How Vulnerability Management Works

Wiz detects vulnerabilities in your code and cloud resources, surfaces Vulnerability Findings on the Security Graph, and then helps you remediate them.

For step-by-step walkthroughs of key scenarios related to vulnerability assessment, see the vulnerabilities scenarios, and for patch management, see the patch management scenarios.

You can watch our webinar on vulnerabilities.

How it works

Vulnerability management in Wiz consists of three stages:

- Detection
- Usage and reporting
- Patch Management

Wiz facilitates vulnerability management in three main ways:

1. Detection
   i. During the codes phases, code libraries are detected via Wiz CLI and VCS Connectors.
   ii. During the runtime phase, hosted technologies and their versions are identified using a variety of detection methods via the Workload Scanner.
   iii. This data is combined with information from third-party vulnerability sources and feeds to determine which code libraries/technologies are susceptible to which vulnerabilities.

The Vulnerability Scanner can be configured to suit your specific needs in numerous ways, such as default ignored paths.

2. Usage and Reporting
   - Basic Concepts:

        - Finding
            - Severity
            - Vulnerability Findings are enriched with several useful properties such as `First Seen`, `Last Seen`, `Fixed Version`, `Fix Date`, and `Validated in runtime`
        - Vulnerability
            - Severity
    - Using the Vulnerability Findings page
    - Export at scale using the Vulnerability Findings page
    - Attack path detection using the Security Graph:
        - Model
        - Why not all the vulnerabilities are within the Graph?
        - Do not export from the Graph at scale
3. Patch Management
    - Vulnerability Findings and Hosted Technology objects help you perform patch management and remediate vulnerabilities.
    - Properties such as `Release Date` and `EOL Date` are supported for many common technologies.
    - After a technology is patched, Wiz automatically resolves any related Vulnerability Findings.

Detection

Wiz applies different detection methods during workload scanning and code scanning to determine whether a piece of code or a workload component is susceptible to a vulnerability.

Detection methods

Each Vulnerability Finding has a detection method attached to it with the details of how it was detected:

Code library detection - code scanning

There is a difference in what Wiz scans between code and runtime phases. In the code phases, vulnerabilities are detected in code libraries via Wiz CLI and version control Connectors, while in runtime they are detected in hosted technologies via workload scanning.

Wiz detects vulnerabilities in code libraries that reside in version control systems, CI/CD platforms, or locally in your code. This includes extracting dependencies and nested dependencies (for lock files) of known libraries, including their versions.

Once the libraries and their versions are extracted, the Vulnerability Scanner matches this information with various sources and advisories (the Wiz Knowledge Base, PyUp, PHP Security Advisories, etc.).

The following programming languages are supported, each with its own detection method:

Language	Package Manager	Method	Data sources
.NET/C#	Nuget	The Vulnerability Scanner finds `packages.config` files and then extracts all direct and nested dependencies, including their versions.	Github Advisory Database
Golang	Go	The Vulnerability Scanner finds `go.mod` files and extracts dependencies' names and versions.	GitHub Advisory Database, Go Vulnerability Database, Wiz Knowledge Base
Java	Gradle, Maven	1. The Vulnerability Scanner finds `gradle.lock` and `pom.xml` files. 2. The files are analyzed to extract every dependency being used and its version.	Maven repository
Java Script/ NodeJS	NPM, YARN, PNPM	The Vulnerability Scanner finds `package-lock.json` , `yarn.lock` , and `pnpm-lock.yaml` files and then extracts all direct and nested dependencies, including their versions.	GitHub Advisory Database, Ecosystem Security Working Group, Wiz Knowledge Base
PHP	Composer	The Vulnerability Scanner finds `composer.lock` files, parses them, and then extracts all direct and nested dependencies, including their versions.	GitHub Advisory Database, FriendsOfPHP, Wiz Knowledge Base
Python	pip, pipenv, Poetry	The Vulnerability Scanner finds `requirements.txt` , `poetry.lock` , and `Pipfile.lock` files, then	GitHub Advisory Database, Python Safety DB, PySec, Wiz

Language	Package Manager	Method	Data sources
		extracts all direct and nested dependencies, including their versions.	Knowledge Base
Ruby	Bundler	The Vulnerability Scanner finds `Gemfile.lock` files, parses them, and extracts all direct and nested dependencies, including their versions.	GitHub Advisory Database, RubySec, Wiz Knowledge Base
Rust	Cargo	The Vulnerability Scanner finds `Cargo.lock` files, parses them, and extracts all direct and nested dependencies, including their versions.	RustSec Advisory Database, Wiz Knowledge Base

Code library detection - Runtime

There is a difference in what Wiz scans between code and runtime phases. In the code phases, vulnerabilities are detected in code libraries via Wiz CLI and version control Connectors, while in runtime they are detected in hosted technologies via workload scanning.

Wiz detects vulnerabilities in code libraries that reside within workloads. This includes extracting dependencies and nested dependencies of known libraries, including their versions. Once the libraries and their versions are extracted, the Vulnerability Scanner matches this information with various sources and advisories (the Wiz Knowledge Base, PyUp, PHP Security Advisories, etc.).

Some library detections, e.g. Log4Shell, use unique detection flows with more conditions than just version and library.

The following programming languages are supported, each with its own detection method:

Language	Method	Data sources	Runtime validation support[1]
.NET	The Vulnerability Scanner finds .NET executables and analyzes the binary files to extract direct and nested dependencies.	GitHub Advisory Database, Wiz Knowledge Base	▯
GO	The Vulnerability Scanner finds `go.mod` files and extracts dependencies' names and versions. Then, it finds Go binaries and extracts dependencies directly.	GitHub Advisory Database, Go Vulnerability Database, Wiz Knowledge Base	▯
Java	1. The Vulnerability Scanner finds `gradle.lock` files, and also searches for all JAR, WAR, EAR, and PAR files and unarchives them to detect their `pom.xml` files. 2. The `pom.xml` and `gradle.lock` files are analyzed to extract every dependency being used and its version. 3. The Java JAR is assessed as a standalone library by using its hash to extract the JAR name and version.[2] 4. This process is repeated recursively for every nested JAR detected.	Maven repository	▯
Scala	1. The Vulnerability Scanner searches for all JAR, WAR, EAR, and PAR files and unarchives them to detect their `pom.xml` files.	GitHub Advisory Database, Wiz Knowledge Base	▯

Language	Method	Data sources	Runtime validation support[1]
	2. The `pom.xml` file is analyzed to extract every dependency being used and its version. 3. The Scala JAR is assessed as a standalone library by using its hash to extract the JAR name and version. 4. This process is repeated recursively for every nested JAR detected.		
NodeJS	1. The Vulnerability Scanner finds `package-lock.json` files, then extracts all direct and nested dependencies, including their versions. 2. The Vulnerability Scanner finds all `yarn.lock` files, then extracts all direct and nested dependencies, including their versions. 3. The Vulnerability Scanner finds all `pnpm-lock.yaml` files, then extracts all direct and nested dependencies, including their versions.	GitHub Advisory Database, Ecosystem Security Working Group, Wiz Knowledge Base	▯
PHP	The Vulnerability Scanner finds `composer.lock` files, parses them, and extracts all direct and nested dependencies, including their versions.	GitHub Advisory Database, FriendsOfPHP, Wiz Knowledge Base	▯
Python	1. The Vulnerability Scanner finds `Pipfile.lock` files, then	GitHub Advisory Database, Python Safety	▯

Language	Method	Data sources	Runtime validation support[1]
	extracts all direct and nested dependencies, including their versions. 2. The Vulnerability Scanner finds `poetry.lock` files, then extracts all direct and nested dependencies, including their versions. 3. The Vulnerability Scanner finds `requirements.txt` files, then extracts all direct and nested dependencies, including their versions.	DB, PySec, Wiz Knowledge Base	
Ruby	The Vulnerability Scanner finds `Gemfile.lock` files, parses them, and extracts all direct and nested dependencies, including their versions.	GitHub Advisory Database, RubySec, Wiz Knowledge Base	▯
Rust	The Vulnerability Scanner finds `Cargo.lock` files, parses them, and extracts all direct and nested dependencies, including their versions.	RustSec Advisory Database, Wiz Knowledge Base	▯

[1] The Wiz Runtime Sensor can identify vulnerabilities executed in runtime for some code libraries. Learn more about vulnerability runtime validation.

[2] Wiz performs hash-based vulnerability detection for all Java archive files. In rare cases, the JAR's manifest does not match the actual name of the JAR, which is used by Wiz to determine the name of the JAR. By using the hash-based detection method, Wiz is able to compare the hash to public Maven repository data to determine the exact name and version of that JAR, and subsequently find vulnerabilities related to that library.

Package detection

Wiz detects vulnerabilities in packages installed on a system using the OS package manager. Wiz uses the package manager installation paths for different distributions to find the versions of installed packages. The following package managers are supported:

OS package manager	System
DNF/RPM/YUM	RedHat/Fedora based Linux distributions
Alpine Package Keeper (apk)	Alpine/Wolfi/Chainguard Linux distributions
APT	Ubuntu/Debian Linux distributions
Zypper	openSUSE/SUSE Linux distributions
Portage	Gentoo Linux distributions
Nix	NixOS Linux distributions
Homebrew	MacOS distributions

Once lists of packages and their versions are obtained, the Vulnerability Scanner uses vendor vulnerability streams to generate vulnerabilities specific to the operating system version and distribution. In the Finding's description, Wiz shows which package manager was used to detect the package, and the operating system the machine is running. This allows Wiz to make sure that remediation recommendations and the `Fixed Version` and `Fixed Date` properties are correct and accurate to that specific operating system version.

For example, a package installed using APT on a machine running Ubuntu 20.04 shows only vulnerabilities relevant to the Ubuntu 20.04 stream.

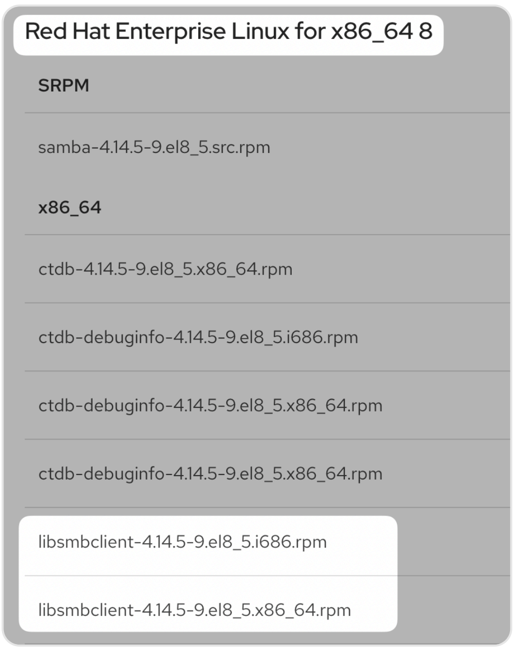
⚠ If a fix is available for the Ubuntu 22.04 stream but not available for the Ubuntu 20.04, Wiz does not show the remediation as the version from the Ubuntu 22.04. This would be a cross-stream/cross-version upgrade which cannot be recommended by Wiz.

However, for EOL Linux operating Systems, the oldest supported OS version is listed as the suggested `Fixed Version` in cases where there is no update available on the EOL/EUS/E4S package stream.

Red Hat EUS/E4S backport detection

The Vulnerability Scanner detects fixed vulnerabilities when the installed package is a backported version unique to specific support plans (EUS, E4S).

For example, a fix for CVE-2011-44142 is available for Red Hat Enterprise Linux 8 in `libsmbclient-4.14.5-9.el8_5` and above:



Red Hat Enterprise Linux for x86_64 8

SRPM

samba-4.14.5-9.el8_5.src.rpm

x86_64

ctdb-4.14.5-9.el8_5.x86_64.rpm

ctdb-debuginfo-4.14.5-9.el8_5.i686.rpm

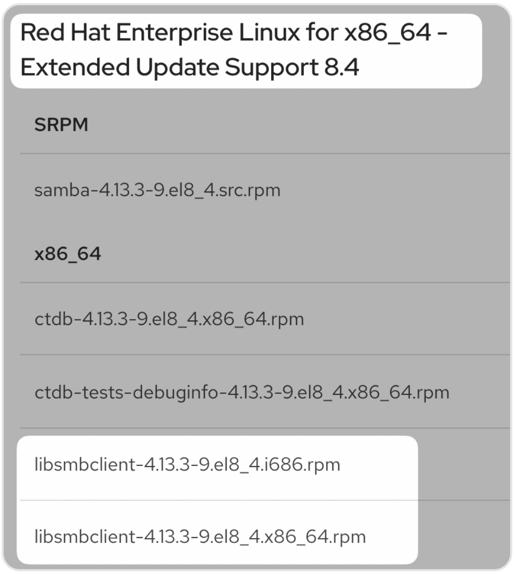
ctdb-debuginfo-4.14.5-9.el8_5.x86_64.rpm

ctdb-debuginfo-4.14.5-9.el8_5.x86_64.rpm

libsmbclient-4.14.5-9.el8_5.i686.rpm

libsmbclient-4.14.5-9.el8_5.x86_64.rpm

However, in Red Hat EUS 8.4, a backported package is available, `libsmbclient-4.13.3-9.el8_4`, even though the version number is lower than Red Hat Enterprise Linux:



Red Hat Enterprise Linux for x86_64 - Extended Update Support 8.4

SRPM

samba-4.13.3-9.el8_4.src.rpm

x86_64

ctdb-4.13.3-9.el8_4.x86_64.rpm

ctdb-tests-debuginfo-4.13.3-9.el8_4.x86_64.rpm

libsmbclient-4.13.3-9.el8_4.i686.rpm

libsmbclient-4.13.3-9.el8_4.x86_64.rpm

Taking these into account, the Vulnerability Scanner does not create a CVE-2011-44142 Vulnerability Finding for `libsmbclient-4.13.3-9.el8_4`, as this package is fixed.

Alpine vulnerabilities

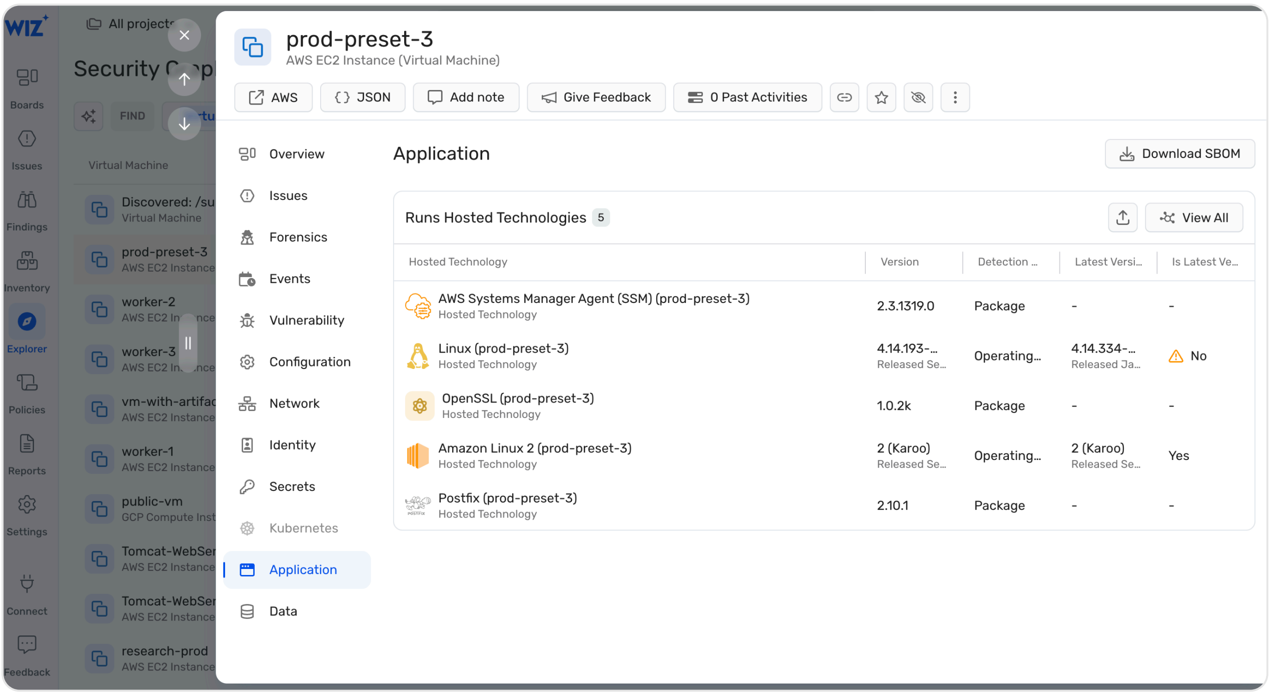
Wiz shows vulnerabilities found on apk-installed packages on Alpine OS containers/servers, regardless of whether or not the affected package is included in an Alpine security advisory. Wiz checks if the installed package is vulnerable based on NVD-affected versions and other upstream sources. However, in cases where the Alpine repository stream for the container/server's OS version does not contain a version of the package that users can upgrade to remediate the vulnerability, they may have to upgrade the container/server's OS version instead. In such cases, Wiz displays the Vulnerability Finding but does not show a "Fixed version" for them.

File path detection

Wiz detects vulnerabilities using only the file system metadata. There are three main detections that are categorized as file path detection:

CPE and hosted technology detection

Parsing each file on the file system, Wiz identifies hosted technologies based on a file name and file contents, then extracts the version by looking for license files and manifests used by that software. This surfaces on the workload a Hosted Technology object on the Security Graph, viewable in the Application tab of the workload's details drawer.



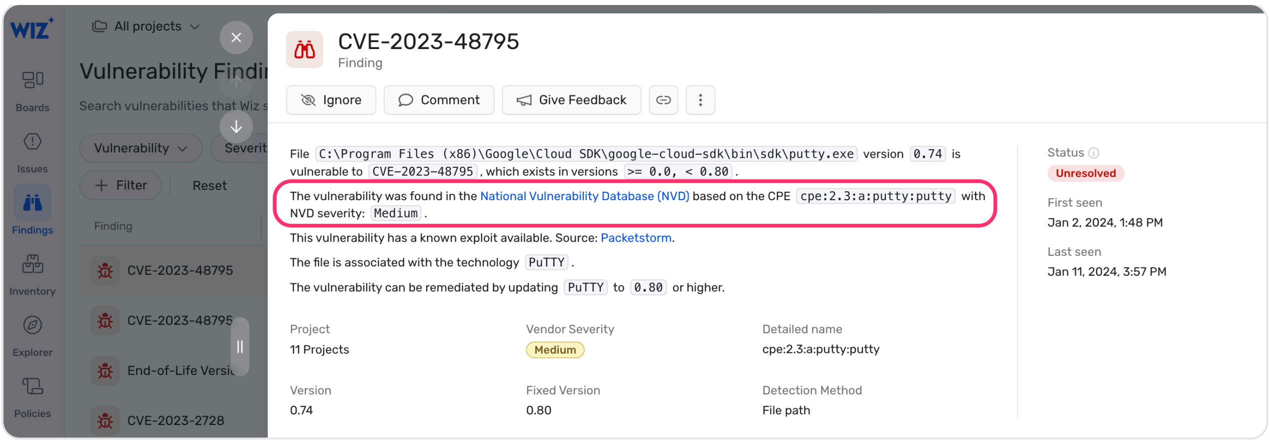
Application

Runs Hosted Technologies 5

Hosted Technology	Version	Detection ...	Latest Versi...	Is Latest Ve...
AWS Systems Manager Agent (SSM) (prod-preset-3) Hosted Technology	2.3.1319.0	Package	-	-
Linux (prod-preset-3) Hosted Technology	4.14.193-... Released Se...	Operating...	4.14.334-... Released Ja...	⚠ No
OpenSSL (prod-preset-3) Hosted Technology	1.0.2k	Package	-	-
Amazon Linux 2 (prod-preset-3) Hosted Technology	2 (Karoo) Released Se...	Operating...	2 (Karoo) Released Se...	Yes
Postfix (prod-preset-3) Hosted Technology	2.10.1	Package	-	-

CVEs published by the National Vulnerability Database (NVD) specify affected software configuration in the form of Common Platform Enumeration (CPE) names for all CVEs, which is a standardized method of identifying software components and their versions. Using the hosted technologies identified earlier, a corresponding CPE is associated. The CPE is then used to match the technology detected with vulnerabilities sourced from NVD. CPE-detected vulnerabilities are sourced not only from NVD but from open-source community repositories as well.

For example, in the image below, PuTTY was determined to be installed on a VM via the file path `C:\Program Files (x86)\Google\Cloud SDK\google-cloud-sdk\bin\sdk\putty.exe`. Note that the vulnerability was found because it matches the CPE name in the CVE, `cpe:2.3:a:putty:putty`.



CVE-2023-48795
Finding

File `C:\Program Files (x86)\Google\Cloud SDK\google-cloud-sdk\bin\sdk\putty.exe` version `0.74` is vulnerable to `CVE-2023-48795`, which exists in versions `>= 0.0, < 0.80`.

The vulnerability was found in the National Vulnerability Database (NVD) based on the CPE `cpe:2.3:a:putty:putty` with NVD severity: `Medium`.

This vulnerability has a known exploit available. Source: Packetstorm.

The file is associated with the technology `PuTTY`.

The vulnerability can be remediated by updating `PuTTY` to `0.80` or higher.

Project	Vendor Severity	Detailed name
11 Projects	Medium	cpe:2.3:a:putty:putty
Version	Fixed Version	Detection Method
0.74	0.80	File path

Status
Unresolved

First seen
Jan 2, 2024, 1:48 PM

Last seen
Jan 11, 2024, 3:57 PM

Wordpress plugin and themes detection

Wordpress plugins and themes are commonly targeted by attackers, exploiting vulnerabilities in those installations. Wiz finds installed Wordpress plugins and themes by file path and surfaces vulnerabilities related to them.

Jenkins plugin detection

Jenkins, an open-source CI/CD automation tool, allows for third-party plugins and modules to be downloaded to enhance the functionality of the tool. Wiz is able to detect these plugins via file path and surface vulnerabilities related to them.

Windows installed-programs detection

Wiz detects vulnerabilities for all programs installed on Windows machines. These vulnerabilities are divided into two:

Microsoft product files

Software shipped by and maintained by Microsoft requires updates if vulnerabilities are found. This is usually handled by installing a specific Windows Knowledge Base (KB) update, which patches a series of Microsoft services, frameworks, and tools on the Windows machine. These KB updates can be security updates or major updates.

For the remediation and fixed version, Wiz always recommends the latest available KB for that specific Windows family and version. However, Wiz also checks if the found software that is affected has been updated as well, even if the recommended KB is installed. If either the KB or the affected software files have not been updated, then Wiz will leave the Finding unresolved.

Third-party product files and drivers

User-installed software on Microsoft servers may be affected by vulnerabilities as well. Wiz detects installed programs using a range of methods, including product files and binaries (.dll), Windows registry keys and their values, and installed executables (.exe).

Wiz checks each vulnerability against a set of criteria to determine if the affected program is vulnerable, such as inspecting registry key state, service configurations, and even file permissions. As these updates aren't fixable by a KB update, Wiz provides a fix version that is relevant to the installed program instead.

Operating System

Wiz detects vulnerabilities affecting operating system components and default packages. These vulnerabilities include the following:

Linux kernel

A critical component of the Linux operating system is the system kernel. The Linux kernel being open-sourced and widely used makes it susceptible to frequent hacking and exploits. Wiz can identify the running kernel version on any Linux VM and detect vulnerabilities specific to the Linux operating system it was sourced from. As with package detection, Wiz will only show the fixed version of the Linux kernel available for that specific operating system version and family. For example, a kernel with a vulnerability affecting an Amazon 2 machine will not be recommended kernel version specific to RHEL, Debian, etc.

Windows Knowledge Base (KB)

Windows services and drivers installed by default on Windows machines often require KB updates to resolve vulnerabilities. As opposed to the Windows installed-programs detection method, where Wiz checks file versions of the program, the operating system detection only checks the Windows KB version and determines if the Windows machine has the fix applied.

Operating system-specific vulnerabilities

Some operating systems ship with default packages and software that can only be fixed by updating the image or patching the operating system. Operating system vulnerabilities are disclosed by the operating system maintainer or vendor themselves. Using operating system-specific advisories, Wiz determines if the operating system version running on the server has vulnerabilities and can be remediated by upgrading to a higher OS version.

Operating systems that Wiz finds vulnerabilities for are:

- Bottlerocket OS
- Container-optimized OS (COS)

- macOS

Runtime scan process

Wiz scans the metadata using the detection methods. The scanning process depends on the resource:

Resource	Scanning process
Virtual Machines (e.g., AWS EC2 instance, GCP compute instance) Hosted Containers (e.g., AWS ECS container, Azure ACA container) Hosted Container Images (e.g., AWS ECR image, Azure container image)	1. Listing all VMs, containers, and container images using cloud service provider API calls. 2. Creating a snapshot of the OS volume (or non-OS disk volume, if this setting is enabled). 3. Analyzing the snapshots of the VMs/containers, including the container images and the writable layers for BOM. 4. Generating Vulnerability Findings based on detected versions in BOM.
Serverless Containers (e.g., AWS ECS Fargate, Google Cloud Run, Azure Container App)	1. Listing all container services using cloud service provider API calls. 2. For each container service, listing containers running and their associated container images. 3. Pulling container images from sourced registry (Docker Hub, ECR, GAR, ACR) to the Vulnerability Scanner. 4. Analyzing the container images for BOM. 5. Generating Vulnerability Findings based on detected versions in BOM.
Serverless Functions[1] (e.g., AWS Lambda, Google Cloud Functions, Azure App Services, Azure Functions)	1. Listing all serverless functions using cloud service provider API calls. 2. For each function, detecting build language and identifying predefined dependency files. 3. Analyzing function files to extract library references and their versions (BOM). 4. Generating Vulnerability Findings based on identified libraries and their detected versions.

> 1 For serverless functions that use a container image, Wiz has a default scan limit to the image size. If you encounter scanning errors related to this, please contact Wiz support.

Usage and Reporting

Wiz surfaces vulnerabilities on the Security Graph and reports.

Security Graph objects

After vulnerability assessment, two different Security Graph objects are created:

- Vulnerability—A cataloged CVE associated with a specific technology (query link)
- Vulnerability Finding—A Wiz-generated enrichment of a vulnerability that takes into consideration the wider context in which it was detected (query link)

For each CVE, Wiz provides you with the unique CVE identifiers and all the related information, such as the release date, severity, impact, attack vector, and complexity. This Security Graph object is called Vulnerability and is sourced directly from our vendor vulnerability streams and the National Vulnerability Database (NVD).

> ⚠ Due to the current substantial backlog of missing vulnerability metadata in NVD, Wiz uses an alternative method in cases of missing data. Learn more about bridging the NVD gap.

When Wiz finds a vulnerability on a workload, it generates a Vulnerability Finding Graph object, which is an instance of that vulnerability for a unique workload. The Finding includes specifics of the affected software, the version of the affected software, fixed version details, remediation recommendations, and more.

Vulnerability Findings on the Security Graph

Not all Vulnerability Findings appear on the Security Graph. Vulnerabilities appear on the Security Graph if they are critical or high severity or CISA KEV exploitable. Vulnerabilities that are neither critical nor high severity and are not CISA KEV exploitable are reported on the Vulnerability Findings page only.

If you would like to list or export all the Vulnerability Findings in your environment (including medium/low and non-CISA KEV exploitable), you can use the Vulnerability Findings page. Learn about exporting vulnerabilities.

Patch management

Vulnerability Findings help you perform patch management by updating your hosted technologies. Specific patch management properties are added to hosted technologies detected on workloads. Using the workload scanning's metadata, Wiz determines hosted technologies installed on a machine. A hosted technology can be an operating system, server application, code framework and library, CI/CD tool, and more. Hosted technologies are detected using the same metadata and detection methods as in vulnerability assessment.

Learn about hosted technologies and see the hosted technology catalog.

Patch recommendations

Wiz offers patch recommendations when it detects multiple CVEs with different fixed versions in code or on a resource. These recommendations are designed to help you quickly identify and patch vulnerable components (OS, packages, and libraries). Each recommendation shows the existing version and the lowest version the component should be patched to in order to remediate its identified vulnerabilities. Additionally, you can find an overview of all Findings related to the specific component, its file path, and actionable instructions for remediation.

Patch recommendations appear in different places depending on the application lifecycle phase:

- For workload scanning during runtime—on the Remediation section in the Vulnerability tab of the resource details drawer.
- For Wiz CLI scans during the CI/CD phase—on the Findings > CI/CD Scans page, in the details drawer of every relevant CI/CD scan.
- For GitHub PR scans during the code phase—on GitHub (include only existing version and fixed version).

Resolving Findings

When Wiz determines that a Vulnerability Finding is not present in your environment anymore, it considers that Finding as resolved. Resolved Vulnerability Findings are retained in a soft-deleted state on the Security Graph for 7 days, during which:

- They are no longer displayed on the Security Graph.
- You can search for them on the Findings > Vulnerability Findings page by adding the Resolved filter.
- You can download a list of resolved Findings in CSV format by adding the Resolved filter. You can also export resolved Findings via the API. Learn about exporting Vulnerability Findings via reports and programmatically via the API.

After 7 days, resolved Vulnerability Findings are hard-deleted and can no longer be viewed anywhere in the portal or API.

Vulnerability Scanner configuration and settings

Scan settings

Wiz provides vulnerability scanning settings that allow you to adjust the process to your needs. Learn about the Vulnerability Scanner settings.

Default ignored paths

There are two cases where Wiz purposely ignores certain paths when analyzing for vulnerabilities:

- Package manager installation folder is not located on an OS disk partition
- (Legacy) Code libraries

Package manager installation folder not on an OS disk partition

If you have non-OS disk scanning enabled, then the folder will be scanned. Otherwise, some vendors recommend to put the `/var/` folder, which is used to store the package manager packages, in a different partition on a different disk. For example, Debian claims in C.3. Recommended Partitioning Scheme, Appendix C. Partitioning for Debian:

> For multi-user systems or systems with lots of disk space, it's best to put /usr, /var, /tmp, and /home each on their own partitions separate from the / partition.

In these cases, the OS disk partition may contain an old version of the `/var/` folder. Wiz does not (currently) scan non-OS disks by default; however, this feature can be enabled. If it's not enabled, when this old version folder gets scanned, vulnerabilities may be detected that are already fixed by the updated packages installation non-OS disk partition of `/var`. Therefore, Wiz detects whether a package manager's files are on a non-OS disk and ignores any package vulnerabilities.

(Legacy) Code libraries

> ⚠ Code library exclusion paths is a legacy feature. All tenants provisioned since May 17th, 2023 do not apply them.
>
> If your tenant was provisioned before then, you may opt to disable code library exclusion paths.

When detecting code library vulnerabilities only in tenants provisioned before May 17th, 2023, some paths and sub-paths are skipped in order to avoid generating an overwhelming number of mostly irrelevant results. You can check if code library exclusion paths are enabled for your tenant using the Vulnerability Scanner settings.

Paths containing the following keywords are ignored on VMs and container images when detecting code library vulnerabilities in tenants provisioned before May 17th, 2023:

```
"/Windows/", "/lib/", "/embedded/", "/sdks/", "/sdk/", "/temp/", "/tmp/",
"/Program Files/", "/Program Files (x86)/",
"/usr/", "/.", "/appdata/", "/ProgramData/", "/workspace/", "/downloads/",
"/$recycle.bin/", "/agent/", "/agents/",
"/appdynamics/", "/plugins/", "/az.storage/", "/opt/", "/microsoft.",
"/Informatica/", "/libs/", "/bin/", "/azagent/",
"/tools/", "/debug", "/work/", "/_work/", "/installation files/", "/caches/",
"/cache/", "backup", "/bk/", "/hadoop/",
"/mnt/", "/kafka/", "/windowsazure/", "/azure", "/jenkins_home/", "test",
"/dev/", "/jboss", "/ThirdParty/", "/pentaho/",
"/jenkins/", "/SumoCollector/", "/build/", "/apache", "/spark", "/confluent",
"/setup/", "/chef-solo/", "/node_modules/",
"/vendor/", "/samples/",
```

Paths containing the following keywords are ignored on serverless functions when detecting code library vulnerabilities in tenants provisioned before May 17th, 2023:

```
"/Windows/", "/lib/", "/embedded/", "/sdks/", "/sdk/", "/temp/", "/tmp/",
"/Program Files/", "/Program Files (x86)/",
"/usr/", "/.", "/appdata/", "/ProgramData/", "/workspace/", "/downloads/",
"/$recycle.bin/", "/agent/", "/agents/",
"/appdynamics/", "/plugins/", "/az.storage/", "/opt/", "/microsoft.",
"/Informatica/", "/libs/", "/bin/", "/azagent/",
"/tools/", "/debug", "/work/", "/_work/", "/installation files/", "/caches/",
"/cache/", "backup", "/bk/", "/hadoop/",
"/mnt/", "/kafka/", "/windowsazure/", "/azure", "/jenkins_home/", "test",
"/dev/", "/jboss", "/ThirdParty/", "/pentaho/",
"/jenkins/", "/SumoCollector/", "/build/", "/apache", "/spark", "/confluent",
"/setup/", "/chef-solo/", "/vendor/",
```

Sub-directories are also skipped, so a code library located in `/foo/` would be checked, but code libraries located in `/foo/Windows/` and `/foo/Windows/blah/` would both be skipped.

Metadata used

Using the workload scanning's metadata, the Vulnerability Scanner is able to determine which vulnerabilities should affect which installed technology. The following metadata is analyzed by the Vulnerability Scanner:

- List of installed packages + versions
- List of programming language libraries + versions

- Operating system information + version
- Names and hashes of files
- (Windows only)Installed programs, services, and KBs + versions

Vulnerability sources

Wiz leverages multiple updated vulnerability databases to keep pace with the ever-changing world of vulnerability assessment. Each scan undergoes public source information fetching, multiple internal research and enrichment phases, and a validation process to finalize each Vulnerability Finding. This provides a unique view of detailed information while keeping false positives to a bare minimum.

Wiz uses the following public sources:

Vulnerability sources

- Alibaba Cloud Linux Security Advisories
- AlmaLinux Security Advisories
- Alpine Security Tracker (Alpine secdb)
- Amazon Linux Security Advisories (1, 2, 2023)
- Apple Security Updates
- CBL-Mariner Vulnerability Database
- CentOS Announce
- Container-Optimized OS Release Notes
- Debian Security Bug Tracker
- Exploit Database
- Fedora Project Security Advisories
- GitHub Security Advisories
- Go Vulnerability Database
- Kubernetes security announcement (AKS, GKE, EKS)
- Microsoft Security Response Center (MSRC)
- Node.js Security Working Group (NSWG)
- National Vulnerability Database (NVD)
- Open Source Vulnerabilities database (OSV)
- Oracle Linux Security Tracker
- OVAL
- Packetstorm
- PHP Security Advisories Database
- Photon Security Advisory
- Python Packaging Advisory Database
- Red Hat Security Advisories[1]
- Rocky Linux Product Errata

- Rubysec Advisory Database
- RustSec Advisory Database
- SUSE Security Tracker
- Ubuntu Security Tracker
- VMware ESXi VIB Patches Release Notes
- VMware Security Advisory
- Wolfi Security Advisory Database
- Wordpress Vulnerability Database

> [1] As of April 22nd, 2024, Wiz holds the Red Hat Vulnerability Scanner Certification. See the Wiz blog to learn about the Wiz and Red Hat partnership.

Wiz updates its Vulnerability Catalog daily and provides out-of-band updates in response to high-priority events (e.g. Patch Tuesday). Furthermore, Wiz scans your entire environment every ~24 hours by default. Therefore, an approximate worst-case timeline for Wiz reporting on a new vulnerability would be:

- Day 1—A new vulnerability is published, Wiz updates its databases internally, and then the updated databases are pushed to customers
- Day 2—A regularly scheduled scan occurs based on the databases from Day 1
- Day 3—Results from the regularly scheduled scan from Day 2 are available in the portal

> If you encounter any vulnerability detection problems, contact Wiz support. Furthermore, if you don't see a vulnerability associated with a resource, but you think it *should* be, please let us know by clicking Share Feedback at the top of the details drawer of the problematic workload (if possible).

Severities of Vulnerabilities and Vulnerability Findings

> Due to the current substantial backlog of missing vulnerability metadata in NVD, Wiz uses an alternative method in cases of missing data. Learn more about bridging the NVD gap.

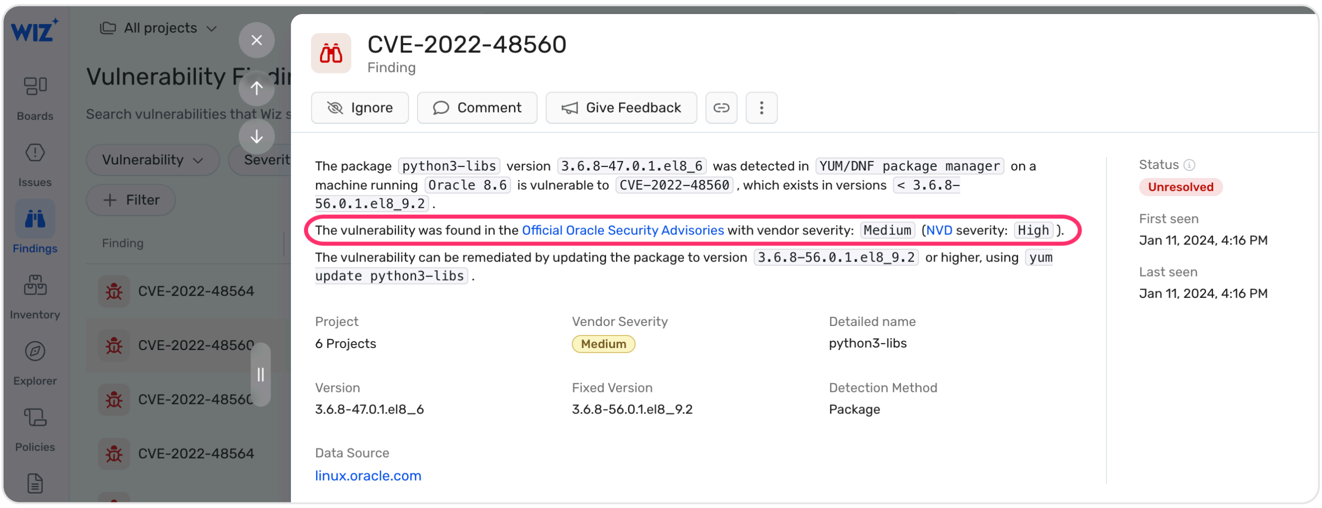
Wiz ingests vulnerability severities from both vendors and the National Vulnerability Database (NVD).

Finding objects on the Security Graph show the vendor severity (if available). Like other objects on the Security Graph, Wiz normalizes the Finding severity between different vendor vulnerability rating scales; this specifically affects Vulnerability Findings from Red Hat Security Advisory, Oracle Linux Errata, CentOS Announce, and Microsoft Security Response Center. See the normalization schema:

Vendor severity	Normalized severity in Wiz
Low	Low
Moderate	Medium
Important	High
Critical	Critical

If there is no vendor advisory or it is not supported by Wiz, the severity of the Finding object is determined by NVD or the Wiz Research team. In contrast, Vulnerability objects on the Security Graph always show NVD severities.

You can compare the severities of the vendor and NVD by inspecting the Finding's description:



CVE-2022-48560
Finding

The package `python3-libs` version `3.6.8-47.0.1.el8_6` was detected in `YUM/DNF package manager` on a machine running `Oracle 8.6` is vulnerable to `CVE-2022-48560`, which exists in versions `< 3.6.8-56.0.1.el8_9.2`.

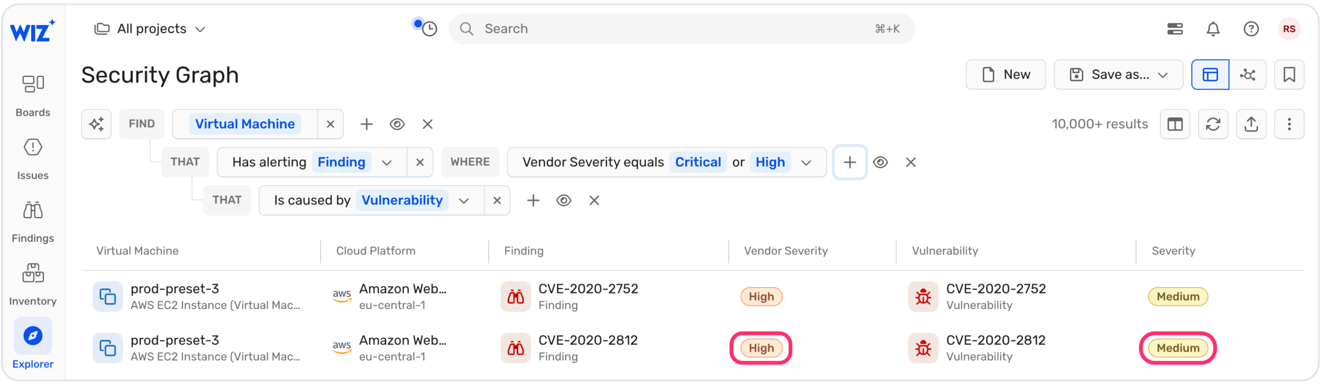
The vulnerability was found in the Official Oracle Security Advisories with vendor severity: `Medium` (NVD severity: `High`).

The vulnerability can be remediated by updating the package to version `3.6.8-56.0.1.el8_9.2` or higher, using `yum update python3-libs`.

Project: 6 Projects	Vendor Severity: Medium	Detailed name: python3-libs
Version: 3.6.8-47.0.1.el8_6	Fixed Version: 3.6.8-56.0.1.el8_9.2	Detection Method: Package
Data Source: linux.oracle.com		

Status: Unresolved
First seen: Jan 11, 2024, 4:16 PM
Last seen: Jan 11, 2024, 4:16 PM

Sometimes, the Finding and Vulnerability objects for the same CVE have different severities. In such cases, we generally prefer the Finding's severity because it has more context and better reflects the actual severity concerning the specific vulnerable product.



FIND Virtual Machine
THAT Has alerting Finding WHERE Vendor Severity equals Critical or High
THAT Is caused by Vulnerability

10,000+ results

Virtual Machine	Cloud Platform	Finding	Vendor Severity	Vulnerability	Severity
prod-preset-3 AWS EC2 Instance (Virtual Mac...	Amazon Web... eu-central-1	CVE-2020-2752 Finding	High	CVE-2020-2752 Vulnerability	Medium
prod-preset-3 AWS EC2 Instance (Virtual Mac...	Amazon Web... eu-central-1	CVE-2020-2812 Finding	High	CVE-2020-2812 Vulnerability	Medium

The severity score of each vulnerability is aligned with NIST's severity score, which is based on the [Common Vulnerability Scoring System (CVSS)](). In CVSS version 3.1, the severity score is composed of an exploitability score (which ranges between 0-3.9) and an impact score (which ranges between 0-6). The exploitability score depends on the attack vector parameter, which can be network (remote), adjacent (same physical/logical network), local, or physical (requires physical access to the computing infrastructure).

Defining Policies

- You can exclude Vulnerability Findings to reduce noise in your environment by creating Ignore Rules. You can also ignore specific Findings manually. Ignored Findings are removed from the Security Graph, reports, and the calculation of compliance scores. Learn about Ignore Rules.
- You can create CI/CD Policies and assign vulnerabilities a grace period (of hours or days) based on the fix date, to allow your development teams to address the vulnerability. During the grace period, Wiz ignores the vulnerability in CI/CD policies and does not fail the policy. Learn how to add a custom policy.

Prominent properties

Aside from basic information in the details drawer such as the Description section and the properties `Detailed Name`, `Detection Method`, and `Version`, Wiz enriches Vulnerability Findings and Vulnerabilities with several unique properties to assist with prioritization and remediation. Here are the most prominent ones:

- Vulnerability Findings:
  - First Seen and Last Seen
  - Vendor Severity
  - Fixed Version and Fix Date
  - (Container images) Layer Build Commands and Base Image Vulnerability
  - Validated in Runtime
- Vulnerabilities:
  - Published Date
  - CVSS properties
  - Exploit and CISA KEV Exploit
  - EPSS
  - Severity

First Seen and Last Seen

The `First Seen` property is the date the vulnerability was first detected on the resource. This date usually correlates with the first successful agentless workload scan for the resource.

Subsequently, the `Last Seen` property is when the vulnerability was last detected on the resource. This date usually correlates with the last successful agentless workload scan for the resource.

Fixed Version and Fix Date

If available, the `Fixed Version` property is provided. Wiz only publishes the relevant version of the fixed version that patches the vulnerability. You can view a list of all Findings with fixed versions using this query.

Wiz also enriches Vulnerability Findings with the vulnerability `Fix Date`. Wiz usually determines the fix date using various feeds since, many times, the vendors do not publish this information. You can review and filter for the fix date in the Security Graph, Vulnerability Findings page, and the Vulnerabilities Report.

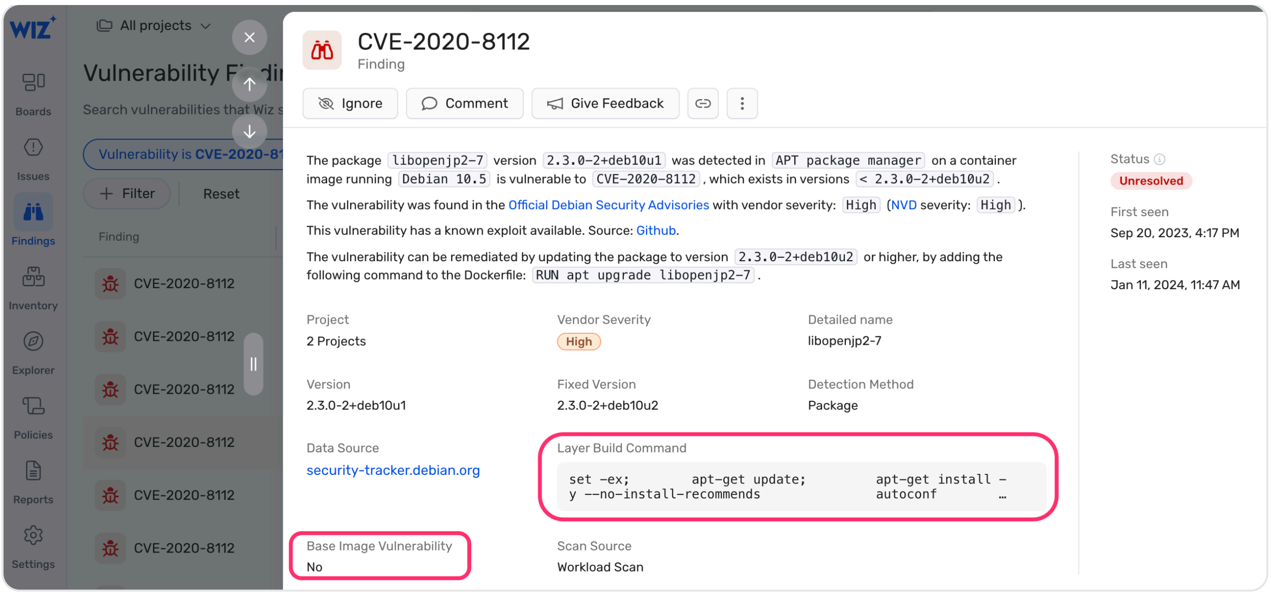
> When Wiz adds a new vulnerability source feed, we override the vendor fix date with a fix date of "Before MM/DD/YYYY", where MM/DD/YYYY is the date the source feed was added.

Layer Build Commands and Base Image Vulnerability

For container images only, Wiz performs per-layer analysis of vulnerabilities, allowing you to identify exactly which detected vulnerability was introduced with which specific image layer in the container image. The Finding displays the Docker file command that introduced the vulnerability and flags base-image vulnerabilities, both in the Vulnerability Findings page and the Security Graph.

Wiz also surfaces the boolean property `Base Image Vulnerability` for Vulnerability Findings. It should help facilitate easy remediation of vulnerabilities that are sourced from non-base images, as usually, base image vulnerabilities are outside of the ability of development teams to resolve.

If a vulnerability is introduced in one layer, and the vulnerable package is updated in a subsequent underlying layer, Wiz takes this into consideration and a Finding will be resolved for the original layer that introduced that vulnerability.



CVE-2020-8112
Finding

The package `libopenjp2-7` version `2.3.0-2+deb10u1` was detected in `APT package manager` on a container image running `Debian 10.5` is vulnerable to `CVE-2020-8112`, which exists in versions `< 2.3.0-2+deb10u2`.

The vulnerability was found in the Official Debian Security Advisories with vendor severity: `High` (NVD severity: `High`).

This vulnerability has a known exploit available. Source: Github.

The vulnerability can be remediated by updating the package to version `2.3.0-2+deb10u2` or higher, by adding the following command to the Dockerfile: `RUN apt upgrade libopenjp2-7`.

Project: 2 Projects	Vendor Severity: High	Detailed name: libopenjp2-7
Version: 2.3.0-2+deb10u1	Fixed Version: 2.3.0-2+deb10u2	Detection Method: Package
Data Source: security-tracker.debian.org	Layer Build Command: `set -ex; apt-get update; apt-get install -y --no-install-recommends autoconf …`	
Base Image Vulnerability: No	Scan Source: Workload Scan	

Status: Unresolved

First seen: Sep 20, 2023, 4:17 PM

Last seen: Jan 11, 2024, 11:47 AM

Additionally, vulnerabilities can be viewed grouped by layer on the container image object, which not only provides you with a more convenient way to view the information and identify the origin of vulnerabilities but also serves as a starting point for remediation. You can view this by clicking the Vulnerability tab in the details drawer of any container image.

Validated in Runtime

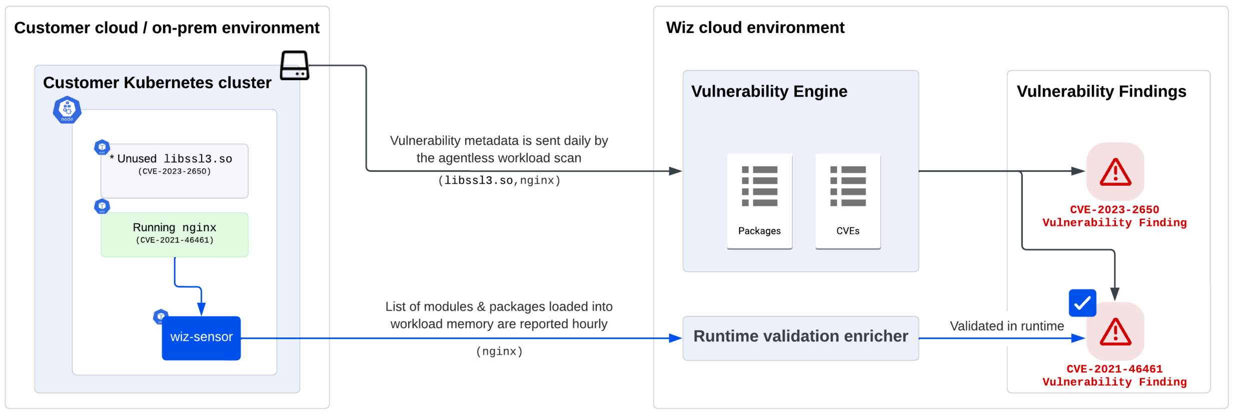
This capability requires a Wiz Runtime Sensor.

The Wiz Runtime Sensor adds runtime signals on top of the agentless vulnerability assessment. This allows Wiz to identify vulnerabilities executed in runtime so you can focus your remediation efforts on active vulnerable packages and their associated resources.

For each container where the Sensor is deployed, it generates a comprehensive list of all the modules which are active and loaded into memory and sends the list to the Wiz backend. When Wiz's Workload Scanner identifies a vulnerable package, we look for it on the list of modules. If the vulnerable package exists on the list, then Wiz generates a Vulnerability Finding for the resource and the vulnerable package, flags the Finding with the `Validated in Runtime` property and marks it with the ✓ icon. Learn about Security Graph icons.

The Sensor validates also ignored Vulnerability Findings. Learn about Ignore Rules.

The property is removed from the Vulnerability Finding if the Sensor does not detect the module loaded in memory within the last 48 hours.


Customer cloud / on-prem environment
Customer Kubernetes cluster
* Unused libssl3.so (CVE-2023-2650)
Running nginx (CVE-2021-46461)
wiz-sensor
Vulnerability metadata is sent daily by the agentless workload scan (libssl3.so, nginx)
List of modules & packages loaded into workload memory are reported hourly (nginx)
Wiz cloud environment
Vulnerability Engine
Packages
CVEs
Runtime validation enricher
Validated in runtime
Vulnerability Findings
CVE-2023-2650 Vulnerability Finding
CVE-2021-46461 Vulnerability Finding

*Unused `libssl3.co` indicates the binary file was not uploaded to memory

For example, let's look at a container with 120 packages, out of which 100 have vulnerabilities and 35 of the 100 vulnerable packages are active and loaded into memory. In this case, Wiz generates 100 Vulnerability Findings for the container, while the 35 Vulnerability Findings associated with the active packages are also flagged with the `Validated in Runtime` property. Use this query to see only Vulnerability Findings that were validated in runtime.

Monitoring and reporting mechanism

Packages

When an executable module (main binary or shared object) belonging to a vulnerable package is loaded into the memory of a process, this package is considered validated in runtime.

To generate the list of all the modules which are active and loaded into memory, the Sensor:

- Enumerates all modules loaded on Sensor startup (using `/proc/<pid>/maps`) to build a coherent map of the system state.
- Monitors all process-start (`exec`) system calls.
- Watches all Linux module load events in real-time (using `mmap`). This guarantees that both the modules loaded statically and the modules loaded dynamically (using `dlopen`) are monitored.

The Sensor monitoring is continuous and not based on polling. If a process or module is not seen for more than 48 hours, then it is considered stale (i.e., not validated in runtime) and therefore excluded from the hourly report sent to Wiz.

Libraries-.NET and Go

Wiz's agentless scanning detects vulnerable libraries used inside the main executable of a .NET or Go program. The Sensor detects when such a program is loaded into memory and marks the vulnerability as validated in runtime.

Libraries–Java

The Sensor monitors active `jar` and `war` files:

- Enumerates all such files loaded into a `java` process on Sensor startup (using the process `fd` map).
- Tracks additional files loaded into the `java` memory.

When a file path marked as active is determined to have a vulnerability by the Vulnerability Scanner, the vulnerability is marked as `Validated in runtime`.

Published Date

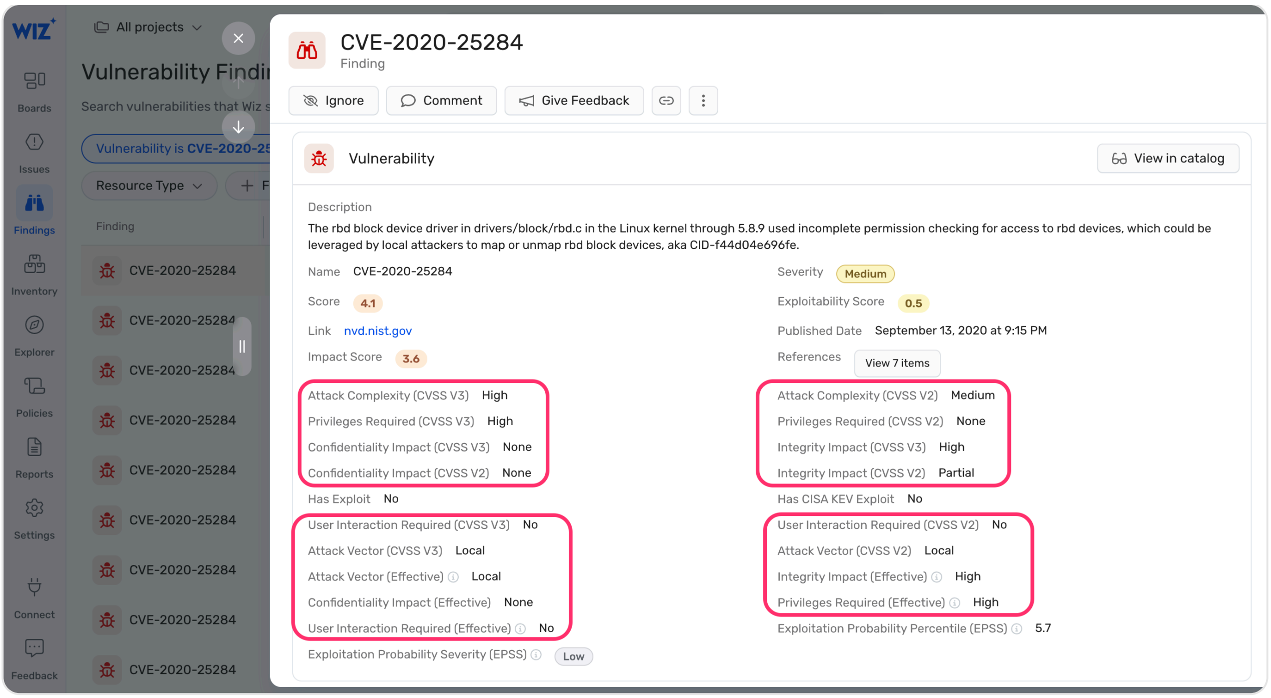
Wiz receives publish dates for vulnerabilities directly from vendor feeds. Like the CVSS properties, it is sourced directly from NVD or CNA. If there is no published CVE in the NVD database, Wiz will show the published date directly from the vendor, i.e., GitHub Advisory Database or PyUP.

CVSS properties

⚠ Due to the current substantial backlog of missing vulnerability metadata in NVD, Wiz uses an alternative method in cases of missing data. Learn more about bridging the NVD gap.

For all vulnerabilities, Wiz propagates CVSS V2 and/or CVSS V3.X metrics from NVD. This includes the following metrics from both CVSS V2 and CVSS V3:

- `Attack Complexity`
- `Attack Vector`
- `Confidentiality Impact`
- `Integrity Impact`
- `Privileges Required`
- `User Interaction Required`


All projects

Vulnerability Findi

Search vulnerabilities that Wiz s

Vulnerability is CVE-2020-25

Resource Type

Finding

CVE-2020-25284
CVE-2020-25284
CVE-2020-25284
CVE-2020-25284
CVE-2020-25284
CVE-2020-25284
CVE-2020-25284
CVE-2020-25284
CVE-2020-25284

CVE-2020-25284
Finding

Ignore | Comment | Give Feedback

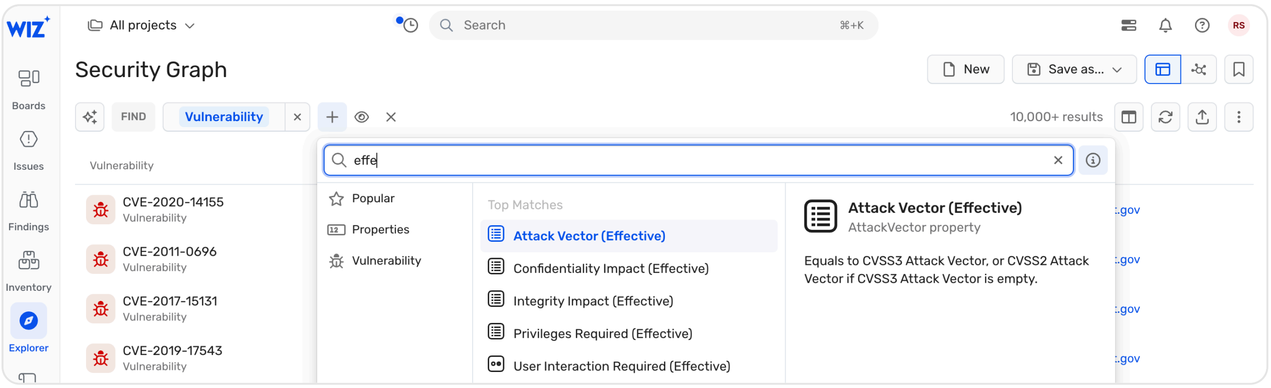
Vulnerability | View in catalog

Description

The rbd block device driver in drivers/block/rbd.c in the Linux kernel through 5.8.9 used incomplete permission checking for access to rbd devices, which could be leveraged by local attackers to map or unmap rbd block devices, aka CID-f44d04e696fe.

Name	CVE-2020-25284	Severity	Medium
Score	4.1	Exploitability Score	0.5
Link	nvd.nist.gov	Published Date	September 13, 2020 at 9:15 PM
Impact Score	3.6	References	View 7 items
Attack Complexity (CVSS V3)	High	Attack Complexity (CVSS V2)	Medium
Privileges Required (CVSS V3)	High	Privileges Required (CVSS V2)	None
Confidentiality Impact (CVSS V3)	None	Integrity Impact (CVSS V3)	High
Confidentiality Impact (CVSS V2)	None	Integrity Impact (CVSS V2)	Partial
Has Exploit	No	Has CISA KEV Exploit	No
User Interaction Required (CVSS V3)	No	User Interaction Required (CVSS V2)	No
Attack Vector (CVSS V3)	Local	Attack Vector (CVSS V2)	Local
Attack Vector (Effective)	Local	Integrity Impact (Effective)	High
Confidentiality Impact (Effective)	None	Privileges Required (Effective)	High
User Interaction Required (Effective)	No	Exploitation Probability Percentile (EPSS)	5.7
Exploitation Probability Severity (EPSS)	Low		

Not all vulnerabilities have both CVSS V2 and CVSS V3 metrics. In such cases, CVSS metrics will not be shown for the vulnerability. To avoid excluding vulnerabilities from Security Graph queries if querying by one of these CVSS versions, Wiz surfaces an `Effective` property for each of the CVSS metrics. This Effective metric is always equal to the CVSS V3 metric value, or the CVSS V2 metric value if the CVSS V3 metric value is unavailable.


All projects

Search

Security Graph

New | Save as...

FIND | Vulnerability

10,000+ results

Vulnerability

CVE-2020-14155
Vulnerability

CVE-2011-0696
Vulnerability

CVE-2017-15131
Vulnerability

CVE-2019-17543
Vulnerability

effe

Popular
Properties
Vulnerability

Top Matches

Attack Vector (Effective)
Confidentiality Impact (Effective)
Integrity Impact (Effective)
Privileges Required (Effective)
User Interaction Required (Effective)

Attack Vector (Effective)
AttackVector property

Equals to CVSS3 Attack Vector, or CVSS2 Attack Vector if CVSS3 Attack Vector is empty.

Exploit and CISA KEV Exploit

Wiz uses Exploit Database, GitHub, and Packetstorm to identify which CVEs have been exploited in the wild (most CVEs are never actually "weaponized" by malicious actors). The exploitability score is taken from the CVSS score and consists of "Impact" and "Exploitability".

The `CISA KEV Exploit` property is sourced from the CISA Known Exploited Vulnerabilities (KEV) catalog and can help you prioritize high-impact vulnerabilities that have active exploits in the wild.

EPSS

The Exploit Probability Scoring System (EPSS) is a scoring system designed to predict the likelihood that a given vulnerability will be exploited in the wild. The EPSS score is calculated using hundreds of factors, such as the technical characteristics of the vulnerability, the reputation of software vendors, and the availability of exploit code in public repositories.

> While CVSS tells you how impactful a vulnerability could be, EPSS predicts how likely it is to be attacked.

Wiz ingests EPSS data for every vulnerability in the Wiz vulnerability catalog. This allows you to supplement EPSS context to your vulnerability queries and remediation processes to better prioritize Issues in your environment by focusing on the vulnerabilities with a higher likelihood of being exploited.

The following EPSS properties are available:

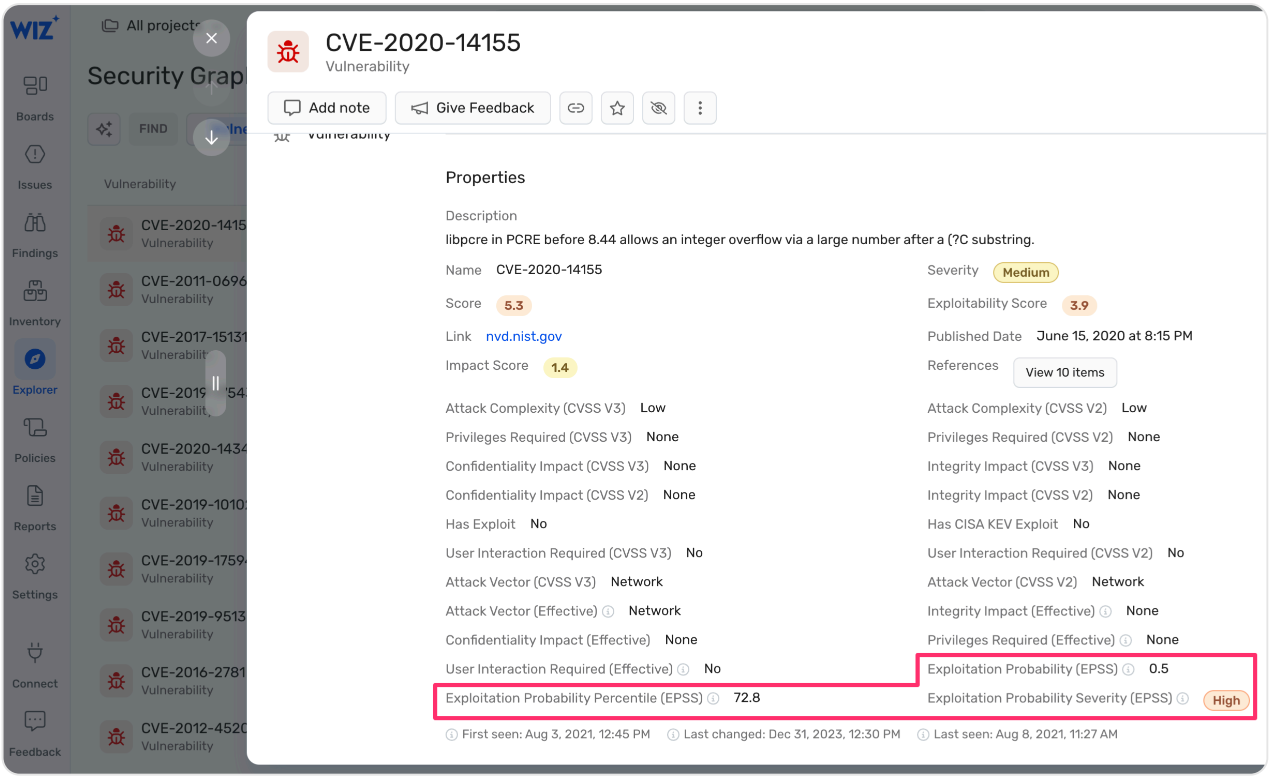
- `Exploitation Probability` –A direct measure of the likelihood of an attacker exploiting this particular vulnerability in the next 30 days, ignoring environmental context. The vast majority of CVEs have a very low exploitation probability due to the difficulty of exploitation or lack of exploit code in the wild, among other factors. This reflects the reality that most vulnerabilities are never exploited in the wild.

  > Wiz rounds this score to 1 decimal place for efficiency reasons, to avoid updating each entry in our database too often (unlike CVSS scores, EPSS scores fluctuate as time goes by).

- `Exploitation Probability Percentile` –The percentage of vulnerabilities less likely to be exploited than this vulnerability. EPSS data provides this as a number between 0-1, which is multiplied by Wiz by a factor of 100 and displayed as a percentage between 0-100 in the Wiz portal. For example, a 90% percentile indicates that 90% of all other vulnerabilities are less likely to be exploited than this one. The higher the percentile is, the higher the relative probability of this vulnerability being exploited, and the higher the relative importance of this vulnerability to be actioned upon if found in your environment.
- `Exploitation Probability Severity` –Severity mapping, calculated as follows based on the Exploitation Probability Percentile:
  - Low: 0-49.9%
  - Medium: 50-64.9%

- High: 65-84.9%
- Critical: 85-100%

  These thresholds were chosen by Wiz based on existing EPSS percentile data to ensure even distribution between critical, high, and medium severities.



CVE-2020-14155
Vulnerability

Properties

Description
libpcre in PCRE before 8.44 allows an integer overflow via a large number after a (?C substring.

Name	CVE-2020-14155	Severity	Medium
Score	5.3	Exploitability Score	3.9
Link	nvd.nist.gov	Published Date	June 15, 2020 at 8:15 PM
Impact Score	1.4	References	View 10 items
Attack Complexity (CVSS V3)	Low	Attack Complexity (CVSS V2)	Low
Privileges Required (CVSS V3)	None	Privileges Required (CVSS V2)	None
Confidentiality Impact (CVSS V3)	None	Integrity Impact (CVSS V3)	None
Confidentiality Impact (CVSS V2)	None	Integrity Impact (CVSS V2)	None
Has Exploit	No	Has CISA KEV Exploit	No
User Interaction Required (CVSS V3)	No	User Interaction Required (CVSS V2)	No
Attack Vector (CVSS V3)	Network	Attack Vector (CVSS V2)	Network
Attack Vector (Effective)	Network	Integrity Impact (Effective)	None
Confidentiality Impact (Effective)	None	Privileges Required (Effective)	None
User Interaction Required (Effective)	No	Exploitation Probability (EPSS)	0.5
Exploitation Probability Percentile (EPSS)	72.8	Exploitation Probability Severity (EPSS)	High

First seen: Aug 3, 2021, 12:45 PM Last changed: Dec 31, 2023, 12:30 PM Last seen: Aug 8, 2021, 11:27 AM

Patch management enrichment properties

After technologies are detected using agentless scanning, some technologies are further assessed to enable patch management. This includes enriching them with the following properties:

- Is Version End of Life
- Is Latest Version
- Latest Version
- Extended Support
- Version End of Life Date
- Latest Version Release Date
- Version Release Date
- Requires restart

Is Version End of Life

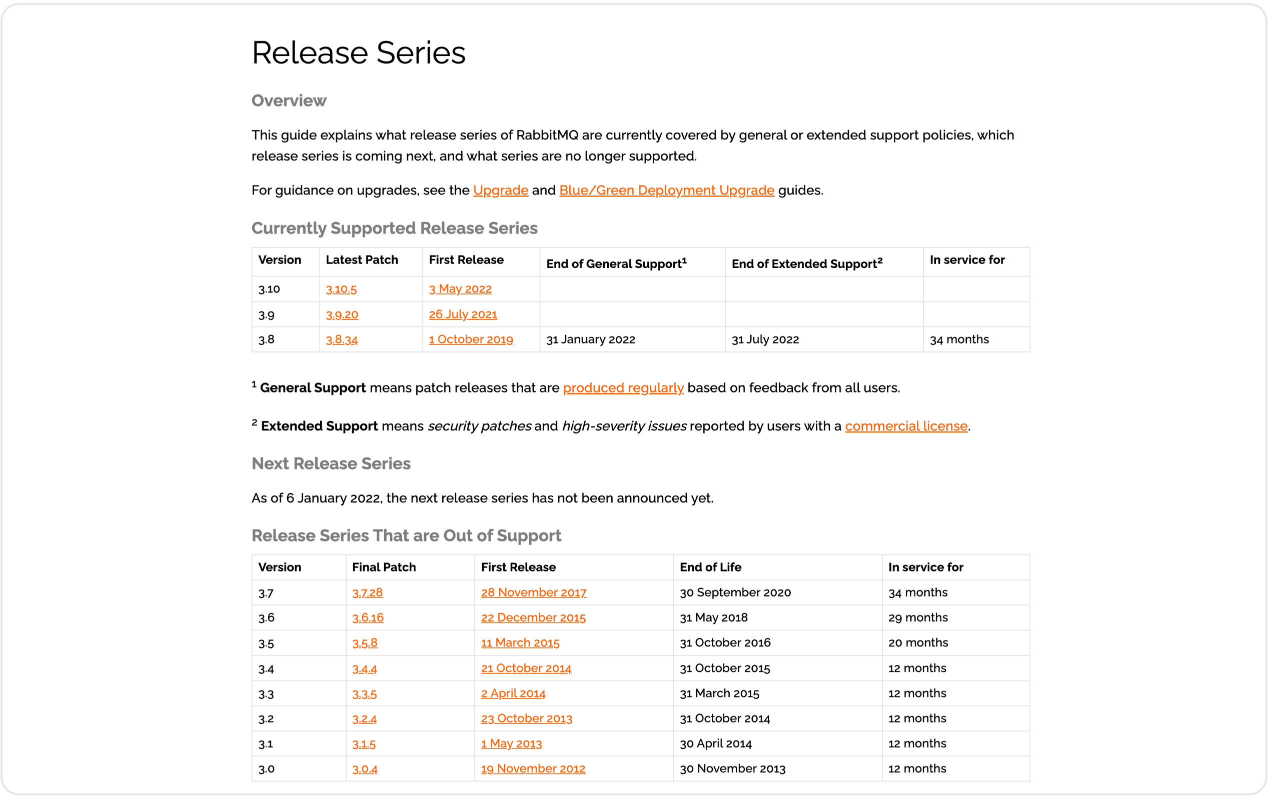
Wiz can identify the technologies listed below as End-of-Life (EOL). This is determined by the Vendor's official data or other publicly available sources, which

Wiz analyzes and feeds into its knowledge base. For example, in RabbitMQ release data, all versions below 3.8 are EOL (as of June 2022).

Is Latest Version and Latest Version

Latest versions are determined using the Wiz knowledge base, which is fed by all versions that have been retrieved across all customers. This means that once Wiz sees a new version multiple times across customer environments, a validation process is triggered that results in the knowledge base being updated with a new latest version.

For technologies where EOL enrichment is supported, Wiz analyzes the different "streams" based on the vendor's data. A "stream" is a specific version range, usually major or minor versions. Once "streams" are analyzed, Wiz suggests the latest version only in the current version stream, unless the entire stream is EOL. In that case (when the current version stream is EOL), the latest version of the next non-EOL stream will be suggested. For example, let's look at RabbitMQ release data:



Release Series

Overview

This guide explains what release series of RabbitMQ are currently covered by general or extended support policies, which release series is coming next, and what series are no longer supported.

For guidance on upgrades, see the Upgrade and Blue/Green Deployment Upgrade guides.

Currently Supported Release Series

Version	Latest Patch	First Release	End of General Support[1]	End of Extended Support[2]	In service for
3.10	3.10.5	3 May 2022			
3.9	3.9.20	26 July 2021			
3.8	3.8.34	1 October 2019	31 January 2022	31 July 2022	34 months

[1] **General Support** means patch releases that are produced regularly based on feedback from all users.

[2] **Extended Support** means *security patches* and *high-severity issues* reported by users with a commercial license.

Next Release Series

As of 6 January 2022, the next release series has not been announced yet.

Release Series That are Out of Support

Version	Final Patch	First Release	End of Life	In service for
3.7	3.7.28	28 November 2017	30 September 2020	34 months
3.6	3.6.16	22 December 2015	31 May 2018	29 months
3.5	3.5.8	11 March 2015	31 October 2016	20 months
3.4	3.4.4	21 October 2014	31 October 2015	12 months
3.3	3.3.5	2 April 2014	31 March 2015	12 months
3.2	3.2.4	23 October 2013	31 October 2014	12 months
3.1	3.1.5	1 May 2013	30 April 2014	12 months
3.0	3.0.4	19 November 2012	30 November 2013	12 months

The streams in this case are according to the minor version: 3.0.x, 3.1.x,...,3.10.x.

Let's say that Wiz detects version `3.2`. In that case:

- `Is Latest Version` = false
- `Latest Version` = 3.8.34 (the latest version of the next non-EOL stream available)
- `Is Version End of Life` = true

Now let's say that Wiz detects version `3.8`. In that case:

- `Is Latest Version` = false (there is a latest patch available)
- `Latest Version` = 3.8.34
- `Is Version End of Life` = false

Extended Support

For the technologies listed below, Wiz dynamically analyses whether the currently installed technology is with extended support (e.g. ESM in Ubuntu, EUS/ELS in Redhat). When a technology supports `Extended Support` analysis, it affects the following areas:

1. The property `Extended Support` is either set with `true` or `false`, based on the detection.
2. The `Is Version End of Life` is dynamically populated according to whether `Extended Support` is enabled. Extended support, by its nature, extends the date of security updates. Wiz reflects this.
3. In terms of vulnerability detection, Extended Support versions are being supported and assessed. For example, if an Extended Support OS is vulnerable to a CVE that is fixed in later versions of the Extended Support OS, the vulnerability will be reported until the version is upgraded.
4. If a vulnerability is fixed only in the Extended Support version, it will be reported on `Extended Support` = false versions. This is to ensure you are aware of fixes that are available.

Version End of Life Date, Latest Version Release Date, and Version Release Date

Version release dates are assessed automatically using external sources and Wiz's own knowledge base. For example, for MongoDB, the assessment automatically determines the EOL date based on MongoDB's Release Notes and Support Policy.

> These assessments are applied only for the technologies listed below.

Requires restart

Wiz can detect whether supported operating systems require a restart in order to fully deploy any patches.

The following operating systems support the `Requires Restart` property:

- AlmaLinux
- Amazon Linux
- CentOS
- Debian
- Oracle Linux
- RedHat
- Rocky Linux
- SUSE
- Ubuntu
- Windows

This property is exposed on the "Linux Kernel" and "Windows Cumulative Update" hosted technologies. To see systems that require a restart, use this query.

Hosted technologies enriched with patch management

The following hosted technologies are enriched with release and EOL dates based on the listed sources:

Hosted technologies

Technology	Release date	EOL	Extended support	Source(s)	Notes
Alpine	▯	▯	—	Alpine Linux	
Amazon 1	▯	▯	—	Amazon Linux AMI Releases and Amazon Linux EOL Update	EOL is `Mainte` `Suppor`
Amazon 2	▯	▯	—	Amazon Linux 2 FAQ	
Amazon Linux 2022	▯	—	—	Amazon Linux 2022 FAQ	
AlmaLinux	▯	▯	—	Wikipedia	
Apache HTTP Server	▯	—	—	Apache HTTP Server Source Code Distributions	

Technology	Release date	EOL	Extended support	Source(s)	Notes
Azure Kubernetes Service	□	—	—	Microsoft	
Bottlerocket	□	□	—	Official Bottlerocket Github and Kubernetes EOL	
CBL-Mariner	□	□	—	CBL-Mariner Documentation	EOL da determ versior + 18 n accord docum `releas availa and ea suppor months`
Celery	□	—	—	PyPI	
CentOS	□	□	—	Wikipedia	
Chef Client	□	—	—	Chef and Chef Client Changelog	
Chef Server	□	—	—	Chef and Chef Server Changelog	
ChefDK	□	—	—	ChefDK Changelog	
Container Optimized OS	□	—	—	Google Cloud	
Couchbase	□	□	—	Couchbase Server Release Notes Couchbase Enterprise Software Support Policy	

Technology	Release date	EOL	Extended support	Source(s)	Notes
Datastax Enterprise	▯	—	—	GitHub	
Debian	▯	▯	—	Debian Release and LTS	
Django	▯	▯	—	Django releases and downloads	EOL is `of ext`
Kubernetes Service (EKS)	▯	—	—	GitHub and Amazon	
ElasticSearch	▯	▯	—	GitHub and Elastic	
ESXi Hypervisor	▯	▯	—	VMware Lifecyle Matrix and VMware vSphere documentation	
Firefox	▯	—	—	Mozilla	
Flask	▯	—	—	PyPI	
Google Chrome	▯	—	—	Google	
Google Kubernetes Engine (GKE)	▯	—	—	Google Cloud release notes, both regular and stable	
Gunicorn	▯	—	—	PyPI	
Haproxy	▯	▯	—	Haproxy	
InfluxDB	▯	—	—	v1 changelog and v2 release notes	
Jenkins	▯	—	—	Jenkins	
Kubernetes	▯	▯	—	GitHub and Wikipedia	
Log4j	▯	—	—	Apache	
macOS	▯	▯		macOS Release Notes	
MariaDB	▯	▯	—	MariaDB	

Technology	Release date	EOL	Extended support	Source(s)	Notes
MongoDB	▯	▯	—	MongoDB [release notes](#) and [support policy](#)	
MySQL	▯	▯	—	[MySQL](#) and [Wikipedia](#)	
.NET	▯	▯	—	Wiz knowledge base	
NGINX	▯	▯	—	[NGINX](#)	Only th branch mainta legacy EOL. E to the release
NodeJS	▯	▯	—	[NodeJS](#) and [Wikipedia](#)	EOL is `Mainte`
PHP[1]	—	▯	—	[PHP EOL page](#) and [PHP changelog page](#)	
PostgreSQL	▯	▯	—	PostgreSQL [releases](#) and [versioning](#)	EOL is `Final`
Python Interpreter[1]	—	▯	—	[Python website](#) and [Wikipedia](#)	
RabbitMQ	▯	▯	—	RabbitMQ [changelog](#) and [versions](#)	
RedHat	▯	▯	▯ (EUS/ELS)	RedHat [articles](#) and [product life cycles](#)	EOL is `Mainte` `Suppor` `Suppor` availab `Extend` `suppor` otherw

Technology	Release date	EOL	Extended support	Source(s)	Notes
Redis	▯	▯	—	GitHub and Redis	
Redis on Windows	▯	—	—	Redis	
Rocky Linux	▯	▯	—	Rocky Linux Official Forum	Non-la versior consid of life, the off Linux F
Ruby[1]	—	▯	—	Ruby website and Wikipedia	
Splunk	▯	▯	—	Splunk release notes and support policy	EOL is `of P3`
SQL Server	▯	—	—	Wiz Knowledge Base	
SQL Server Studio	▯	—	—	Wiz Knowledge Base	
Subversion	▯	—	—	Apache	
SUSE	▯	▯	—	SUSE	EOL is `Genera`
TimescaleDB	▯	—	—	TimescaleDB Changelog	
Tomcat	▯	▯	—	Apache and Wikipedia	
Ubuntu	▯	▯	▯ (ESM)	Ubuntu	EOL is `of Sta` if avail `of Lif`
Wolfi/Chainguard	—	—	—	Wolfi/Chainguard does not have EOL	

> [1] EOL detection is supported for this technology. Being very common, it is not included in the Inventory to reduce noise.

FAQ

Questions? Take a look at the FAQ.

Updated 7 days ago

← Export Vulnerability Findings

Vulnerability Management Tutorials →