CPE” field. When populated, this field will associate the entered CPE with the uploaded Asset. When the Asset is analyzed, the CPE will be used to look up additional vulnerabilities that may be directly associated with the asset rather than the components inside of it.

The Asset CPE can be added, changed, or removed from an Asset by re-uploading the file and setting the Asset CPE field to the desired value. When the Asset is re-uploaded it will automatically reprocess to correct the vulnerability information.

5.) Review

Simply review the provided information and click “Upload” to upload the asset to the platform. Active uploads are displayed in the Task Tray in the bottom right corner of the product. Users can choose to upload additional assets or navigate around the product while there are active uploads.

When the upload completes, the NetRise Platform will automatically check if that file has already been uploaded to your organization. If the uploaded file’s hash is found the Name, Vendor, and Version and Last Modified fields will be updated on the original upload. A new asset record will not be created.

Overview

The Overview page is the default landing page that users see when logging into the NetRise Platform. This page provides a variety of statistics that represent all assets that have been uploaded to the platform. When logging in the first time, no data will be displayed on this page. Navigate to the Assets page to upload your first asset and begin populating data into the NetRise Platform.

Across the top of the screen are four widgets which display general metrics about the firmware uploaded to the platform:

- Total number of assets
- Total number of files that have been extracted from all assets
- Total *unique* vulnerabilities (CVEs) that have been identified
- Active analysis jobs that are currently executing

The remainder of the charts on the Dashboard are focused on highlighting risk, and specifically vulnerabilities that have been identified across all assets in the platform. All displayed charts and graphics are interactive and can be clicked to view the data and/or reports relative to what section of the chart was clicked on. More information on these interactions can be found in the subsequent sections specific to each chart.

Asset Reports by Risk Category

Displays the distribution of assets by the risk category (severe, significant, moderate, conservative, negligible, or undetermined). More information on risk scores and categories can be found in the Report Summary section.

Actions

Clicking on one of the portions of this donut chart will open a drawer that shows the list of reports that fall under that particular risk category. Clicking on either the report icon (on the left) or the risk score (on the right) will take the user to the report for that specific asset. Any time reports are displayed in this manner, they can be directly accessed through this mechanism.

High-Profile Vulnerabilities

This widget displays information about any vulnerabilities that are associated with well known named exploits that are found in the Organization. Counts of unique exploits, associated CVES and affected assets are shown in the widget.

Actions

Any of the three icons in this widget can be clicked to view the data from different perspectives. Clicking on the top “Unique Exploits” icon will open a drawer that displays the list of named exploits that are associated with identified CVEs. For each identified exploit, the “Affected Assets” icon can be clicked, which will then display a list of assets in which those CVEs and exploits were identified.

Clicking on the middle “Associated CVEs” icon will open a drawer that lists all identified CVEs that have been attributed to (or leveraged by) the named threat actors mentioned above. All CVEs in this list can be further investigated by clicking on the magnifying glass on the left side of the table, which will display data identical to what is seen when investigating a CVE on the Vulnerabilities tab of an individual report. Additionally, clicking the number in the "In Assets" field will display a list of assets in which that particular CVE was identified.

Lastly, clicking on the bottom "Affected Assets" icon will open a drawer that shows all assets in which the associated CVEs have been identified.

CISA Known Exploited Vulnerabilities

This widget operates nearly identically to the High-Profile Vulnerability widget mentioned above, but displays data regarding vulnerabilities that currently appear in CISAs Known Exploited Vulnerability (KEV) Catalog that are found in the Organization. Counts of unique vulnerabilities with associated KEVs, KEVs that are past due and assets that have KEVs associated with them.

Actions

As with the High-Profile Vulnerability widget (above), users can click any of the three icons ("Unique KEVS", "Past Due", and "Affected Assets") to open a drawer which displays the relevant data in the same format mentioned above.

Vulnerabilities of Interest

This chart displays vulnerabilities (CVEs) found in the Organization that have known exploits that either:

- Exist "in the wild"
    - This means the vulnerability has been exploited by malicious actors, but has not been definitively tied to a named threat actor, ransomware campaigns, or botnets
- Have been leveraged in known ransomware campaigns
- Have been leveraged as part of a known botnet
- Are associated with named threat actors

This chart can be toggled at the top right to either display total counts of all CVEs identified (by clicking on "Occurrences") or to show only unique CVEs identified (by clicking on "Unique")

Vulnerabilities by Priority Score

This chart classifies vulnerabilities (CVEs) that are found in the Organization into a Priority Score of 1 to 5.

The Priority Score is determined by the following criteria:

- Priority 1: Vulnerabilities on the CISA KEV
- Priority 2: CVSS Base Score greater than 6 and EPSS Score greater than 0.2
- Priority 3: CVSS Base Score greater than 6 and EPSS Score less than 0.2
- Priority 4: CVSS Base Score less than 6 and EPSS Score greater than 0.2
- Priority 5: CVSS Base Score less than 6 and EPSS Score less than 0.2

NetRise recommends investigating and addressing vulnerabilities in the following order: Priority 1, Priority 2, Priority 3, Priority 4, Priority 5.

Vulnerabilities by Age

This chart displays the total number of vulnerabilities that have been identified across all assets, broken down by CVE age (based on when the CVE was published) and CVSS severity.

Actions

Clicking on any of the bars in the three aforementioned vulnerability charts will open a drawer that displays the unique CVEs associated with that particular category subset of vulnerabilities:

- The CVE ID
- The name of the component in which the CVE was identified
- The total number of occurrences of that CVE across all firmware images
- The number of firmware images in which that CVE is present

All CVEs in this list can be further investigated by clicking on the magnifying glass on the left side of the table, which will display data identical to what is seen when investigating a CVE on the Vulnerabilities tab of an individual report. Additionally, clicking on the number in the 'In Assets field will display a list of assets in which that particular CVE was identified.

Changing the Overview Scope

By default, the Overview page shows information about all of the Assets that exist in the Organization regardless of which Asset Groups they belong to. The scope of the Dashboard can be changed to only show information about Assets that belong to specific Asset Groups by using the "Asset Groups" dropdown menu in the top right corner of the page. See the Asset Groups section for more details about Asset Groups.

Asset Groups

Asset Groups provide a way to group Assets together to help keep Assets organized, see aggregated metrics, scope the Overview page, and scope Searches. There are no limits to the number of Asset Groups that can be created. Similarly, there are no limits to the number of Asset Groups that an Asset can belong to allowing for unlimited ways to divide and view information in the Organization.

Please note that users must have an “Owner” or “Operator” role to create, delete, or modify Asset Groups. See Appendix C: Detailed Permissions Matrix for more detailed permission information.

Creating Asset Groups

Asset Groups can be created by navigating to the “Groups” tab of the left-hand navigation and then clicking the “Create group” button in the top right corner of the page.

The only required field to create an Asset Group is the “Group Name” which must be unique in the Organization. This allows empty “placeholder” Asset Groups to be created to help with organizing future Assets.

An optional “Description” can be added to the Asset Group to help explain the purpose of the Asset Group. This description is displayed in various places including the Asset Group List and any tooltips displayed next to the Asset Group name such as in the “Manage Groups” action accessed from the Assets page.

When creating an Asset Group, it is also possible to upload a new Asset that will automatically be added to the Asset Group, as well as selecting one or more Assets from the Assets dropdown menu, which will be added to the Asset Group once it is created.

Deleting Asset Groups

Asset Groups can be deleted by opening the “Actions” menu (3 dots) to the right of the Asset Group’s row and selecting the “Delete Group” option. When clicking this option, a confirmation message will be displayed asking to confirm the action.

When deleting an Asset Group only the Asset Group is deleted. Any Assets that are in the Asset Group will remain in the Organization and will also remain in any other Asset Groups that they belong to.

Editing Asset Groups

The Asset Group Name and Description can be changed by opening the “Actions” menu (3 dots) to the right of the Asset Group’s row and selecting the “Edit Group” option.

Adding Assets to Asset Groups

There are many ways to add Assets to Asset Groups:

- On the Groups page
    - During Asset Group creation,
    - Open the “Actions” menu to the right of an Asset Group and select “Add Asset”,
- On the Assets page

    - During Asset Upload,
    - Open the “Actions” menu to the right of an Asset and select “Manage Groups”,
    - Select the checkbox to the left of any number of Assets and select “Add to Groups”
- On an Asset Report
    - Open the “Actions” menu in the top right corner of the page and select “Manage Groups”

An Asset can belong to any number of Asset Groups; however, it can not belong to the same Asset Group multiple times.

Removing Assets from Asset Groups

There are many ways to remove Assets from Asset Groups:

- On the Groups page, expand an Asset Group, then open the “Actions” menu to the right of an Asset. Select the “Remove from Group” option.
- On the Assets page, open the “Actions” menu to the right of an Asset and select “Manage Groups”
- On an Asset Report, open the “Actions” menu in the top right corner of the page and select “Manage Groups”

The Assets page also includes an additional workflow to allow all select Assets to be removed from all of the Asset Groups that they belong to. To do this, select any number of checkboxes to the left of the Assets and then click the “Remove from Groups” button at the top of the table.

Asset Reports

Accessing a report from the Assets page will bring you to a report specific to the asset that was uploaded. The following sections outline the different analysis areas and artifacts that can be found in a report.

There are some options that are available in all report tables, including the option to toggle table columns (click the settings icon at the top right of any table), show or hide column filters (click the filter icon in the top right of any table), and correlations (described in detail in the following section).

Correlations

For nearly every datapoint in an asset, a field named "correlations" will be present, which displays the number of other assets in the current organization where that same datapoint was found. Correlations are generated in different ways depending on the associated data point:

- Components: Compares the CPE and/or PURL (identifier) of the component
- Binaries: Compares the SHA-256 hash of the binary
- Credentials: Compares the combination of username and password (string comparison)
- Misconfigurations: Compares the results (pass, fail, not applicable) of the misconfiguration check
- Vulnerabilities: Compares the CVE ID
- Private Key: Compares the hash of the private key
- Public Key: Compares the hash of the public key
- Certificate: Compares the SHA-1 fingerprint of the certificate

The correlations value can be clicked to display the other firmware reports in which that particular datapoint was also present. For more information on use cases regarding correlations, please see this entry in the NetRise Knowledge Base.

Image Metadata

Metadata about the current asset can be found in the breadcrumbs at the very top of the Report page. Here, the user can click the "copy" icon next to the report GUID to copy the URL to the report, or click the "i" icon to display additional metadata:

- The organization that created the report
- The date and time when the report was created
- The SHA256 hash of the asset
- The MD5 hash of the asset

Metadata about the current asset is located in the top section of the report, which is visible regardless of the current report tab that is selected. Here the following metadata can be found:

- Name – The name of the asset
- Product – The vendor, product, and version (if applicable) of the asset
- The SHA256 hash of the asset
- Architecture – CPU architecture of the asset

- Last Modified - The timestamp indicating the last time the report data was modified

Summary

The summary page displays high-level charts and information summarizing the data that is found in the other Report tabs. Table 1 describes the available charts and the Report tab they are associated with:

Name	Description	Associated Tab
Analysis Summary	Total number of issues found with a drop down count by specific result (Vulnerability, SBOM, Misconfigurations, Binaries, Cryptography, Credentials, File System.	Summary
Vulnerability Information	Breaks down vulnerabilities by criticality (CVSS Score) Low, Medium, High, Critical. **Note:** This number excludes vulnerabilities that have a Status of False Positive, Not Affected, Resolved, or Resolved with Pedigree. See the Vulnerability Remediations section for more details.	Vulnerabilities
Top Offenders	Shows the components (plus the operating system, if applicable) with the highest number of vulnerabilities. **Note:** This number excludes vulnerabilities that have a Status of False Positive, Not Affected, Resolved, or Resolved with Pedigree. See the Vulnerability Remediations section for more details.	Vulnerabilities
Cryptography	Displays identified certificates, public keys, and private keys. Breaks down certificates with invalidities and keys where the paired key was also identified.	Cryptography
Exploited Vulnerabilities	Highlights the same vulnerabilities that are shown in the Vulnerabilities of Interest chart on the Dashboard.	Vulnerabilities
Credentials	Shows counts of user accounts (with or without a password hash) and other identified password hashes in the image. Also displays a count of the number of password hashes that have been cracked.	Credentials
SBOM Component Makeup	Displays the type and number of components identified in the asset.	SBOM

Misconfiguration Checks	Two charts which show the results of the various misconfiguration checks.	Misconfigurations
Binary Protections	Shows the percentage of binary protections that are enabled across all identified binaries.	Binaries
Malware Findings	Displays any malicious files that were detected.	Summary
Component Licenses	Shows the prevalence of all identified component licenses.	SBOM

Table 1: Report Summary Charts

Vulnerabilities

Displays all identified CVEs associated with known third party components in the asset. The top-level table displays the following information for each vulnerability:

- **CVE ID:** The CVE ID of the vulnerability
- **Severity:** The CVSS severity of the vulnerability
    - Defaults to CVSSv3 if available, otherwise will display CVSSv2
- **Component:** The name of the affected software component
- **Version:** The version of the affected software component
- **Vendor:** The vendor of the affected software component
- **Exploit Maturity:** The maturity level of exploits that were identified for the vulnerability, which is one of three values:
    - Unproven: No exploit has been identified for the vulnerability
    - Proof of Concept: Exploit(s) exist, but have not been observed as weaponized
    - Weaponized: Exploit(s) exist, and have been leveraged in ransomware campaigns or botnets, are associated with known threat actors or named attacks, or exist as part of an exploit framework (e.g. Metasploit)
- **Correlations:** Described above
- **Status:** Displays the VEX Status applied to the vulnerability. See the Vulnerability Remediations & Vulnerability Exploitability eXchange (VEX) section for more details.

Optional table columns include:

- **Complexity:** The CVSS complexity for the vulnerability
    - Defaults to CVSSv3 if available, otherwise will display CVSSv2
- **File:** The file in which the vulnerability exists
    - This is a link (in cases where the vulnerable component is not the kernel or operating system itself) that will take the user to the file in the File System tab.
- **Vector:** The CVSS attack vector for the vulnerability
    - Defaults to CVSSv3 if available, otherwise will display CVSSv2

Quick filters are available above the vulnerabilities table to provide the ability to filter by CVSS criticality, the various exploit categories that are available (ransomware, botnets, known actors, known attacks, or all vulnerabilities with exploit code), or if the vulnerability appears in the CISA KEV Catalog.

Vulnerabilities can be expanded by clicking the magnifying glass on the left side of the desired entry, which will open a drawer with more details about a vulnerability, including:

- CVSS Description
- Published and modified dates for the CVE
- The source database for the CVE
- References to security advisories, solutions, and tools related to the vulnerability
- CVSSv2 and CVSSv3 details, where applicable
- Exploit Prediction Scoring System (EPSS) score
- Common Weakness Enumeration (CWEs) associated with the vulnerability
- Exploit availability information and references to exploit code that exists, where applicable

Vulnerability Remediations & Vulnerability Exploitability eXchange (VEX)

Vulnerability remediations allow statuses to be applied to vulnerabilities to help communicate if a product is affected by a vulnerability, if the vulnerability exists but is not exploitable, if the product is not affected by a vulnerability, and many other cases. These statuses can be used to help track work and progress within an organization. They can also be used to communicate these statuses externally in the form of a Vulnerability Exploitability eXchange (VEX) document.

There are 3 ways to start a Vulnerability Remediation workflow:

1. In the Vulnerabilities tab of a Report, click the Remediate button in the Actions column of the table,
2. In the Vulnerability Details drawer, click the Remediate button in the top right corner of the drawer,
3. Select multiple Vulnerabilities in the Vulnerabilities tab of a Report and click the Remediate button in the banner that is displayed at the top of the table

The Vulnerability Remediation workflow is a three step modal. Step 1 allows the VEX Status to be selected. VEX Statuses can be one of:

- **Unreviewed:** This issue has either not been reviewed or remediated.
    - This is the default status.
- **Exploitable:** Actions are recommended to remediate or address this vulnerability.
- **False Positive:** The vulnerability is not specific to the component or service and was falsely identified or associated.
- **In Triage:** It is not yet known whether these product versions are affected by the vulnerability. An update will be provided later.
- **Not Affected:** No remediation is required regarding this vulnerability.
- **Resolved:** Represents that these product versions contain a fix for the vulnerability.

- **Resolved with Pedigree:** The vulnerability has been remediated and evidence of the changes are provided in the affected components' pedigree containing verifiable commit history and/or diff(s).

Step 2 allows more specific details of the analysis to be recorded. All statuses allow for a free-form text field called “Details” to be populated. This field can be used to communicate notes internally as well as externally in a VEX document. Additional fields will be presented for the following Statuses:

- **Exploitable:** An optional field named “Response” will be available. This field can be used to indicate a manufacturer’s response to the exploitability of a vulnerability and can be used by consumers to determine the best course of action. Available options in the Response field are:
  - Cannot Fix
  - Will not fix
  - Update
  - Rollback
  - Workaround Available
- **Not Affected:** An optional field named “Justification” will be available. This field can be used to state the rationale of why the status was selected. Available options in the Justification field are:
  - Code not present
  - Code not reachable
  - Requires configuration
  - Requires dependency
  - Requires environment
  - Protected by compiler
  - Protected at runtime
  - Protected at perimeter
  - Protected by mitigating control

Step 3 acts as a confirmation page and provides an option to immediately open the SBOM Export modal.

After a Vulnerability Remediation is applied to a Vulnerability, a Note icon will be displayed in the Status column. Hovering over this icon will display the following information:

- The Justification field, if provided
- The Response field, if provided
- The Detail field, if provided
- The user who last updated any of the fields

Removing Risk from Reports

The following status will remove risk from the Report and Organization:

- False Positive
- Not Affected
- Resolved
- Resolved with Pedigree

Removing risk from a Report will cause the following to happen:

- The Asset's Risk Score will decrease
- The Vulnerability Information, Top Offenders, and Exploited Vulnerabilities widgets on the Report Summary page will not include the remediated vulnerabilities
- The High Profile Vulnerabilities, CISA KEV, Vulnerabilities of Interest, Vulnerabilities by Priority Rating, and Vulnerabilities by Age widgets on the Dashboard will not include the remediated vulnerabilities

SBOM (Software Bill of Materials)

Component Listing

Displays all identified software components, with the following information at the top level of the table:

- **Name:** Name of the component
- **Version:** Version of the component (if applicable)
- **Vendor:** Vendor that developed the software (if applicable)
- **Type:** Component type, including an icon for specific package managers, where applicable
- **License:** Open source license for the component (if applicable)
- **Verified Via:** The method(s) by which the component was identified (described further in the Component Identification section below)
- **Vulnerabilities:** Vulnerabilities identified for the component
- **Correlations:** Defined above
- **Exploits:** Information about exploit availability, as described in the Vulnerabilities of Interest section. Hover over the icons for more information

SBOM Export

The SBOM can be exported from the platform by clicking the "Download SBOM" button above the component table. The following export formats are supported:

- CycloneDX
  - JSON and XML
- SPDX
  - JSON and TAG

Additionally, an options are available to include additional data with an SBOM

Known Unknowns: Include any additional files that the Platform identifies in the asset but is not associated with a known third party component. More information on Known Unknowns can be found in the NTIA-defined minimum elements of an SBOM.

NetRise Fidelity – Includes a custom namespaced property displaying NetRise's identification confidence. Possible values are Low, Medium, High and Certain.

Vulnerabilities – Includes identified CVEs and vulnerability advisory references for each component.

VEX Analysis - Includes VEX-related fields (Status, Justification, Response, Comments) for any Vulnerabilities that have them set. The Vulnerability toggle must be enabled when this option is selected. See the Vulnerability Remediations & Vulnerability Exploitability eXchange (VEX) section for more information. Currently VEX-related fields are only available in a CycloneDX conforming format.

Quick Filter Options

Above the component table are a variety of ways to group and filter components:

- **Severity chips:** Critical, high, medium, and low can be toggled to include components that have vulnerabilities with the relevant CVSS severity.
- **Grouping:** The grouping menu provides options to group components by the following criteria:
    - **Vendor:** Group by the software vendor for the component
    - **License:** Group by the open source license used by the component
    - **Type:** Group by the component type
    - **Package Manager:** Group by the package manager used to install the component
- **Has Exploits:** Toggle this option to only show components that have vulnerabilities with publicly available exploits

Component Details

All components in the table can be expanded by clicking the magnifying glass on the left side of the row in the table. This will open a drawer that provides additional context about the component, including:

- Additional details about the exploits that were identified, if applicable
- Vulnerabilities that are associated with the component
- Known exploits that are associated with the component
- Files that are associated with the component
    - These are links that can be clicked to navigate to the applicable file in the File System tab of the report
    - Each file can be expanded with the down caret to show MD5, SHA1 and SHA256 hashes of the file
- Dependencies and dependents of the component
    - These are links that can be clicked to view the applicable dependency or dependent. If the linked component has further dependencies or dependents, those will also be present and clickable to navigate the dependency tree
    - Where applicable, an "On Disk" indicator will be present next to the dependency or dependent name, which indicates that the dependency exists on the flat image and is not something that is resolved at runtime.

Component Identification

The NetRise Platform utilizes numerous component identification methods, which contribute to the fidelity (essentially a confidence level) associated with the given component. These component identification methods are grouped into three primary categories, as described below.

Heuristic
Indicates a component that was identified through signature-based means such as string searches, regular expression matches, and other proprietary heuristic methods.

Cryptographic Hash
Indicates a component that was identified through a variety of hash-based mechanisms. The two primary hash-based detections that are in use are the following:

Function Hashing
All binary/application type components are broken down into their individual functions, which are hashed and then compared to a known dataset of over 1 trillion function hashes. This data can be used to determine a percentage of functions that match known components, and assign a confidence value to the unknown component as a result of this comparison.

Package Hashing
For potential libraries or packages that are identified, hashes of individual package files can be taken and compared to a known dataset - similar to the analysis that is done with function hashing – to determine if a large percentage of the package files match a known sample. This data can be used to assign a confidence value to the

Package Manager
Indicates a component that was identified through parsing package manifest information.

Component Fidelity
Components can be identified through more than one method, and will be labeled as such in the "Verified Via" column in the top-level table, as well as in the component details when expanding the component entry. The method(s) by which a component is identified will contribute to the "Fidelity" of that component, which is a field that is present in SBOM when exported. Fidelity can include the following values:

- **Low:** Indicates a low confidence in the component information. Typically will occur when a component is only identified through heuristic methods and/or does not have a definitive version.
- **Medium:** Indicates a moderate confidence in the component information. Typically will be the value for kernel module components
- **High:** Indicates a high level of confidence in the component information. Typically occurs if the component is identified through multiple methods, but also often is the case for hash and package manager detections individually.

- **Certain:** The highest level of confidence which typically occurs with exact hash matches to known software repositories.

Misconfigurations

The misconfigurations table pulls data from a wide variety of analysis plugins and correlates relevant data points looking for a variety of configurations that could pose risk to the system. The following misconfiguration checks are available:

Misconfiguration Name	Description	Risk	Remediation
Weak hashing algorithms found	Identifies passwords hashed with weak algorithms	Weak hashing algorithms (MD5, SHA1, and descrypt) could provide attackers with easy access to plaintext passwords	Replace weak hashing algorithms with stronger hashing algorithms
Users with no password set	Identifies user accounts that do not have a password set	Generally considered bad practice. Could allow a malicious actor to easily access the system.	Disable login or set a password for the affected user(s)
Overly permissive access to passwd files	Identifies 'passwd' files that are writable by non-root accounts	Could allow a malicious actor to easily modify passwords, login shells, and other permissions for privilege escalation.	Modify the permissions to restrict access to the affected file(s)
Multiple users with UID 0	Identifies if multiple user accounts have a UID of 0	Typically, UID 0 is reserved for the 'root' user. Any other accounts configured with this UID could inadvertently obtain root access to the system.	Change the UID(s) for the affected user(s)
Binaries with memory corruption vulnerabilities	Correlates identified CVEs with binary protection mechanisms, identifying binaries that are vulnerable to	The presence of a memory corruption vulnerability paired with a lack of binary protection mechanisms greatly increases the	Update the binaries to patched versions and/or enable binary protections for affected binaries

	a memory corruption vulnerability and do not have all memory corruption binary hardening mechanisms enabled.	likelihood of an exploit attempt succeeding which could easily lead to privilege escalation or code execution.	
Authorized key with private key on the system	There is a public key used to authenticate with a matching private key on the same system.	Having both parts of a key pair on the same system poses a risk because a malicious actor could easily use these two pieces of information to decrypt data, intercept communications, or authenticate to other systems where this key pair is used.	Remove the private key from the filesystem, or remove it from the authorized keys list
fstab should always have permissions of 0644	Identifies if the /etc/fstab file has permissions other than 0644 (rw-r-r)	fstab having permissions other than 0644 could lead to mount points being changed by unauthorized users	Modify the permissions of the fstab file to 0644
Insecure services start at boot	Identifies if commonly abused insecure services (e.g. telnet, FTP, and TFTP) are enabled at boot time	There are services (e.g. telnet) which are known to be insecure in their operation, and can be leveraged by attackers to gain unauthorized access to data or the system as a whole	Replace or disable known insecure services
SELinux is disabled	Identifies if SELinux is found on the system and is explicitly disabled	SELinux (Security-Enhanced Linux) is a security architecture for Linux systems that allows	Modify the configuration to enabled SELinux and re-enable protections.

		administrators more control over user access to the system. Typically, SELinux is enabled by default, and therefore can be considered a risk if it is disabled. More information can be found here.	
Sudoers file missing	The sudoers file was not found in the default location (/etc/sudoers)	The sudoers file allows administrators to allocate system rights to users. When this file is not present, it could lead to unauthorized users running commands with elevated privileges.	Install a sudoers file with a a sufficiently locked down policy for system accounts (if using the sudo binary)
Cronjobs found with weak permissions	The files executed by cron have weak permissions, such as being world-writeable	Attackers could embed malicious code in these files and have them automatically executed on the system, often with high privileges.	Modify the permissions to restrict access to the executed files.
One or more compilers exist	One or more compiler was identified in a common installation path	Compilers that are present on a system can be leveraged for post exploitation code execution	Remove the compilers from production images to avoid exploitation
GTFOBins installed with setuid bit enabled can lead to privilege escalation	Identifies the presence of executables that are commonly used for post exploitation privilege escalation or lateral movement within the system.	The presence of these binaries, coupled with the fact that the setuid bit (which runs executable files with the permission of the owner, rather than the current user) can lead to various	Limit the permissions or remove the identified binaries from the manufactured image

		post-exploitation activities. More information can be found here.	
Services missing configuration files	Identifies services such as sshd and apache that don't have explicit configuration files	Services without explicit configuration files are subject to run with default settings, which often include insecure configurations	Include configuration files for the affected services, and implement secure configuration settings
Telnet server exists	Identifies if the telnet binary/service is present	Telnet is an inherently insecure protocol, due to unencrypted authentication	Remove telnet server from the installation or disable it
Insecure URL	A URL was identified (in any file) that uses an unencrypted protocol or contains passwords or other potentially sensitive information	Potential exposure of sensitive data	Review the identified URL(s) and determine if sensitive data is exposed. If so, take necessary action to obfuscate such data.

For each configuration check, there will be one of three results:

- **Failed**: The misconfiguration is present in the asset
- **Passed**: The misconfiguration is not present in the asset
- **Not Applicable**: Check is not applicable to the asset. For example, the "Authorized key with private key on the system" will display "not applicable" if there are no keys present on the system rather than displaying passed or failed.

Individual misconfiguration checks can be expanded using the magnifying glass on the left side of the row to open a drawer which shows the following details:

- **Name:** The name of the misconfiguration check
- **ID:** The unique identifier for the misconfiguration check
- **Description:** A description of what the misconfiguration check is looking for
- **Suggested Remediation:** Suggested actions that can be taken to eliminate or mitigate the risk associated with the misconfiguration
- **Failed Results:** References to the artifacts that were identified causing the misconfiguration check to fail (if applicable)

Binaries

The binaries table displays information regarding binaries that were identified in the extracted filesystem, specifically binary protection/hardening mechanisms and whether or not they are enabled in each binary. For each binary, the following details are displayed:

- **Name:** The name of the file
- **RELRO:** Described in detail below
- **Stack Canary:** Described in detail below
- **NX:** Described in detail below
- **PIE:** Described in detail below
- **Correlations:** Described above
- **File:** The full path to the file

RELRO

RELRO, or Relocation Read-Only is an exploit mitigation strategy that renders some of the binary sections read-only. There are two types of RELRO, partial and full.

- **Partial:** forces the Global Offset Table (GOT) to come before the BSS in memory, eliminating the risk of a buffer overflow on a global variable overwriting GOT entries.
- **Full:** makes the entire GOT read-only which removes the ability to perform a "GOT overwrite" attack, where the GOT address of a function is overwritten with the location of another function or a ROP gadget an attacker wants to run.

More information on RELRO can be found in this RedHat blog. Column values will either read Partial, Full, or Off.

Stack Canary

A stack canary is a value placed on the stack so that it will be overwritten by a stack buffer that overflows to the return address. It allows detection of overflows by verifying the integrity of the canary before function return. Column values will either read On or Off.

NX

The abbreviation NX stands for non-execute or non-executable segment. It means that the application, when loaded in memory, does not allow any of its segments to be both writable and executable. Entries in this column will either read On or Off.

PIE

Position Independent Executables (PIE) are an output of the hardened package build process. A PIE binary and all of its dependencies are loaded into random locations within virtual memory each time the application is executed. This makes Return Oriented Programming (ROP) attacks much more difficult to execute reliably. Column values will read one of the following:

- **On**: PIE is in place
- **Off:** no PIE in place
- **DSO**: the executable is a dynamic shared object which is position independent by default.

Cryptography

This Report tab has three sub-tabs, each of which are described below:

Certificates

The certificates table displays any identified X.509 certificates, including the following fields at the top level:

- **File:** The file in which the certificate data was found
- **SHA-256 Fingerprint:** The SHA-256 fingerprint of the certificate
- **Status:** Highlights any issues or invalidities with the certificate. If no issues are found, the column will display "Valid". If issues are found, the column will display "Invalid." For a full list of certificate invalidities, please refer to Appendix B: Glossary of X.509 Certificate Invalidities.
- **Correlations:** Described above.
- **Effective Permissions:** The permissions for the file in which the certificate data was found
- **Actions:** A button that lets you download the file containing the certificate

In addition to the top-level fields, each certificate (row) can be expanded by clicking on the arrow on the left side of the row to reveal additional metadata about the identified certificate. For more information about any of the subsequent fields, or on X.509 certificates in general, please reference additional documentation here.

Public Keys

The public keys table displays all public keys that were identified in the asset. This includes public keys that are part of certificates, as well as any other identified public keys. Each row in the table shows the following details for each public key:

- **File Path:** The file in which the public key was found
- **Algorithm:** The cryptographic algorithm used to create the public key, as well as the entropy value
- **Found Private Key**: Whether (True) or not (False) the associated private key was identified (and therefore the keypair can be considered compromised)
- **Match Hash:** A proprietary hash used to correlate public and private key pairs. It can also be used to identify keys with the same cryptographic properties.
- **Correlations:** Defined above.
- **Effective permissions:** The permissions for the file in which the public key was found
- **Actions:** A button that lets you download the file containing the public key.

Private Keys

The private keys table displays all private keys that were identified in the firmware image. Each row in the table shows the following details for each private key:

- **File Path:** The file in which the private key was found
- **Algorithm:** The cryptographic algorithm used to create the public key, as well as the entropy value

- **Found Public Key**: Whether (True) or not (False) the associated public key was identified (and therefore the keypair can be considered compromised)
- **Unique Hash:** A proprietary hash used to correlate public and private key pairs. It can also be used to identify keys with the same cryptographic properties.
- **Correlations:** Defined above
- **Effective permissions:** The permissions for the file in which the private key was found
- **Actions:** A button that lets you download the file containing the private key

Credentials

This Report tab has two sub-tabs, each of which are described below:

User Accounts

This tab displays any identified user accounts (and their associated metadata) that were identified in common Linux user account locations including /etc/passwd and /etc/shadow, as well as any files with 'passwd' or 'shadow' in their name. For each identified user account, the following fields are displayed:

- **Username:** The name of the user
- **Hash:** The password hash for the user
- **Hash Type**: The hash algorithm used for the password (if applicable), otherwise it will display one of the following:
    - **"no_login"**: For "Hash" values of 'X' or '*'
        - Password-based login for that user is not permitted
    - **"locked":** For "Hash" values of '!'
        - User account is locked
    - "**no_password":** For blank "Hash" values
        - No password set for that user
- **Cracked:** When a password hash is identified, this field denotes whether or not the password has been successfully cracked by the NetRise Platform. Cleartext passwords are intentionally not surfaced in the user interface, and can be requested by contacting NetRise Support.
    - **True:** hash was successfully cracked
    - **False:** hash has not yet been cracked
- **User Info:** Notes provided about that user account
- **Home:** Path to that user's home directory
- **Shell:** Path to that user's default shell
- **Correlations:** Defined above

Detected Hashes

This tab displays all detected password hashes in the asset (including those seen on the User Accounts tab) with the following information:

- **Hash:** The password hash
- **Type:** The hash type

- **Cracked:** Whether or not the hash has been cracked
- **Rounds:** The number of hash rounds used when generating the password hash
- **Hashcat mode:** The hashcat mode that was used in attempting to crack the hash. For more information on Hashcat, see the official page here.
- **Location:** The file in which the hash was identified

File System

The file system tab contains the entire file system that was extracted from the asset (if applicable). The left side of the display allows the user to explore the folder structure of the file system, while the right side displays the files within the selected folder, including the file name, id, permissions, size, and modified date. Individual files can be downloaded by checking the box next to the desired file and clicking the download button that appears at the top-right of the table.

Search

The NetRise Platform has two search mechanisms: Artifact Search and Trace Search.

Artifact Search allows the user to search assets for individual data points extracted from assets in your organization such as authentication hashes, CVEs, system configurations, certificates, and private keys.

Trace Search is an AI-powered code and text search which allows the user to search for text/code patterns and keywords in the files extracted from their assets.

Artifact Search

Artifact Search allows the user to query all assets for any data points that have been identified during analysis. To run a query, type the desired search term in the search box and press the "enter" key or click the "search" button. Artifact Search queries act as keyword searches within individual records.

Returned results will populate in the right-hand side of the display. Additionally, search results can be filtered by a variety of fields using the options on the left side of the display. Possible filters will vary but will include the parser that yielded the data point, the file path where the data was identified, the asset (by Component ID) to which the data point belongs to, as well as the severity of vulnerabilities in the search results. Multiple filters can be selected at once to narrow down search results while exploring the data. More information about each parser and subsequent fields can be found in the Parsers section of this document.

Trace Search

Trace Search uses a technology called Text Embeddings to create a representation of humans' natural language that a computer can understand and operate on. Essentially text embeddings allow computers to understand the "intent" of a particular script, block of code, or text file. This understanding allows a human to ask questions in ways that we understand and the computer can quickly and intelligently find things that are similar in their meaning or intent.

When the NetRise Platform extracts and analyzes an asset it automatically generates text embeddings for all Python, shell script, and plain text files. Once text embeddings have been generated, users can query their assets using code snippets and natural language to find matches ranked by the similarity percentage. It is designed to allow users to easily find vulnerable coding practices, risky configurations, and other text blocks that may be interesting to the user.

Trace Search results are returned as blocks of text within each unique file so users can easily see the affected parts of the file even if that file exists in different locations on multiple assets. The List View allows users to easily see the different blocks of text that matched the search while the Graph View presents a visual representation of the relationships between affected files, assets, and packages.

Trace Search has 2 search modes, Semantic and Keyword, which can service different use cases. Both are explained in more detail below.

Semantic Mode

The Semantic mode allows users to search for patterns and functionally similar blocks of text and code. For example, the following Python code could be abused to execute arbitrary code:

```
import requests
import subprocess

address = request.args.get("address")
cmd = "ping -c 1 %s" % address
subprocess.Popen(cmd, shell=True)
```

The next block of code is functionally equivalent to the above code but is structured a bit differently:

```
import requests as network_stuff
from subprocess import Popen as run

run("ping -c 1 %s" % network_stuff.args.get("address"), shell=True)
```

While functionally similar, it would be difficult for a keyword or exact text search to find these matches at scale. Semantic searching allows the user to search for things that are functionally similar and because semantic searching has some "grammatical" knowledge, it also lets users search for sentences using natural language. See examples below phrases that could be used to find coding or configuration issues within text files.

"Python using popen to execute code from a variable"

"A Python network socket is created and connects to a system which can then send data to /bin/bash"

"Python's interactive interpreter is created and executed with a user-defined payload"

"A hardcoded JSON Web Token (JWT) is found in a file which can be abused by malicious actors"

In Semantic Mode, it is recommended to start with a high Match Similarity value (75% or higher) and lower the value to gradually increase the number of results as low Match Similarity values (74% or lower) can lead to an overwhelming number of results.

Keyword Mode

The Keyword mode allows users to search with a more familiar experience by pinpointing exact text, words, and code within a file. For example, if a user wanted to determine if a specific string (such as an API key or password) is present in any analyzed files, they can input the specific value in the search box and see which files, assets, and packages it is found in.

In Keyword Mode, it is recommended to start with a very low Match Similarity value (10% or lower) and gradually increase the value to narrow the scope of results. This is because the Match Similarity is calculated on the entire matched block of text. In other words, a small keyword search term may only account for 10 characters in a 500 character block of text.

Searchable FIle Types

- Python
- Shell scripts
- Plain-text files

Parsers

The NetRise Platform analyzes firmware images with a variety of parsers, all with specific purposes that generate various interesting data points or items for review. The following table describes each of the parsers used by the platform, including the name of the parser, a description of what the parser is designed to identify, and the various fields available for search within each parser.

Parser Name	Description / Purpose	Fields
authorizedKey	Identifies SSH authorized key entries which can be used to authenticate to the SSHD server running on the system. Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	**Algorithm:** The encryption algorithm associated with the key **File Offset:** The file offset that the key was found at in the associated file **Key Strength:** A value of Weak, Adequate, or Strong to help determine if a key provides the necessary protections. **Match Hash:** A proprietary hash value used to correlate public and private key pairs. **Bit Size:** The size of the key
binary_protections	Analyze all binary files for the following hardening mechanisms: RELRO, Stack Canary, NX, and PIE. Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	**relro:** Whether or not RELRO is enabled **canary:** Whether or not stack canary is enabled **nx:** whether or not the No Execute (NX) flag is set **pie:** Whether or not PIE is in use (for more information see Binaries)
busyboxconf	Extracts configuration files from Busybox configuration files to help assess the configuration of Busybox on the system. Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	**Category:** Displays the category of the busybox configuration file associated with the Key and Value. **Key:** An individual configuration option within the busybox configuration file (such as “su”, “mount”, “passwd”, “ifconfig”, etc.) **Value:** The configured value of the key.

Parser Name	Description / Purpose	Fields
certificate	Displays all of the fields associated with each X.509 certificate identified in each asset. Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	The number of fields accessible from this parser are too numerous for this document. In general you can expect to find: ● Issuer information, ● Subject information, ● Public key strength and information ● Revocation status, ● Serial numbers, ● Thumbprints
component	Identify known components within the firmware image. Only verified components will be included in the output for this parser. In order for a component to be verified, it's component name, vendor, and version must be valid (validation is done via the NVD with the CPE dictionary https://nvd.nist.gov/products/cpe) Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	**cpe_uri:** the CPE of the component (in URI format) **license_type:** The type of license for the component **license_method:** how the license information was obtained **license_ID:** A short identifier for the license **license_text:** The text of the license, base64 encoded **version:** The version of the component **version_method:** How the version information was obtained **component_type:** The type of component **vendor:** The component vendor **name:** The name of the component **cpe_fs:** The CPE of the component (in formatted string format)
cronjobs	Identify cron files and parse the relevant lines in the file that reflect the various cron jobs therein. Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	**cronjob:** The entire cronjob string to be run

Parser Name	Description / Purpose	Fields
cve	Given firmware components (and their versions), conduct a lookup against the National Vulnerability Database (NVD) for CVEs	**severity:** The severity of the vulnerability **attack_complexity:** the difficulty or complexity of the attack associated with the CVE **base_score:** The base CVE score from 1-10 **integrity_impact: T**he impact to the integrity of the component or system **vector_string:** The vector string for the CVE **description:** A short description of the CVE **impact_score:** A score representing the direct consequence of a successful exploit **cve-id**: The ID for the CVE **attack_vector:** The context by which vulnerability exploitation is viable **privileges_required:** The level of privileges an attacker must possess before successfully exploiting the vulnerability **availability_impact:** The impact to the availability of the component or system **cvss_version:** The version of CVSS for this particular CVE **user_interaction:** Requirement for human user (other than the attacker) to participate in the compromise of the component or system **scope:** whether a vulnerability in one vulnerable component impacts resources in components beyond its security scope **cpe:** the CPE (in formatted string format) of the affected component **base_severity:** The base severity of the vulnerability **confidentiality_impact**: the impact to the confidentiality of the component or system **exploitability_score**: A formula-driven score that represents the exploitability of the vulnerability More information on CVSS-related fields can be found in the CVSS Specification.

Parser Name	Description / Purpose	Fields
database_files	Identify database files within the firmware image. Currently looks for the following known database file extensions: ‘.db’, ‘.sqlite’, ‘.sqlite3’ Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	**database_file:** The full path of the database file that was identified
group	Parses data from /etc/groups to help identify potential account configuration issues. Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	**additional_users:** A list of users present in the group **cracked:** Shows as True if the group has a password that was cracked **group_id:** The ID number of the group on the system **groupname:** The name of the group **hash:** The hash of the group’s password **hash_type:** The type of hash detected for the group’s password **password:** The plaintext value of the group’s password hash **plaintext:**
hash	Search for password hashes within an asset. Scans all files. Supports the following hash formats: MD5, Bcrypt, SHA-256, SHA-512	**cracked:** Shows as “true” if the plaintext of the hash has been identified **hash:** The hash that was identified **hashcatNumber:** The hashcat mode used to crack the password **length:** The length of the password hash **numRounds:** The number of rounds of the hash algorithm used to generate the password **offset:** The offset in the file where the hash was found **plaintext:** The plaintext value of the hash **type:** The hash algorithm used to generate the hash

inittab	Parses values found in /etc/inittab to help identify activities that ocurr when a system starts up. Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	**action:** The action associated with a process that init will take. **id:** An identifier used to reference the process being actioned by init **process:** The process or file being executed by init **runlevels:** The runlevel of the process.

Parser Name	Description / Purpose	Fields
kernelconfig	Enumerates kernel configuration flags to identify risks in how the kernel has been configured or compiled. Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	**key:** The configuration item **value:** The value set for the configuration item
kmodinfo	Identifies kernel modules in each asset and parses relevant information for component identification purposes. Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	**description:** The description of the kernel module provided by the developer **license:** The software license associated with the kernel module **vermagic:** The version of the kernel module **alias:** Aliased names for the kernel module that can be used to access it **depends:** Other kernel modules or capabilities that the kernel module depends on **intree:** Indicates if the kernel module is in a dependency tree **retpoline:** Identifies if the kernel module was compiled with retpoline which helps prevent various types of exploitation.

Parser Name	Description / Purpose	Fields
linuxauth	An aggregate parser that combines data from “passwd” and “shadow” files into a single record for each user. Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	**last_change:** the last time the user’s password was changed **home_dir:** the home directory of the user **groupname:** the name of the group associated with the user **cracked:** if the plaintext of the user’s password has been recovered **warn: t**he period (in days) before the user is warned about their password expiration **password:** the plaintext password (or hash) present in the “passwd” or “shadow” file **inactive:** the number of days after password expires that account is disabled **group_id:** the id of the group associated with the user **shell:** the user’s default shell **user_id:** the unique numerical ID of the user on the system **user_info:** a description or notes for the user **expire:** the date of expiration of the account, expressed as the number of days since Jan 1, 1970 **maximum:** the maximum number of days the password is valid, after that user is forced to change her password again **minimum:** the minimum number of days required between password changes **hash:** the hash of the user’s password **hash_type:** the hashing algorithm used to generate the user’s password hash **username:** the username of the user
malware	Run a malware scan against the extracted filesystem. These entries should be investigated closely.	**signature:** The “friendly” name or signature of the malware identified

Parser Name	Description / Purpose	Fields
passwd	Search for data that looks like Linux passwd file data. Used to identify potential credentials and other useful information about users. Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	**username:** The username **password:** The password (if present) **group_id:** The GID of the user **user_info:** A short text field containing information about the user account **shell:** The default shell for the user **home_dir:** The home directory of the user **user_id:** The UID of the user
publickey	Identifies cryptographic public keys in standalone files as well as embedded within other files. Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	**private: a** true/false value determining if this key is a private key. **uniq_hash:** the hash of various cryptographic items which is internally used to uniquely identify this certificate. **key_strength:** an evaluation of how cryptographically secure this certificate is. **e:** the cryptographic entropy of the certificate **bit_size:** the key size of the certificate **n:** one of the cryptographic constants used to generate the certificate **file_offset:** the offset in the file where this certificate was identified **algo:** The algorithm used with this certificate **match_hash:** the hash of various cryptographic items which is internally used to match this certificate with its private key

Parser Name	Description / Purpose	Fields
privatekey	Identifies cryptographic private keys in standalone files as well as embedded within other files. Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	**private: a** true/false value determining if this key is a private key. **uniq_hash:** the hash of various cryptographic items which is internally used to uniquely identify this private key. **key_strength:** an evaluation of how cryptographically secure this certificate is. **e:** the cryptographic entropy of the private key **bit_size:** the key size of the private key **n:** one of the cryptographic constants used to generate the private key **file_offset:** the offset in the file where this certificate was identified **algo:** The algorithm used with this private key **match_hash:** the hash of various cryptographic items which is internally used to match this private key with any compatible public keys
publicprivatekeypairs	Identifies public and private keypair hashes. This information is typically used in conjunction with the *publickey* and *privatekey* parser data to be useful. Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	**privkey_hash:** The hash of the private key that works with the public key. **pubkey_hash:** The hash of the public key that works with the private key.
services	Parsers system services to identify installed applications and tasks that may be run on an automatic basis. Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	**exec_start:** the program or script to run with its arguments **description:** a description of the service

Parser Name	Description / Purpose	Fields
shadow	Search for data that looks like Linux shadow file data. Used to identify potential credentials and other useful information about users. Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	**username:** The username **hash_type:** The type of the password hash **minimum:** The minimum number of days required between password changes **maximum:** The maximum number of days the password is valid **expire:** days since Jan 1, 1970 that account is disabled i.e. an absolute date specifying when the login may no longer be used **last_change**: Days since Jan 1, 1970 that password was last changed **inactive:** The number of days after password expires that account is disabled **password:** The password (encrypted) for the user **warn:** The number of days before password is set to expire that the user is warned that their password must be changed
shebang	Searches files for “#!” (she-bang) which typically indicates the application that should be used to execute a file. Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	**context:** The string that matched the search parameters **exec**: The application or file that should be used to execute the file
soname	Finds shared objects by searching for files with “.so” in their name Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	**arch:** a numerical enum which is used for internal purposes **bitness:** a numerical enum which is used for internal purposes **endianness**: a numerical enum which is used for internal purposes **file_md5:** the MD5 hash of the identified file **soname:** the name of the identified file

Parser Name	Description / Purpose	Fields
sshpukey	Identifies SSH public keys on the system and parses their attributes for easy searching. Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	**algo:** The algorithm of the key. **file_offest:** The offset in the file where the key was found. **key_strength:** The relative strength of the key (weak, adequate, strong). **curve:** For certain algorithms, display the value of the curve used to calculate the key. **x:** For certain algorithms, display the x value of the curve. **y:** For certain algorithms, display the y value of the curve. **e:** For certain algorithms, display the entropy of the key. **n:** For certain algorithms, display one of the cryptographic constants used to generate the certificate **bit_size:** For certain algorithms, display the size of the key **KeyString:** Display the string of the key **User:** Display the user associated with the key **Value:** Display the full value of the key
sysconfig	Parses files in /etc/sysconfig/ to determine default system and service configurations Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	**fname:** The name of the file analyzed **key:** the configuration option **value:** the value of the configuration option
sysctlconf	Analyzes sysctl configurations to determine what services and configuration a system will boot with Note that the presence of this data is not necessarily an indicator or risk or maliciousness.	**key:** the name of the configuration option **value:** the value of the configuration option

Appendix A: NetRise Contacts

Support

For any technical issues or concerns, please visit the Knowledge Base at kb.netrise.io or reach out to NetRise support via email at support@netrise.io.

Appendix B: Glossary of X.509 Certificate Invalidities

For more information about any of the subsequent fields, or on X.509 certificates in general, please reference additional documentation here.

CA Not authed for EXTKEY usage: When an intermediate or root certificate does not permit a requested extended key usage.

CA Not authorized for this name: When an intermediate or root certificate has a name constraint which doesn't permit a DNS or other name in the leaf certificate.

CA Self signature invalid: The certificate identifies itself as a certificate authority (which should be self signed) but has an invalid signature.

Certificate invalid: When the invalidity type is not supported by our engine, but has still be determined to be invalid.

Constraint violation: The certificate is attempting to be used in a way it is not allowed to.

Expired: The certificate is past its expiration date.

Hostname error: When the set of authorized hostnames does not match the requested one.

Incompatible usage: When the certificate's key usage indicates that it may only be used for a different purpose.

Name constraints without SANS: When a leaf certificate does not contain a Subject Alternative Name extension, but a CA certificate contains name constraints, and the Common Name can be interpreted as a hostname.

Name mismatch: When the subject name of a parent certificate does not match the issuer name in the child.

Not authorized to sign: When a certificate is signed by another which isn't marked as a CA.

Private key compromised: Someone, which may include us, has compromised the private key, meaning it is insecure.

Revoked: AA Compromise: The AA validated in the attribute certificate is known or suspected to have been compromised.

Revoked: Affiliation Changed: A CRL or OCSP request indicated that the certificate no longer is associated with that CA.

Revoked: CA Compromised: A CRL or OCSP request indicated that the certificate authority was somehow compromised.

Revoked: Certificate Hold: The CA indicates that it will not vouch for this certificate. A more precise reason may be provided by the CA in the future.

Revoked: Cessation of Operation: The CA has been decommissioned and is no longer used.

Revoked: Key Compromised: A CRL or OCSP request identified the private key associated has been compromised, making the certificate insecure.

Revoked: Privilege Withdrawn: The certificate was revoked because a privilege contained within the certificate has been withdrawn.

Revoked: Remove from CRL: If a certificate is unrevoked after a Certificate Hold reason is issued, it will still be listed in the CRL. This reason indicates that the certificate should be removed from the CRL.

Revoked: Superseded: A replacement certificate has been issued for reasons unrelated to expiration, compromise, or revocation.

Revoked: Unknown Reason Value: The certificate has been revoked but does not have a reason conforming to RFC 5280.

Revoked: Unspecified: A CRL or OCSP request identified the certificate as revoked but did not provide a reason.

Too many constraints: When the number of comparison operations needed to check a certificate is excessive.

Too many intermediates: When a path length constraint is violated.

Unconstrained name: When a CA certificate contains permitted name constraints, but leaf certificate contains a name of an unsupported or unconstrained type.

Unknown authority: When the certificate issuer is unknown or the issuing certificate can not be acquired.

Appendix C: Detailed Permissions Matrix

	Owner	Operator	Member
Dashboard			
View Dashboard	Yes	Yes	Yes
Assets			
View Asset List	Yes	Yes	Yes
Upload Assets	Yes	Yes	No
Edit Name Vendor, Product, Version fields	Yes	Yes	No
View Analysis Log	Yes	Yes	Yes
View Asset Report (all tabs)	Yes	Yes	Yes
Download Asset	Yes	Yes	Yes
Download Extracted Asset	Yes	Yes	Yes
Download SBOM	Yes	Yes	Yes
Set Remediations	Yes	Yes	No
Asset Groups			
Create Asset Group	Yes	Yes	No
Delete Asset Group	Yes	Yes	No
Edit Asset Group	Yes	Yes	No
Add Asset to Asset Group	Yes	Yes	No
Remove Asset to Asset Group	Yes	Yes	No
Search			
Run Artifact Search Query	Yes	Yes	Yes
Run Trace Search Query	Yes	Yes	Yes

User Management			
View User Management Page	Yes	Yes	Yes
Invite User to Organization	Yes	No	No
Promote User Role	Yes	No	No
Demote User Role	Yes	No	No
Remove User	Yes	No	No
Disable User	Yes	No	No
Delete User	Yes	No	No
Reset User’s Password	Yes	No	No
Profile			
Update Name	Yes	Yes	Yes
Change Theme	Yes	Yes	Yes
Update Password	Yes	Yes	Yes

Appendix D: Glossary

Platform-Specific Terms

NRP: Short for NetRise Priority Rating. NetRise Priority Rating (NPR) is a dynamic metric updated every 8 hours that represents a vulnerability's likelihood of being exploited and its severity. Rather than relying only on a vulnerability's CVSS score to determine priority, NPR takes a combination of the following data points into account:

- CVSS Impact and Exploitability scores
- Exploit Prediction Scoring System (EPSS) scores
- If the vulnerability has known exploit code available
- If the vulnerability is known to be leveraged by ransomware groups, botnets, named threat actors, or has been otherwise found in-the-wild

These features are coalesced to create a normalized score which is represented between 0 and 10. 0 representing the lowest risk and 10 representing the highest risk.

Proof of Concept: In the context of exploit availability for a particular CVE, a Proof-of-Concept value indicates that at least one exploit exists but has not been observed as weaponized by malicious actors.

Tag: A unique string that can be used to mark files or other data points. Example use cases may be to mark similar files with the same tag, or to mark files for further analysis or review.

Unproven: In the context of exploit availability for a particular CVE, an Unproven value indicates that no exploit has been identified for the vulnerability.

Weaponized: In the context of exploit availability for a particular CVE, a Weaponized value indicates that an exploit exists and has been leveraged in ransomware campaigns or botnets, is associated with known threat actors or named attacks, or exist as part of an exploit framework (e.g. Metasploit)

General Terms

CISA: Short for the Cybersecurity and Infrastructure Security Agency. A United States Government agency responsible for strengthening the cybersecurity and infrastructure protection across all levels of the government.

CPE: Short for Common Platform Enumeration. CPE is a structured naming scheme for information technology systems, software, and packages. Based upon the generic syntax for Uniform Resource Identifiers (URI), CPE includes a formal name format, a method for checking names against a system, and a description format for binding text and tests to a name. CPEs are most commonly used for uniquely identifying a piece of software or device and looking up vulnerabilities for it. More information can be found here: https://nvd.nist.gov/products/cpe.

CVE: Short for Common Vulnerabilities and Exposures. A CVE is one instance of a vulnerability that has been documented and assigned a CVE ID number by the MITRE Corporation. More information can be found here: https://cve.mitre.org/

CVSS: Short for Common Vulnerability Score System. The Common Vulnerability Scoring System (CVSS) provides a way to capture the principal characteristics of a vulnerability and produce a numerical score reflecting its severity. The numerical score can then be translated into a qualitative representation (such as low, medium, high, and critical) to help organizations properly assess and prioritize their vulnerability management processes. More information can be found here: https://www.first.org/cvss/.

CWE: Short for Common Weakness Enumeration. CWE is a community-developed list of common software and hardware weakness types that have security ramifications. A "weakness" is a condition in a software, firmware, hardware, or service component that, under certain circumstances, could contribute to the introduction of vulnerabilities. The CWE List and associated classification taxonomy serve as a language that can be used to identify and describe these weaknesses in terms of CWEs. More information can be found at https://cwe.mitre.org/index.html

EPSS: Short for Exploit Prediction Score System. The Exploit Prediction Scoring System (EPSS) is a data-driven effort for estimating the likelihood (probability) that a software vulnerability will be exploited in the wild. While other industry standards have been useful for capturing innate characteristics of a vulnerability and provide measures of severity, they are limited in their ability to assess threats. EPSS fills that gap because it uses current threat information from CVE and real-world exploit data. The EPSS model produces a probability score between 0 and 1 (0 and 100%). The higher the score, the greater the probability that a vulnerability will be exploited. More information can be found at https://www.first.org/epss/.

KEV: Short for Known Exploited Vulnerability. This is commonly used as a shorthand way to reference CISA's Known Exploited Vulnerability Catalog. More information about the KEV can be found here: https://www.cisa.gov/known-exploited-vulnerabilities-catalog.

NVD: Short for National Vulnerability Database. The NVD is the United States Government of standards-based vulnerability management data sponsored by the National Institute of Standards (NIST). NVD includes databases of security checklists, security related software flaws, misconfigurations, product names, and impact metrics. It is the mechanism that allows CVEs to exist at scale. More information can be found here: https://nvd.nist.gov/.

PURL: Short for Package URL. A purl or package URL is an attempt to standardize existing approaches to reliably identify and locate software packages. A purl is a URL string used to identify and locate a software package in a mostly universal and uniform way across programming languages, package managers, packaging conventions, tools, APIs and databases. Such a package URL is useful to reliably reference the same software package using a simple and expressive syntax and conventions based on familiar URLs. More information can be found here: https://github.com/package-url/purl-spec.

SBOM: Short for Software Bill of Materials. A Software Bill of Materials (SBOM) is a nested inventory for software, a list of ingredients that make up software components. SBOMs are quickly becoming one of the most foundational parts of software transparency and supply chain security.