



Community-Powered Vulnerability Management

Team Members:

Marlon Iglesias Diaz, Theodore Atis, Daniel Granados

Product Owner:

Masoud Sadjadi

Purpose:



COLLECTIVE EXPERTISE:
HARNESSING THE KNOWLEDGE
OF A DIVERSE COMMUNITY TO
IDENTIFY VULNERABILITIES.



RAPID DETECTION:
ACCELERATING THE
DISCOVERY OF SECURITY
THREATS THROUGH
COLLABORATIVE EFFORTS.



EFFECTIVE MITIGATION:
ENHANCING CYBERSECURITY
RESILIENCE BY LEVERAGING
SHARED INSIGHTS FOR
PROMPT ACTION.

Introduction to Vulnerability Management

- Vulnerability management is the systematic process of discovering, assessing, and mitigating security weaknesses in digital systems

Here are some of the software we used for vulnerability testing

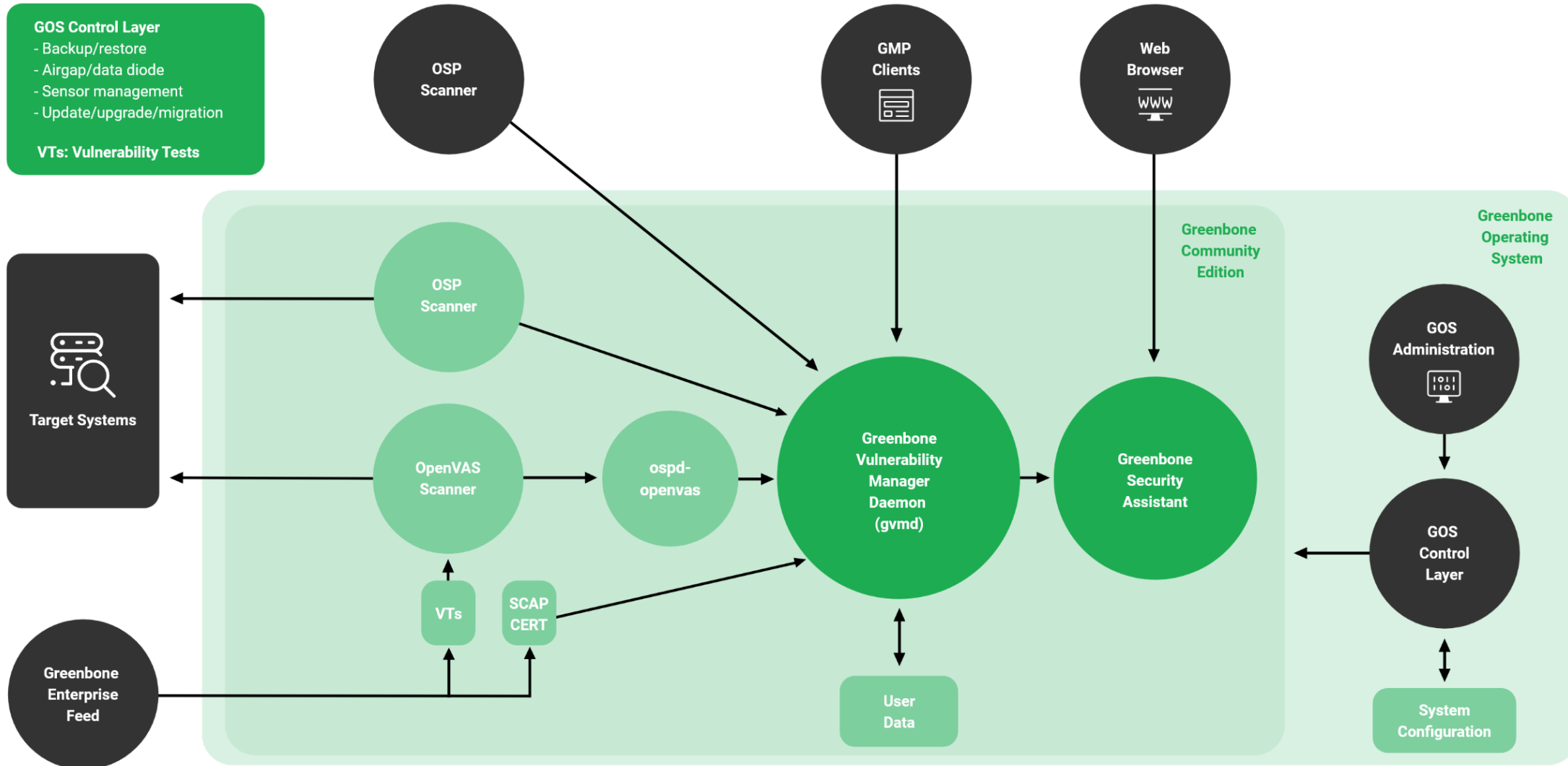


Greenbone Community Edition

- The Greenbone Community Edition, initially named "OpenVAS," is developed by Greenbone for its Enterprise products. This open-source edition enables Linux distributions to package and distribute its software components.



Greenbone





Greenbone Vulnerability Manager (gvmd)

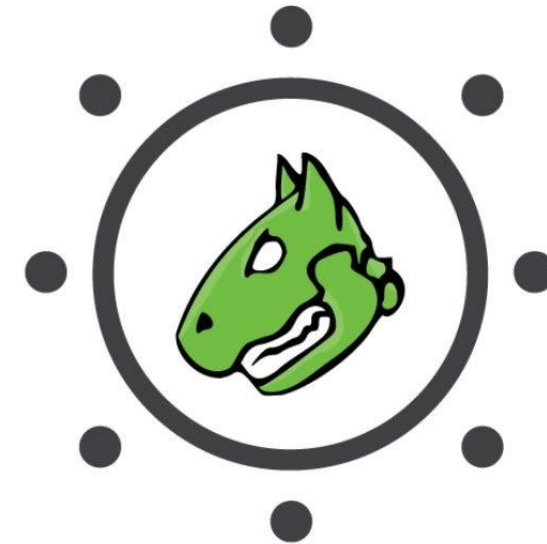
- gvmd serves as the central component that transforms basic vulnerability scanning into a comprehensive vulnerability management solution. It oversees and coordinates the activities of the OpenVAS Scanner through the use of the Open Scanner Protocol (OSP).



Greenbone
Vulnerability
Manager
Daemon
(gvmd)

OpenVAS Scanner

- OpenVAS Scanner is a robust scanning engine designed to perform vulnerability tests (VTs) on target systems. To accomplish this, it utilizes regularly updated and comprehensive feeds.



OpenVAS

Open Vulnerability Assessment Scanner

FIU

Engineering
& Computing

FLORIDA INTERNATIONAL UNIVERSITY



OpenVAS Explained

- **OpenVAS** categorizes vulnerabilities using **CVEs** and **CPEs** as a part of the **NVD** to create a **CVSS** all of which is managed by **NIST**. It finds those vulnerabilities through **NVTs / VTs**.
- CVE - Common Vulnerabilities and Exposure
- CPE - Common Platform Enumeration
- NVD - National Vulnerability Database
- CVSS - Common Vulnerability Scoring System
- NIST - National Institute of Standards and Technology
- NVT / VT – Network Vulnerability Tests / Vulnerability Tests



CVE - a standardized identification system for uniquely naming and tracking security vulnerabilities.

CPE - standardized method for naming and describing software and hardware platforms.

NVD - a detailed repository that provides information on vulnerabilities, offering details on their elimination, severity, potential impact, and affected products.

CVSS - a framework for assessing and communicating the severity of vulnerabilities through a numerical score based on factors such as impact, exploitability, and scope.

NIST - a U.S. federal agency that develops and promotes measurement standards

NVT / VT – scripts or plugins used in vulnerability scanning tools to automate the detection of security weaknesses in networks and systems.

OpenVas Explained

DFN-CERT Advisory - German:
*Deutsches Forschungsnetz, abbreviated
as DFN*

*services include categorizing,
distributing, and rating advisories from
various software vendors and
distributors.*

CERT-Bund Advisory - *The Computer
Emergency Response Team of the
German Federal Office for Information
Security (BSI)*

*issues preventive recommendations,
identifying vulnerabilities in hardware
and software, proposing solutions for
known vulnerabilities, aiding public
agencies in responding to IT security
incidents, and suggesting diverse
mitigation measures.*

OpenVAS is based in Osnabrück, Germany





Greenbone Setup

Source

- Through your Linux machine or virtual machine, clone the source code repository using Git, install the necessary dependencies, configure the build, compile the source code, and then install the Greenbone components.

Docker

- A Docker container is a compact, self-sufficient software package containing all essential components for running an applications. This enables the deployment of the containerized application across various environments with ease.



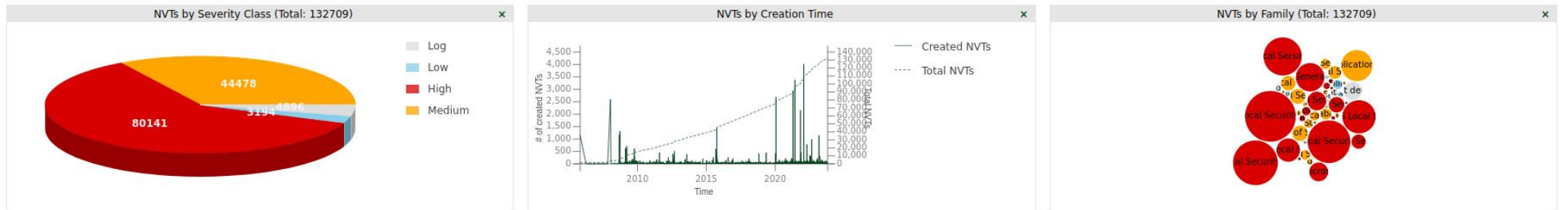
Scan Results

Information	Results (241 of 381)	Hosts (11 of 11)	Ports (20 of 20)	Applications (2 of 2)	Operating Systems (1 of 1)	CVEs (1 of 1)	Closed CVEs (0 of 0)	TLS Certificates (1 of 1)	Error Messages (0 of 0)	User Tags (0)
1 - 100 of 241										
Vulnerability		Severity ▼	QoD	Host		Location		Created		
				IP	Name					
OpenVAS Framework Components End Of Life Detection		10.0 (High)	80 %	192.168.117.12	scan-target.greenbone.net	general/tcp		Thu, Oct 18, 2018 2:09 PM UTC		
OS End Of Life Detection		10.0 (High)	80 %	192.168.126.4	scan-target-3.greenbone.net	general/tcp		Thu, Oct 18, 2018 2:07 PM UTC		
OS End Of Life Detection		10.0 (High)	80 %	192.168.117.12	scan-target.greenbone.net	general/tcp		Thu, Oct 18, 2018 2:08 PM UTC		
Anonymous FTP Login Reporting		6.4 (Medium)	80 %	192.168.126.52		21/tcp (IANA: ftp)		Thu, Oct 18, 2018 2:12 PM UTC		
Cleartext Transmission of Sensitive Information via HTTP		4.8 (Medium)	80 %	192.168.0.127	scan-target-4.greenbone.net	80/tcp (IANA: www-http)		Thu, Oct 18, 2018 2:09 PM UTC		
SSH Weak Encryption Algorithms Supported		4.3 (Medium)	95 %	192.168.116.4		22/tcp (IANA: ssh)		Thu, Oct 18, 2018 2:07 PM UTC		
SSH Weak Encryption Algorithms Supported		4.3 (Medium)	95 %	192.168.0.12	scan-target-2.greenbone.net	22/tcp (IANA: ssh)		Thu, Oct 18, 2018 2:11 PM UTC		
SSH Weak MAC Algorithms Supported		2.6 (Low)	95 %	192.168.116.9		22/tcp (IANA: ssh)		Thu, Oct 18, 2018 2:07 PM UTC		

NVTs



NVTs 132709 of 132709



1 - 10 of 132709

Name	Family	Created ▼	Modified	CVE	Severity	QoD
Debian: Security Advisory (DLA-3626)	Debian Local Security Checks	Mon, Oct 23, 2023 4:24 AM UTC	Mon, Oct 23, 2023 4:24 AM UTC	CVE-2023-36054	6.5 (Medium)	97 %
Debian: Security Advisory (DLA-3625)	Debian Local Security Checks	Mon, Oct 23, 2023 4:24 AM UTC	Mon, Oct 23, 2023 4:24 AM UTC	CVE-2023-5349	5.0 (Medium)	97 %
Debian: Security Advisory (DLA-3624)	Debian Local Security Checks	Mon, Oct 23, 2023 4:24 AM UTC	Mon, Oct 23, 2023 4:24 AM UTC	CVE-2023-44981	9.1 (High)	97 %
Debian: Security Advisory (DLA-3622)	Debian Local Security Checks	Mon, Oct 23, 2023 4:24 AM UTC	Mon, Oct 23, 2023 4:24 AM UTC	CVE-2023-40743	9.8 (High)	97 %
Debian: Security Advisory (DSA-5530)	Debian Local Security Checks	Mon, Oct 23, 2023 4:24 AM UTC	Mon, Oct 23, 2023 4:24 AM UTC	CVE-2022-30122 CVE-2022-30123 CVE-2022-44570 CVE-2022-44571 CVE-2022-44572 CVE-2023-27530 CVE-2023-27539	10.0 (High)	97 %
SUSE: Security Advisory (SUSE-SU-2023:4152-1)	SuSE Local Security Checks	Mon, Oct 23, 2023 4:21 AM UTC	Mon, Oct 23, 2023 4:21 AM UTC	CVE-2023-22081	5.3 (Medium)	97 %
SUSE: Security Advisory (SUSE-SU-2023:4142-1)	SuSE Local Security Checks	Mon, Oct 23, 2023 4:21 AM UTC	Mon, Oct 23, 2023 4:21 AM UTC	CVE-2020-36766 CVE-2023-1192 CVE-2023-1206 CVE-2023-1859 CVE-2023-2177 CVE-2023-4004 CVE-2023-40283 CVE-2023-42753 CVE-2023-4389 CVE-2023-4622 CVE-2023-4623 CVE-2023-4881 CVE-2023-4921	7.8 (High)	97 %
SUSE: Security Advisory (SUSE-SU-2023:4141-1)	SuSE Local Security Checks	Mon, Oct 23, 2023 4:21 AM UTC	Mon, Oct 23, 2023 4:21 AM UTC	CVE-2023-4692 CVE-2023-4693	5.0 (Medium)	97 %
SUSE: Security Advisory (SUSE-SU-2023:4140-1)	SuSE Local Security Checks	Mon, Oct 23, 2023 4:21 AM UTC	Mon, Oct 23, 2023 4:21 AM UTC	CVE-2023-4692 CVE-2023-4693	5.0 (Medium)	97 %
Slackware: Security Advisory (SSA:2023-295-01)	Slackware Local Security Checks	Mon, Oct 23, 2023 4:17 AM UTC	Mon, Oct 23, 2023 4:17 AM UTC	CVE-2021-32142 CVE-2023-1729	7.8 (High)	97 %

Apply to page contents

1 - 10 of 132709

(Applied filter: sort-reverse=created rows=10 first=1)

FIU

Engineering
& Computing

FLORIDA INTERNATIONAL UNIVERSITY



Making NVTs

1

NVTs are created using NASL scripts, which tests for vulnerabilities by checking for known security issues in target systems.

2

NASL, commonly known as the Nessus Attack Scripting Language, is a script language utilized by vulnerability scanners such as Nessus and OpenVAS.



Custom NVTs

- New NVTs

To create a custom NVT library you can start by creating simple device specific NASL script to serve as a foundation and reference for your complex NVTs later on.

- Adding to NVTs

Another approach to niche vulnerability testing is to modify a current NASL script to fit the needs of the device, network, or system.





Simple NVTs

Building Blocks:
cover fundamental
security checks that
are widely applicable

Progressive Learning:
beginners and new
members of the
community can
contribute

Adaptability:
can be adapted and
modified to suit
specific
circumstances



Adding to NVTs



- Currently there is a Privilege Escalation Vulnerability (CVE-2023-20198)
- To quickly have an NVT able to check for the vulnerability you can modify or reference similar NVT before official NVT are added to the library



Our NVTs

- Simple NVT to check for ipv6 connections

In this case, there were no NVTs dedicated for the detection of ipv6 which can be then reference by future more complex NVTs that check for vulnerabilities with ipv6.

- Modifying current NVT to check for CVE-2023-20198

Modifying a current NVT with NASL script to check for the CVE-2023-20198 vulnerability.



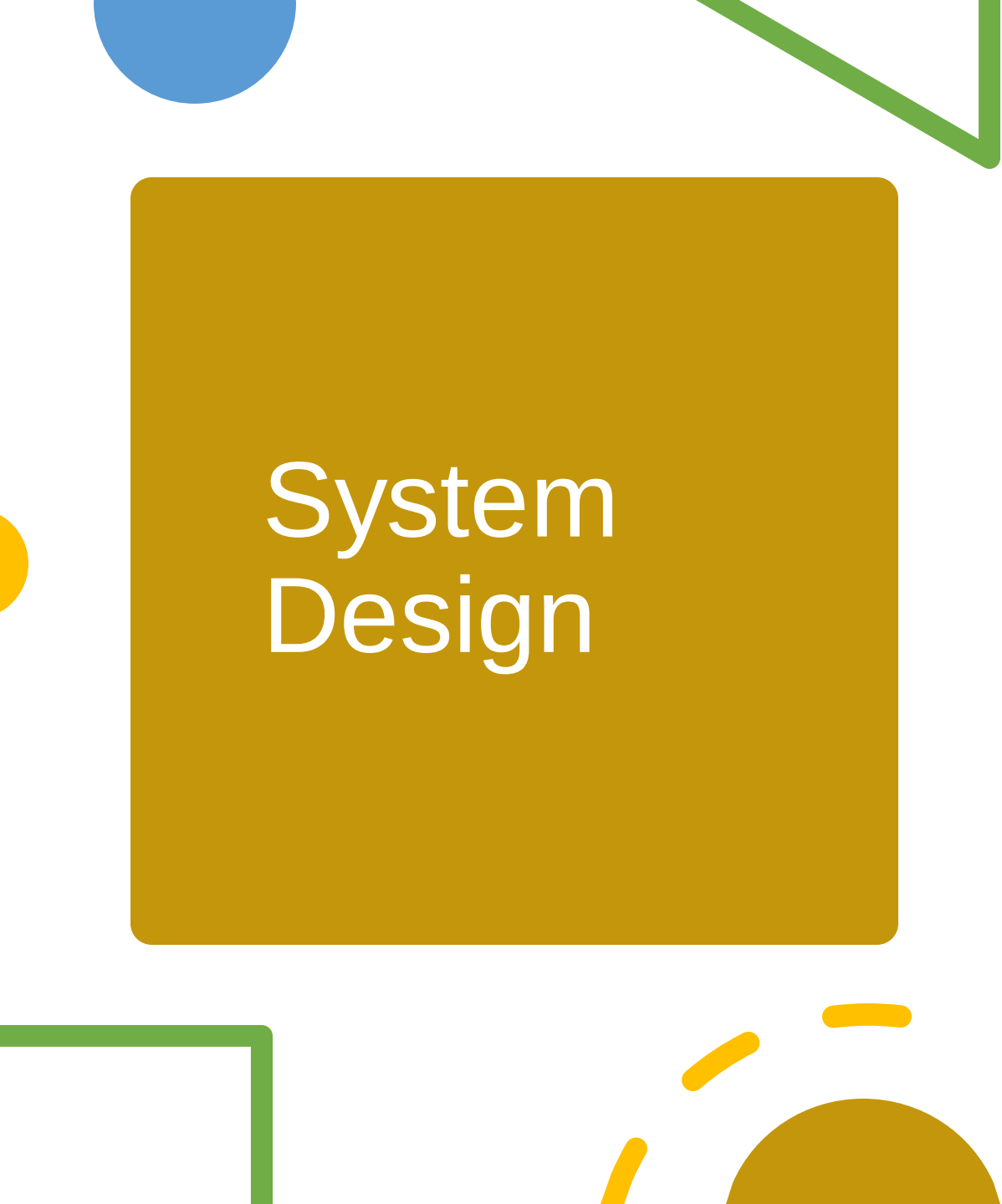
How?

All of this is only possible
because it is open-source and
their communities for assistance.



GitLab DevSecOps CI/CD

- DevSecOps is a combination of development, security, and operations. It is an approach to software development that integrates security throughout the development lifecycle.
- CI/CD is a continuous method of software development, where you continuously build, test, deploy, and monitor iterative code changes.



System Design

- The code for their software project and store it in a central place called a Git repository on GitLab.
- A simple file is created called `.gitlab-ci.yml` where they define how they want their code to be tested and deployed.
- Whenever changes to the code are made, an automatic process kicks in (CI). This process checks if the code still works and didn't break anything by running automated tests.
- This process checks if the code still works and didn't break anything by running automated tests.
- Once the code is tested and packaged, there's another automatic process (CD) that can deploy the software to different places.
- Throughout the process, GitLab keeps track of what's happening, providing logs and reports so developers can see if everything is going smoothly.
- By automating these processes and integrating them into a single platform, GitLab's DevSecOps CI/CD streamlines development workflows, enhances collaboration, and improves the overall efficiency of software delivery while incorporating security measures at every step.



GitLab DevSecOps Explained

- Developers start by pushing their code to a Git repository hosted on GitLab.
- Alongside the code, CI/CD configurations are defined using a file (commonly `.gitlab-ci.yml`). This file outlines the steps and processes to be executed during the CI/CD pipeline.
 - Whenever changes to the code are made, an automatic process kicks in (CI). This process checks if the code still works and didn't break anything by running automated tests.
 - Once the code is tested and packaged, there's another automatic process (CD) that can deploy the software to different places.





GitLab Common Terms

- **The .gitlab-ci.yml file:** To use GitLab CI/CD, you start with a .gitlab-ci.yml file at the root of your project which contains the configuration for your CI/CD pipeline.
- **Runners:** The agents that run your jobs, they can run on physical machines or virtual instances. The runner loads the image, clones your project and runs the job either locally or in the container.
- **Pipelines:** Pipelines are made up of jobs and stages. Jobs define what you want to do and are grouped into stages. Each stage contains at least one job, and typical stages might be build, test, and deploy.
- **CI/CD Variables:** Helps customize jobs by making values defined elsewhere accessible to jobs.



Testing

- Pictured here is some of the YAML code that went into setting up the GitLab security tests.
- We're running container scans, dynamic and static application security tests as well as dependency scanning.

```
ci > security_scans.gitlab-ci.yml
1  ---
2  # All GitLab Security Scans and Reports with downloadable artifacts that expire in 16 weeks.
3
4  gemnasium-dependency_scanning:
5    stage: test
6    needs: [build-job]
7    allow_failure: true
8    interruptible: true
9    script:
10     - echo NOOP
11  artifacts:
12    reports:
13     dependency_scanning: samples/dependency-scanning.json
14    expire_in: 16 weeks
15
16  container-scanning:
17    stage: test
18    needs: [build-job]
19    allow_failure: true
20    interruptible: true
21    script:
22     - echo NOOP
23  artifacts:
24    reports:
25     container_scanning: samples/container-scanning.json
26    expire_in: 16 weeks
27
28  dast:
29    stage: test
30    needs: [build-job]
31    allow_failure: true
32    interruptible: true
33    script:
34     - echo NOOP
35  artifacts:
36    reports:
37     dast: samples/dast.json
38    expire_in: 16 weeks
39
```

Results

- In contrast here are the various successful GitLab security tests and their artifacts.

Scan details					Hide details
DAST	4 vulnerabilities				Download results
SAST	60 vulnerabilities				Download results
Container Scanning	8 vulnerabilities				Download results
API Fuzzing	6 vulnerabilities				Download results
Coverage Fuzzing	1 vulnerability				Download results
Secret Detection	5 vulnerabilities				Download results

Severity		Tool		Hide dismissed	
All severities		All tools		☒	

☐	Severity	Vulnerability	Identifier	Tool	
☐	Critical	eval with argument of type Identifier src/html/index.html	ESLint rule ID security/detect-eval-with-expression + 1 more	SAST GitLab	☐ ⓘ ↗ ⌛
☐	Critical	Generic Object Injection Sink src/html/index.html	ESLint rule ID security/detect-object-injection + 1 more	SAST GitLab	☐ ⓘ ↗ ⌛
☐	Critical	eval with argument of type Identifier src/js/main.js	ESLint rule ID security/detect-eval-with-expression + 1 more	SAST GitLab	☐ ⓘ ↗ ⌛

Results

Deserialization of Untrusted Data

Description

A possible escalation to RCE vulnerability exists when using YAML serialized columns in Active Record < 7.0.3.1, <6.1.6.1, <6.0.5.1 and <5.2.8.1 which could allow an attacker, that can manipulate data in the database (via means like SQL injection), the ability to escalate to an RCE.

Severity: ● Critical

Project: Vulnerability_Management / DevSecOps_Security_Reports

Tool: Dependency Scanning

Scanner: Gemnasium

Location

File: dependency-scanning-files/Gemfile.lock

Links

- <https://discuss.rubyonrails.org/t/cve-2022-32224-possible-rce-escalation-bug-with-serialized-columns-in-active-record/81017>
- <https://github.com/advisories/GHSA-3hhc-qp5v-9p2j>
- <https://github.com/rails/rails/commit/611990f1a6c137c2d56b1ba06b27e5d2434dcd6a>
- <https://github.com/rubysec/ruby-advisory-db/blob/master/gems/activerecord/CVE-2022-32224.yml>
- <https://groups.google.com/g/rubyonrails-security/c/MmFO3LYQE8U>
- <https://nvd.nist.gov/vuln/detail/CVE-2022-32224>

Identifiers

- GHSA-3hhc-qp5v-9p2j
- CVE-2022-32224
- Gemnasium-b60c2d6b-9083-4a97-a1b2-f7dc79bff74c

Solution

Upgrade to versions 5.2.8.1, 6.0.5.1, 6.1.6.1, 7.0.3.1 or above.

- When you select a specific vulnerability you can see a window that looks like this talking about the cause and solution to fix.



Results

- This is the vulnerability report dashboard on GitLab that has a more executive level display of all vulnerabilities for a given project.

Vulnerability report + Submit vulnerability Export

The Vulnerability Report shows results of successful scans on your project's default branch, manually added vulnerability records, and vulnerabilities found from scanning operational environments. [Learn more.](#)

Development vulnerabilities 187 Operational vulnerabilities 0

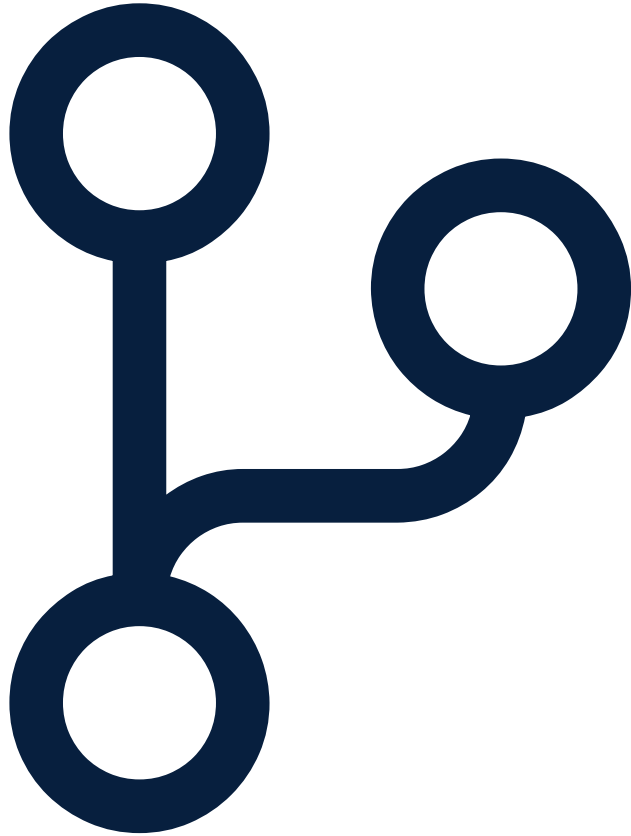
Security reports last updated 1 week ago #1070623396 ⚠ Parsing errors in pipeline • SBOMs last updated 1 week ago #1070623396

Critical	High	Medium	Low	Info	Unknown
17	61	83	7	0	19

Status: Needs triage +1 more Severity: All severities Tool: All tools Activity: All activity

Group by: None

☐	Detected	Status	Severity	Description	Identifier	Tool	Activity
☐	2023-11-13	Needs Triage	Critical	Deserialization of Untrusted Data dependency-scanning-files/Gemfile.lock	CVE-2022-32224 + 2 more	Dependency Scanning GitLab	



Wazuh

- Wazuh is a free and open source security platform that unifies XDR and SIEM capabilities.



Wazuh Components



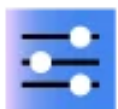
W. indexer

This component indexes and stores alerts generated by the Wazuh server



W. server

The Wazuh server analyzes data received from



W. dashboard

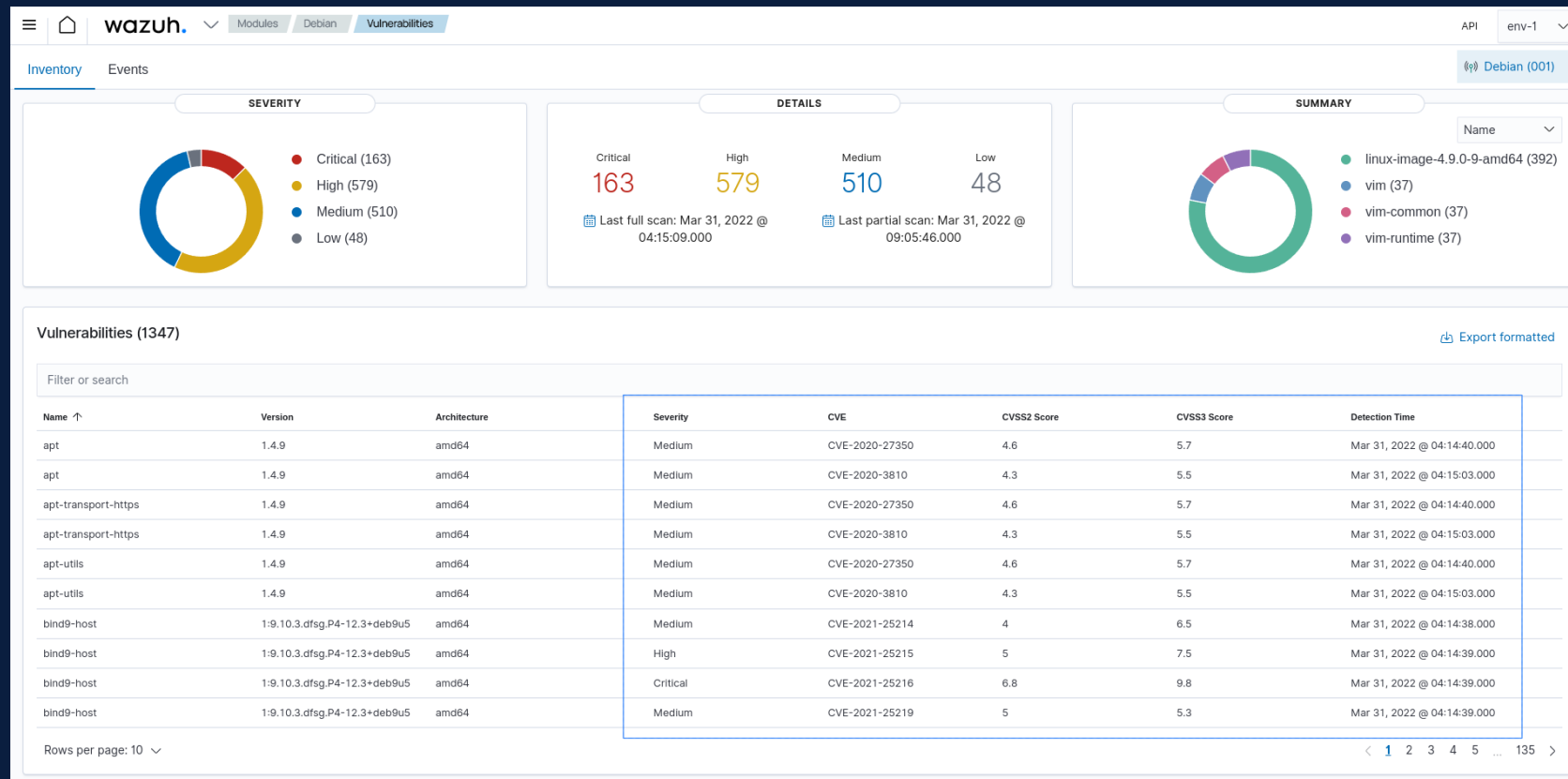
The Wazuh dashboard is the web user interface for data visualization, analysis, and management.

Wazuh Agent

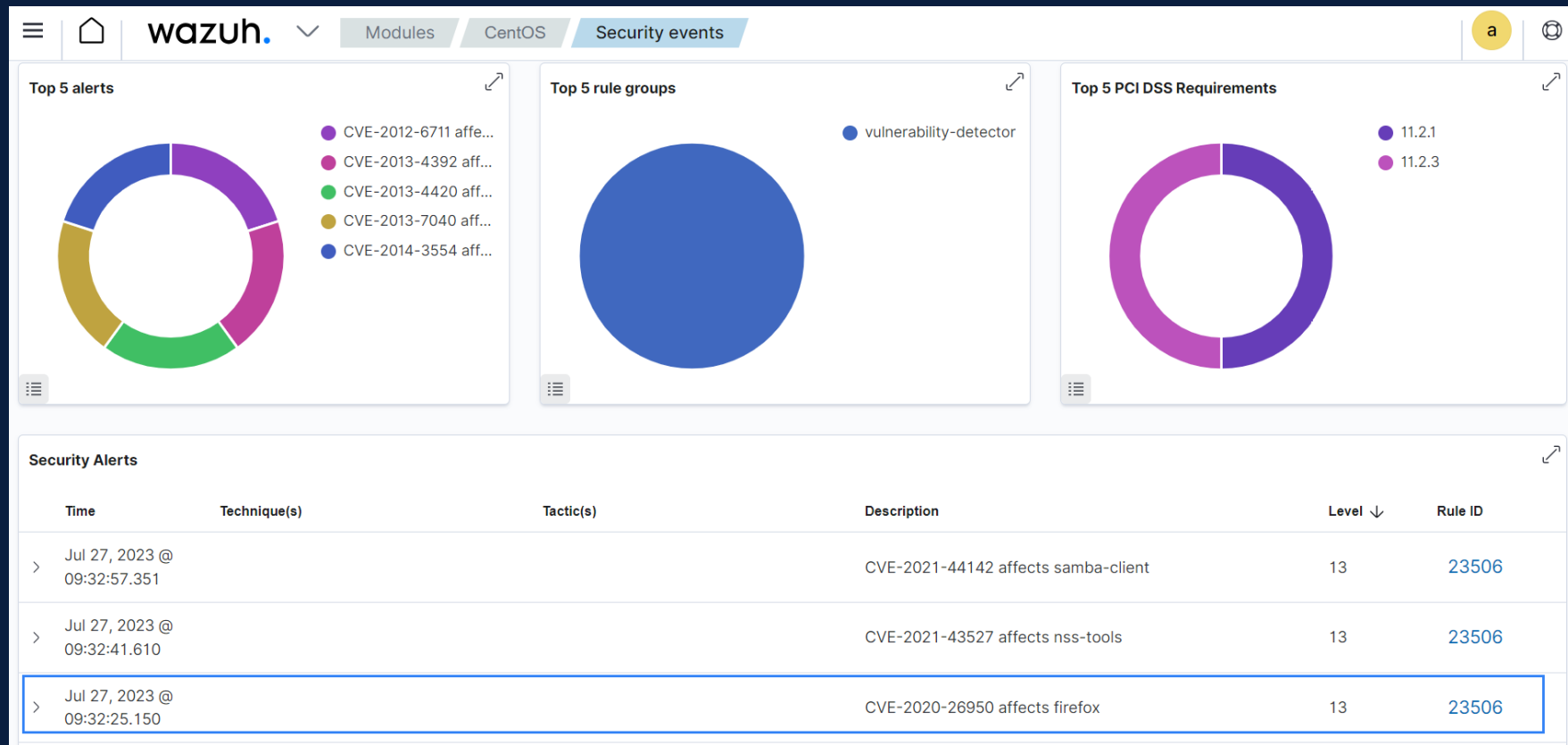
- Wazuh agents are installed on endpoints such as laptops, desktops, servers, cloud instances, or virtual machines. They provide threat prevention, detection, and response capabilities.



Vulnerability Detection



Vulnerability Alert



Track Remediation

Wazuh interface showing vulnerability tracking and remediation status.

Wazuh Vulnerabilities Table:

Time	data.vulnerability.package.name	data.vulnerability.cve	data.vulnerability.severity	data.vulnerability.status
Jul 19, 2023 @ 12:38:32.384	Windows 11	CVE-2021-40449	High	Solved
Jul 19, 2023 @ 12:38:32.373	Windows 11	CVE-2021-43217	Critical	Solved

Expanded document (JSON):

```
{  "_index": "wazuh-alerts-4.x-2023.07.19",  "agent.id": "011",  "agent.ip": "FE80:0000:0000:0000:80C4:2B1A:5A30:8E5E",  "agent.name": "Windows 11",  "data.vulnerability.cve": "CVE-2021-43217",  "data.vulnerability.cvss.cvss2.base_score": 7.500000,  "data.vulnerability.cvss.cvss3.base_score": 9.800000,  "data.vulnerability.package.architecture": "x64",  "data.vulnerability.package.name": "Windows 11",  "data.vulnerability.package.version": "10.0.22621.1848",  "data.vulnerability.published": "Dec 15, 2021 @ 02:00:00.000",  "data.vulnerability.references": "https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2021-43217, https://nvd.nist.gov/vuln/detail/CVE-2021-43217",  "data.vulnerability.severity": "Critical",  "data.vulnerability.status": "Solved",  "data.vulnerability.title": "CVE-2021-43217 affecting Windows 11 was solved",  "data.vulnerability.type": "OS",  "data.vulnerability.updated": "Jul 12, 2022 @ 03:00:00.000",  "decoder.name": "json"}
```





Risk Assessment for Greenbone

- Creating and modifying Network Vulnerability Tests (NVTs) within the Greenbone system involves careful consideration of potential security risks, such as the inadvertent introduction of vulnerabilities if not developed and tested rigorously. This necessitates thorough code reviews, extensive testing, and adherence to secure coding practices. Compatibility issues must be addressed through diverse testing across different environments, and documentation should clearly outline system requirements. Maintenance challenges, including the risk of outdated custom tests, require a regular review process to ensure ongoing effectiveness. Mitigating false positives or negatives involves continuous testing and collaboration with the open-source community for refinement based on real-world feedback. Additionally, resource-intensive aspects of development and maintenance need efficient management, with priorities aligned to risk assessment. When adapting tests to specific circumstances, a systematic approach involves identifying security needs, engaging stakeholders, designing custom tests, comprehensive testing, clear documentation, and a continuous improvement cycle. Sharing these adaptations with the open-source community fosters collaboration and benefits a broader audience facing similar challenges.





Risk Assessment for GitLab DevSecOps

- Running custom scans as part of your software development or security processes carries both benefits and risks. On the positive side, custom scans provide the flexibility to tailor security assessments according to specific project requirements, helping identify vulnerabilities that might be unique to your application or environment. However, these custom scans introduce potential risks, including the possibility of false positives or negatives if not properly configured or validated. Additionally, there's a risk of inadvertently impacting the stability of the system during the scanning process. The effectiveness of custom scans heavily depends on the accuracy of the scan configurations, and any misconfigurations may lead to overlooking actual security threats or triggering unnecessary alarms. Therefore, a thorough validation and testing process for custom scans, along with regular updates to account for evolving security threats, are essential to mitigate these risks and ensure the reliability of the overall security assessment process.





Risk Assessment for Wazuh

- Integrating Wazuh into your security infrastructure for scanning purposes offers advantages in terms of threat detection and monitoring but also presents certain risks. Wazuh, as an open-source security information and event management (SIEM) tool, provides capabilities for log analysis, intrusion detection, vulnerability detection, and more. While it enhances your ability to identify potential security incidents, misconfigurations or incomplete deployment of Wazuh could result in false positives or, conversely, miss actual threats. Additionally, the effectiveness of Wazuh scans depends on timely updates of its rules and configurations to address emerging threats. There's also a risk of increased system resource utilization during intensive scanning periods, potentially impacting overall system performance. Rigorous testing, regular updates, and careful configuration management are crucial for mitigating these risks and ensuring Wazuh contributes effectively to your security posture without introducing unnecessary disruptions or overlooking genuine security threats.

