LINUX SECURITY

part_4.sh

Author: aimery de La Roncière @ aimerydlr
University: ECE Paris – Ing5 CYB
YEAR: 2025-2026

Table of contents

I. Vulnerability assessment and Patch Management

Vulnerability assessment and patch management are essential pillars of a proactive cybersecurity strategy. They ensure that known security weaknesses are identified, assessed, and remediate in a timely manner to reduce the risk of exploitation. This chapter presents an overview of the fundamental standards and databases that catalog vulnerabilities, explores leading automated vulnerability assessment tools, and covers best practices for managing software updates and IT assets.

I.1. Technical Standards and Vulnerability Databases

Common Weakness Enumeration (CWE)

- Created: 2006
- Maintained by: MITRE Corporation
- URL: https://cwe.mitre.org/

CWE is a formal catalog of common software weaknesses and programming errors. Unlike specific vulnerabilities, CWE focuses on the types of flaws that cause vulnerabilities, such as buffer overflows, cross-site scripting, or improper input validation. By identifying these generic weakness classes, developers and security analysts can understand root causes and improve secure coding practices. CWE is widely used to guide static analysis tools, security training, and vulnerability research.

Example use: When a vulnerability is reported in CVE, it often references a CWE to describe the underlying weakness category.

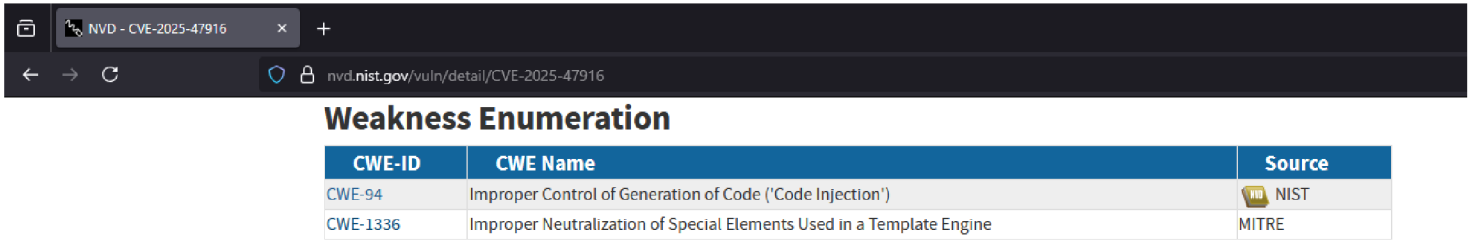

Figure 1 - Related CWE for CVE-2025-47916

Source: https://nvd.nist.gov/vuln/detail/CVE-2025-47916

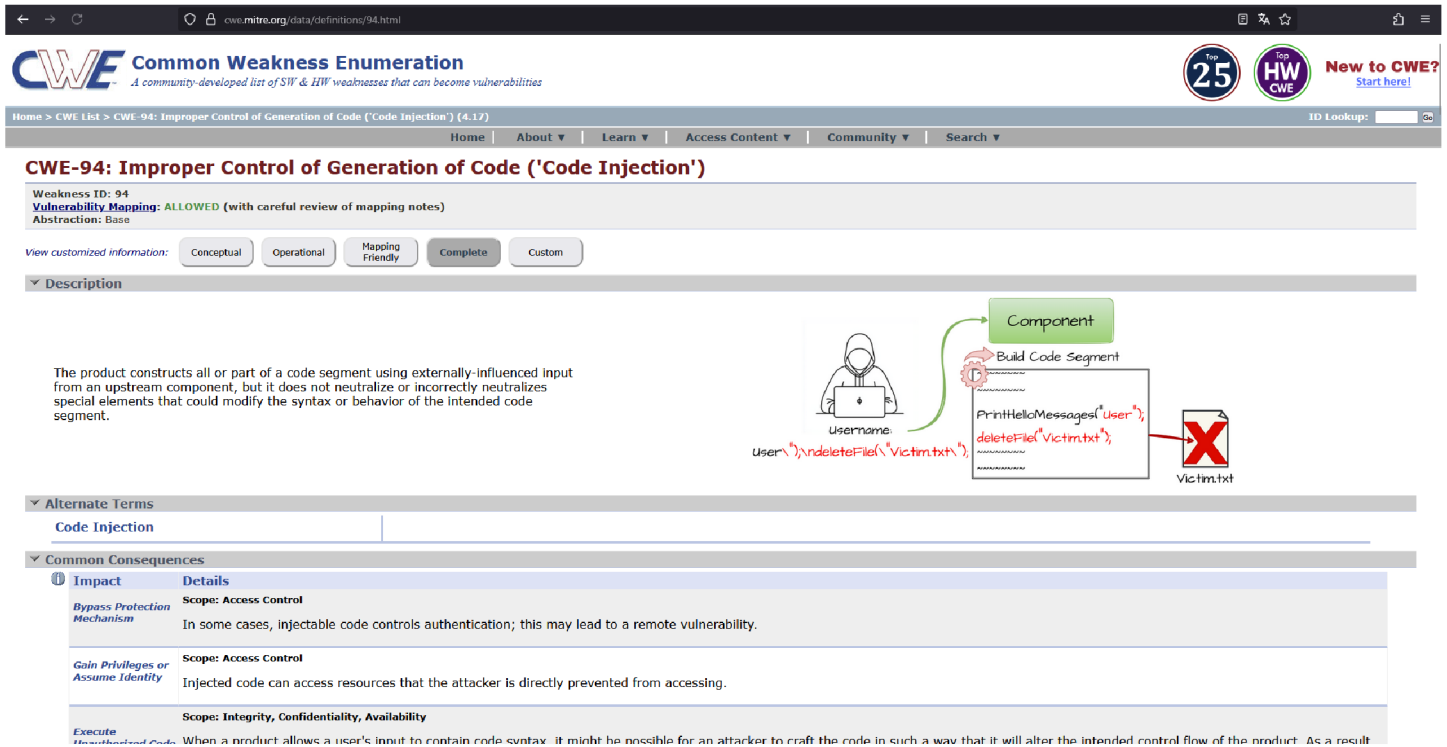

Figure 2 - CWE-94: Improper Control of Generation of Code ('Code Injection')

Source: https://cwe.mitre.org/data/definitions/94.html

Common Vulnerabilities and Exposures (CVE)

- Created: 1999
- Maintained by: MITRE Corporation
- URL: https://www.cve.org/

CVE is the industry-standard catalog that assigns unique identifiers to individual, publicly disclosed cybersecurity vulnerabilities and exposures. Each CVE entry (e.g., CVE-2025-47916) represents a specific vulnerability in a software product or hardware device. CVE serves as a global naming standard, enabling security tools and vendors to share consistent information about vulnerabilities. CVE itself does not provide detailed vulnerability metrics but acts as an index for referencing vulnerability reports, advisories, and patches.

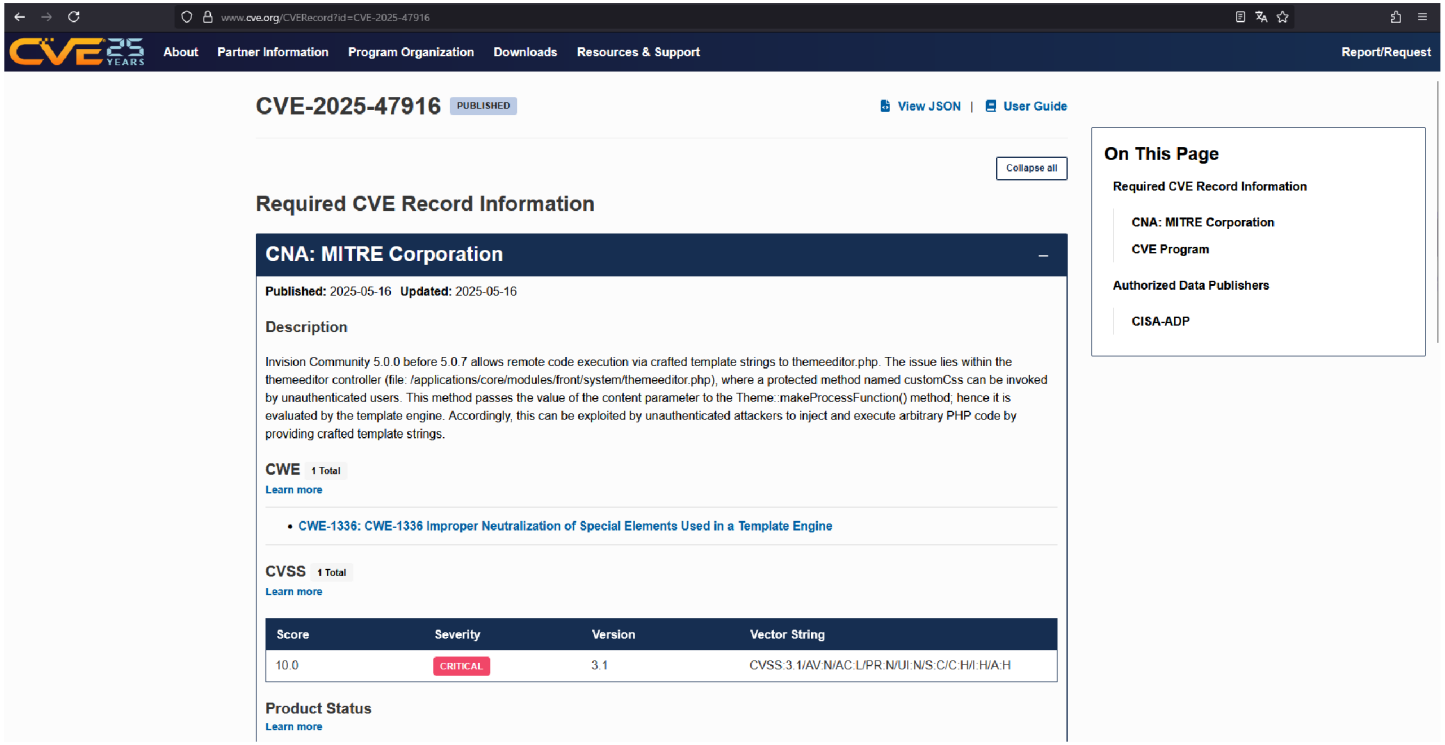

Figure 3 - CVE-2025-47916

Source: https://www.cve.org/CVERecord?id=CVE-2025-47916

National Vulnerability Database (NVD)

- Created: 2005
- Maintained by: National Institute of Standards and Technology (NIST)
- URL: https://nvd.nist.gov/

The NVD supplements CVE by providing enhanced metadata, including severity scores (CVSS), impact metrics, and fix information. It is a comprehensive repository of vulnerability management data used by government agencies, enterprises, and security tools to automate risk assessment and compliance. The NVD also publishes vulnerability feeds consumable by scanners and patch management systems.

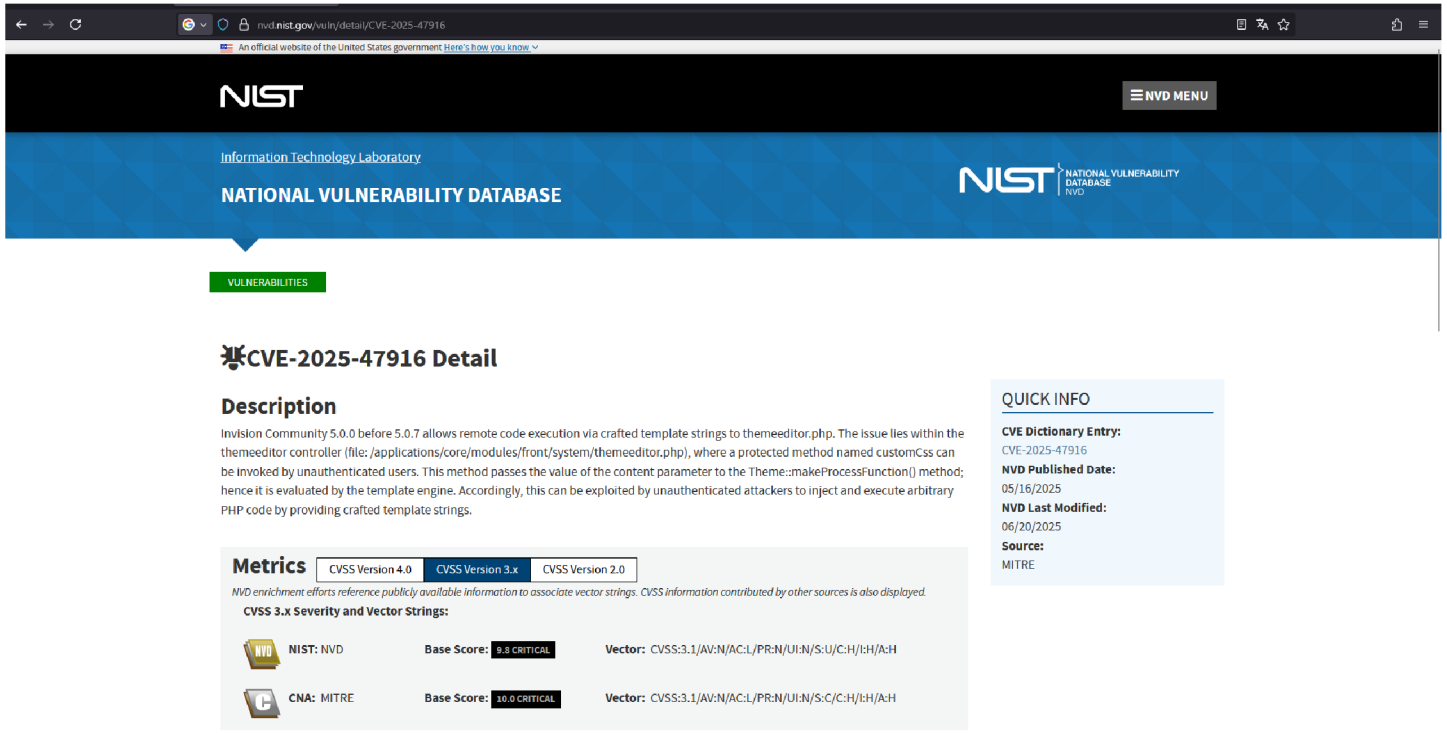

Figure 4 - NVD details for CVE-2025-47916

Source: https://nvd.nist.gov/vuln/detail/CVE-2025-47916

Common Vulnerability Scoring System (CVSS)

- Created: 2005
- Maintained by: FIRST (Forum of Incident Response and Security Teams)
- URL: https://www.first.org/cvss/

CVSS is a standardized framework to quantify the severity of vulnerabilities on a numeric scale from 0 (low) to 10 (critical). This scoring facilitates objective risk prioritization and consistent communication across teams.

Structure of the CVSS Score

- Base Metrics (Required)

These reflect the intrinsic properties of the vulnerability, independent of environment or time. This is the core of CVSS and always required.

Metric	Description	Values
Attack Vector (AV)	How the vulnerability is exploited	Network (N), Adjacent (A), Local (L), Physical (P)
Attack Complexity (AC)	Conditions beyond the attacker's control required?	Low (L), High (H)
Privileges Required (PR)	Level of privileges needed by the attacker	None (N), Low (L), High (H)
User Interaction (UI)	Is user interaction required?	None (N), Required (R)
Scope (S)	Can the exploit affect other system components?	Unchanged (U), Changed (C)
Confidentiality (C)	Loss of confidentiality?	None (N), Low (L), High (H)
Integrity (I)	Loss of data integrity?	None, Low, High
Availability (A)	Loss of availability?	None, Low, High

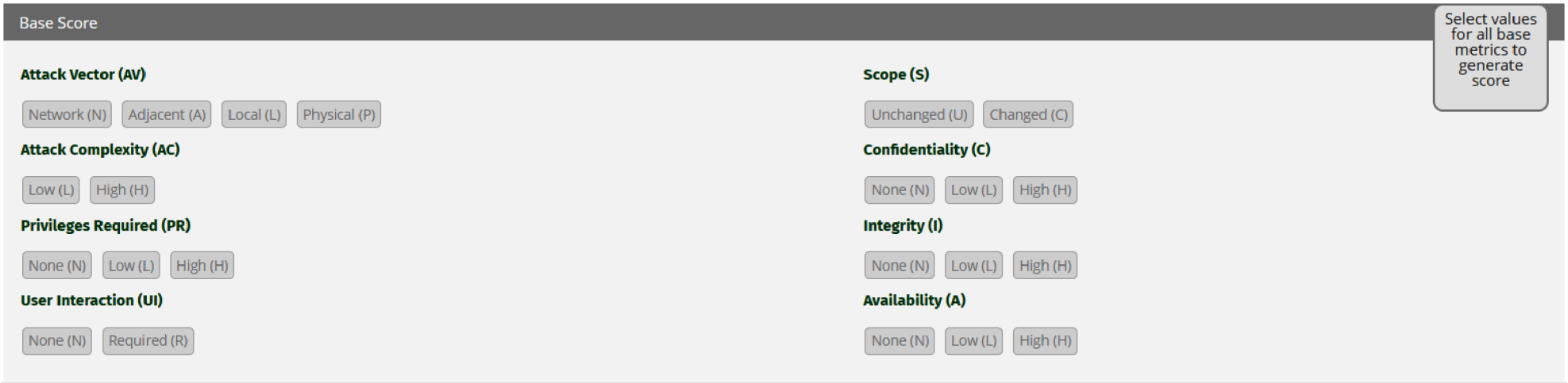

Figure 5 - Base Score CVSS calculator

Source: https://www.first.org/cvss/calculator/3-1

- Temporal Metrics (Optional)

These account for factors that change over time, such as exploit availability and patch status.

Metric	Description
Exploit Code Maturity (E)	Is there an exploit in the wild?
Remediation Level (RL)	Is a fix available and what kind?
Report Confidence (RC)	How reliable is the vulnerability report?

Figure 6 - Temporal Score CVSS calculator

Source: https://www.first.org/cvss/calculator/3-1

- Environmental Metrics (Optional)

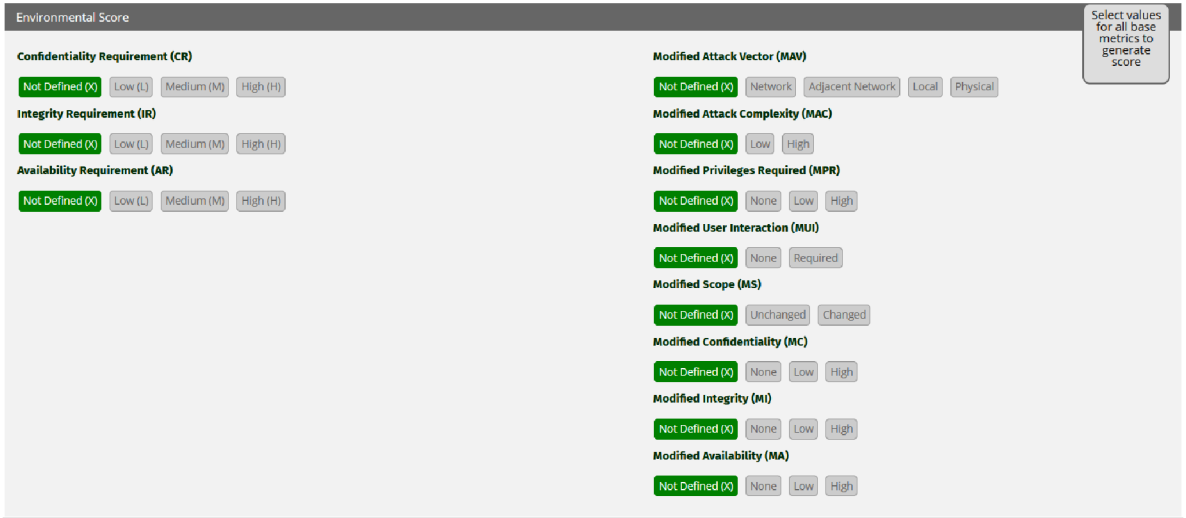

Figure 7 - Environmental Metrics CVSS calculator

Source: https://www.first.org/cvss/calculator/3-1

Score Interpretation

Score Range	Severity
0.0	None
0.1 – 3.9	Low
4.0 – 6.9	Medium
7.0 – 8.9	High
9.0 – 10.0	Critical

Version History

Version	Release Date	Key Features
CVSS v1	2005	Initial release. Adopted by NVD. Limited adoption due to design flaws and lack of flexibility.
CVSS v2	2007	Introduced Base, Temporal, and Environmental metric groups. Widely adopted (especially in the NVD and CVE ecosystem).
CVSS v3.0	June 2015	Improved granularity and accuracy. Added **Scope**, reworked privilege metrics. Better suited for modern threats.
CVSS v3.1	June 2019	Clarified metric definitions. Improved documentation and scoring guidance. Became the most used version to date.
CVSS v4.0	October 2023	Major overhaul. Introduced **Supplemental Metrics** and **Exploitability**, **Automatable**, and **Recovery** metrics. Designed to better align with EPSS and modern threat intelligence. **Not yet adopted by all tools.**

Key Limitations

- CVSS does not assess exploit activity (use EPSS or LEV for that)
- It does not account for exposure (e.g., vulnerability behind a firewall)
- The Base Score is generic and may not reflect actual risk in your context

Exploit Prediction Scoring System (EPSS)

- Created: 2020
- Maintained by: FIRST and community contributors
- URL: https://www.first.org/epss/

EPSS is a predictive model estimating the likelihood that a given vulnerability will be actively exploited within the next 30 days. By analyzing historical exploit trends, vulnerability attributes, and threat intelligence data, EPSS helps security teams anticipate and prioritize patches based not only on severity but on exploitation probability. EPSS complements CVSS by adding a temporal risk factor to vulnerability management.

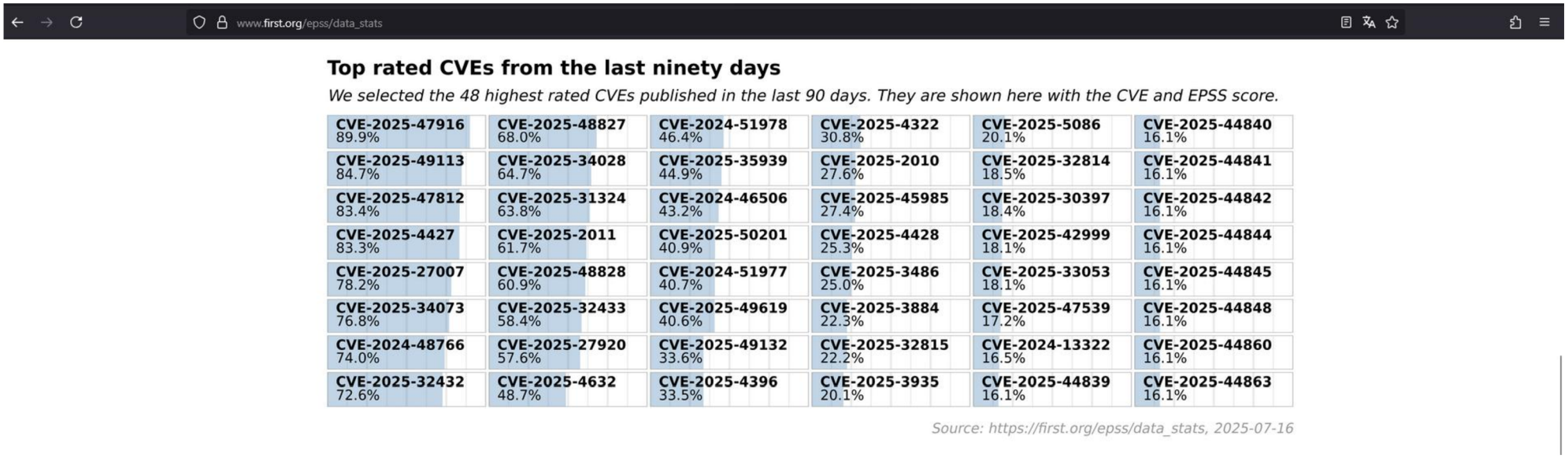

Figure 8 - EPSS score for top rated CVE

Source: https://www.first.org/epss/data_stats

MITRE ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge)

- Created: 2013
- Maintained by: MITRE Corporation
- URL: https://attack.mitre.org/

ATT&CK is a knowledge base of adversary tactics and techniques derived from real-world cyber attacks. It describes the behavior and methods attackers use to infiltrate, move laterally, escalate privileges, and exfiltrate data. ATT&CK provides a common language for threat hunting, detection, and defense and helps security teams build effective detection rules and incident response strategies. Unlike CVE or CWE, ATT&CK focuses on attack behaviors rather than individual vulnerabilities. The ATT&CK framework is structured based on observations from real-world incidents and publicly available threat intelligence. It is updated continuously with contributions from the cybersecurity community.

MITRE ATT&CK is often mapped with:

- CVEs (which identify vulnerabilities)
- NIST frameworks
- Sigma, YARA, and OSQuery rules

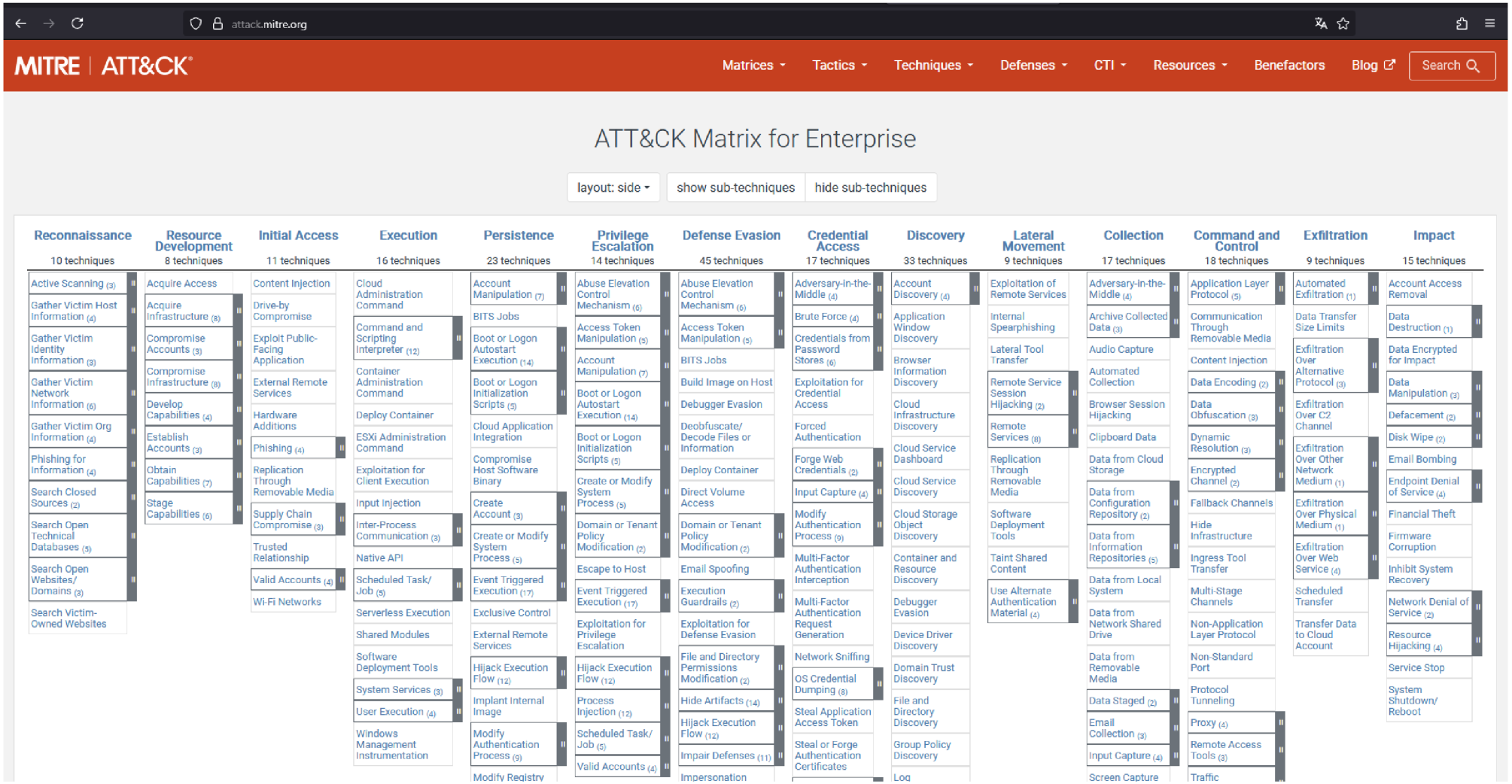

Figure 9 - MITRE ATT&CK main website page

Source: https://attack.mitre.org/

MITRE D3FEND

- Created: 2021
- Maintained by: MITRE Corporation
- URL: https://d3fend.mitre.org/

D3FEND is a knowledge base of cybersecurity defensive techniques and countermeasures, designed as a complement to ATT&CK. It catalogs methods defenders can implement to protect systems against attacker techniques, such as sandboxing, privilege separation, or encryption. D3FEND assists security architects in designing robust defense strategies grounded in standardized concepts.

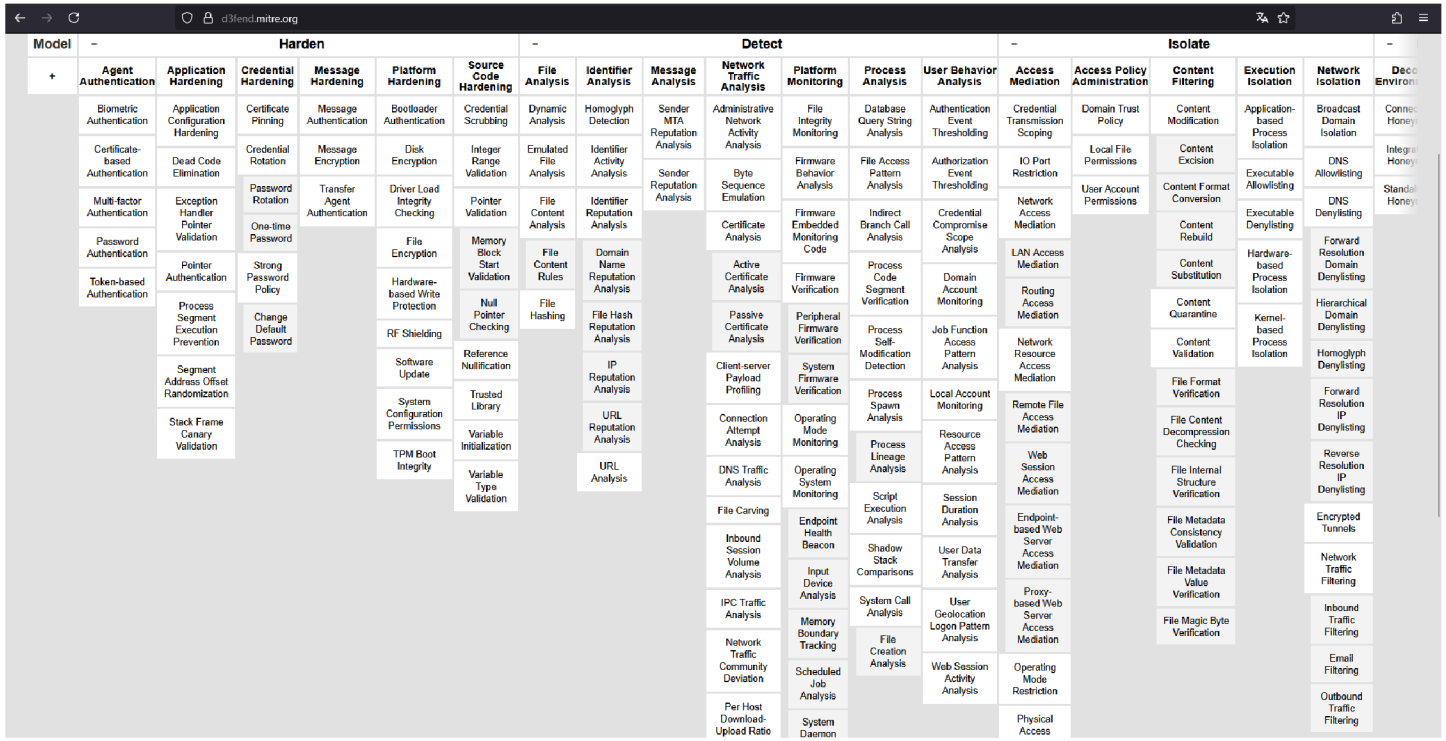

Figure 10 - MITRE D3FEND main website page

Source: https://d3fend.mitre.org

CPE (Common Platform Enumeration)

- Created: 2006
- Maintained by: MITRE Corporation
- URL: https://nvd.nist.gov/products/cpe

CPE is a standardized method for naming and identifying classes of applications, operating systems, and hardware devices. It enables consistent referencing of products across various security tools and databases.

A CPE name uniquely identifies a product using a structured URI format:

```
cpe:2.3:<part>:<vendor>:<product>:<version>:<update>:<edition>:<language>:<sw_edition>:<target_sw>:<target_hw>:<other>
```

Example:

```
cpe:2.3:a:apache:http_server:2.4.51:*:*:*:*:*:*:*
```

This entry would refer to Apache HTTP Server version 2.4.51.

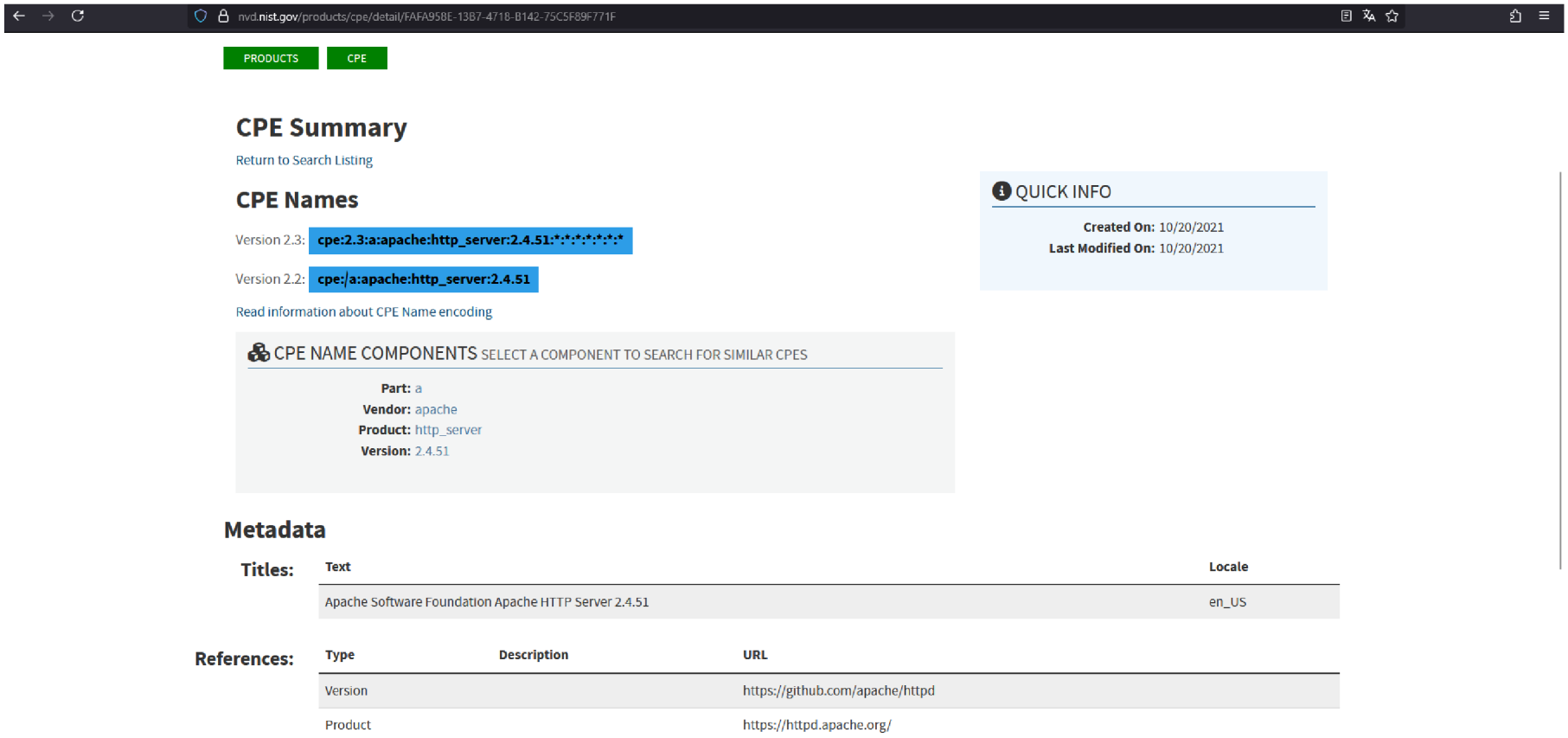

Figure 11 - CPE détails for apache server 2.4.51

Source: https://nvd.nist.gov/products/cpe/detail/FAFA958E-13B7-4718-B142-75C5F89F771F

Security Content Automation Protocol (SCAP)

- Created: 2007
- Maintained by: National Institute of Standards and Technology (NIST)
- URL: https://csrc.nist.gov/projects/security-content-automation-protocol

SCAP is a suite of open standards that enable automated vulnerability management, policy compliance, and security measurement. It integrates various specifications such as CVE, CPE (Common Platform Enumeration), and CVSS. SCAP allows security tools to exchange standardized data formats for automated scanning, assessment, and reporting, improving interoperability between scanners and security products.

OpenSCAP – Open Security Content Automation Protocol

OpenSCAP is an open-source framework designed to assess and enforce system security and compliance. It provides a free implementation of the SCAP (Security Content Automation Protocol) standards. OpenSCAP enables system administrators and security professionals to automate security audits and verify that systems conform to established security baselines.

Documentation: https://github.com/OpenSCAP/openscap

- Key Features
  - *Compliance auditing*: Evaluate systems against benchmarks like CIS, PCI-DSS, DISA STIGs.
  - *Vulnerability scanning*: Detect known issues based on CVEs and OVAL definitions.
  - *Security baselining*: Validate configurations across multiple hosts to ensure standardization.
  - *Remediation scripts*: Generate automated fixes for non-compliant systems.

I.2. Automated vulnerability assessment - OpenVAS

Vulnerability assessment is the process of identifying, classifying, and prioritizing security weaknesses in systems, applications, and networks. It forms a critical part of a defense-in-depth strategy, allowing administrators to detect misconfigurations and software vulnerabilities before attackers can exploit them. However, *it's important to note that vulnerability assessments do not replace penetration testing*. While vulnerability assessments focus on identifying and categorizing potential vulnerabilities, penetration testing involves actively exploiting those vulnerabilities to understand the real-world impact and effectiveness of security measures. Both are essential components of a comprehensive security strategy.

OpenVAS (Open Vulnerability Assessment Scanner) is a powerful, free, and open-source vulnerability scanning framework. It is part of the Greenbone Vulnerability Management (GVM) suite and provides automated scanning, reporting, and compliance tools suitable for enterprise environments.

Documentation:

- https://www.openvas.org/
- https://github.com/greenbone/

How OpenVAS Works

OpenVAS uses a database of Network Vulnerability Tests (NVTs) written in NASL (Nessus Attack Scripting Language). The scanner performs a variety of checks including:

- Port scanning (TCP/UDP)
- Service and version detection
- Operating system fingerprinting
- Web application vulnerability checks
- CVE-based software vulnerability scans
- Credentialed and non-credentialed scanning
- The scan results are scored using the CVSS (Common Vulnerability Scoring System) and classified by severity (Low, Medium, High, Critical).

Architecture Overview

OpenVAS typically includes the following components:

- OpenVAS Scanner – the engine that performs vulnerability scans
- GVMd (Greenbone Vulnerability Manager daemon) – manages scan tasks and user permissions
- gvmd-cli / GSA (Greenbone Security Assistant) – web and CLI interfaces for managing scans and results
- PostgreSQL database – stores scan results and configurations
- NVT Feed – updated regularly with vulnerability test scripts from Greenbone

Typical Use Cases

- Internal network scanning to identify vulnerable machines and services
- Audit of exposed services on perimeter firewalls and DMZ hosts
- Web application assessments for known CVEs
- Compliance checks (e.g., PCI-DSS, ISO 27001, CIS benchmarks)
- Integration into CI/CD pipelines for DevSecOps (with CLI or API)

Screenshots

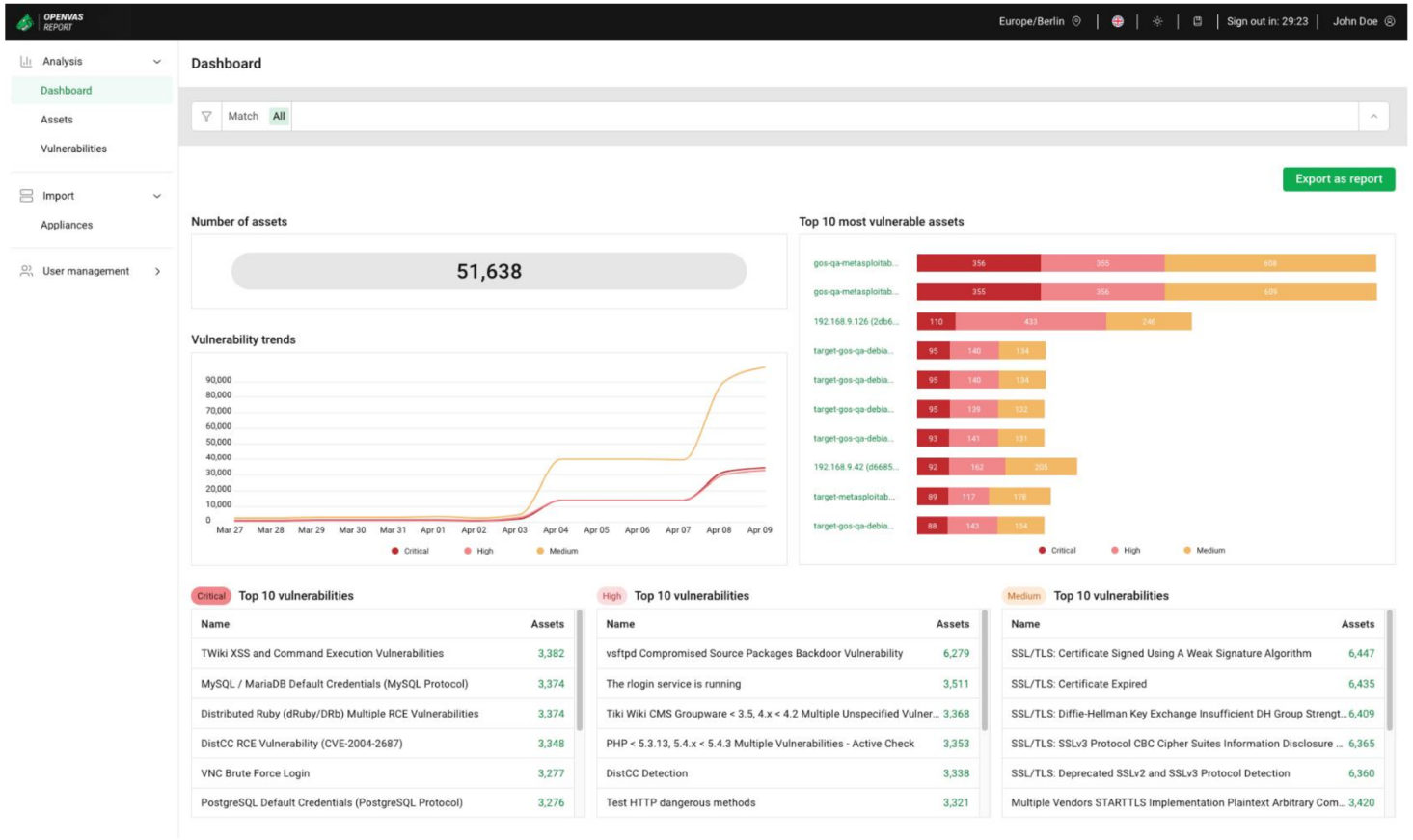

Figure 12 - OpenVAS dashboard

Source: https://docs.greenbone.net/OPENVAS-REPORT-Manual/en/_images/dashboard.png

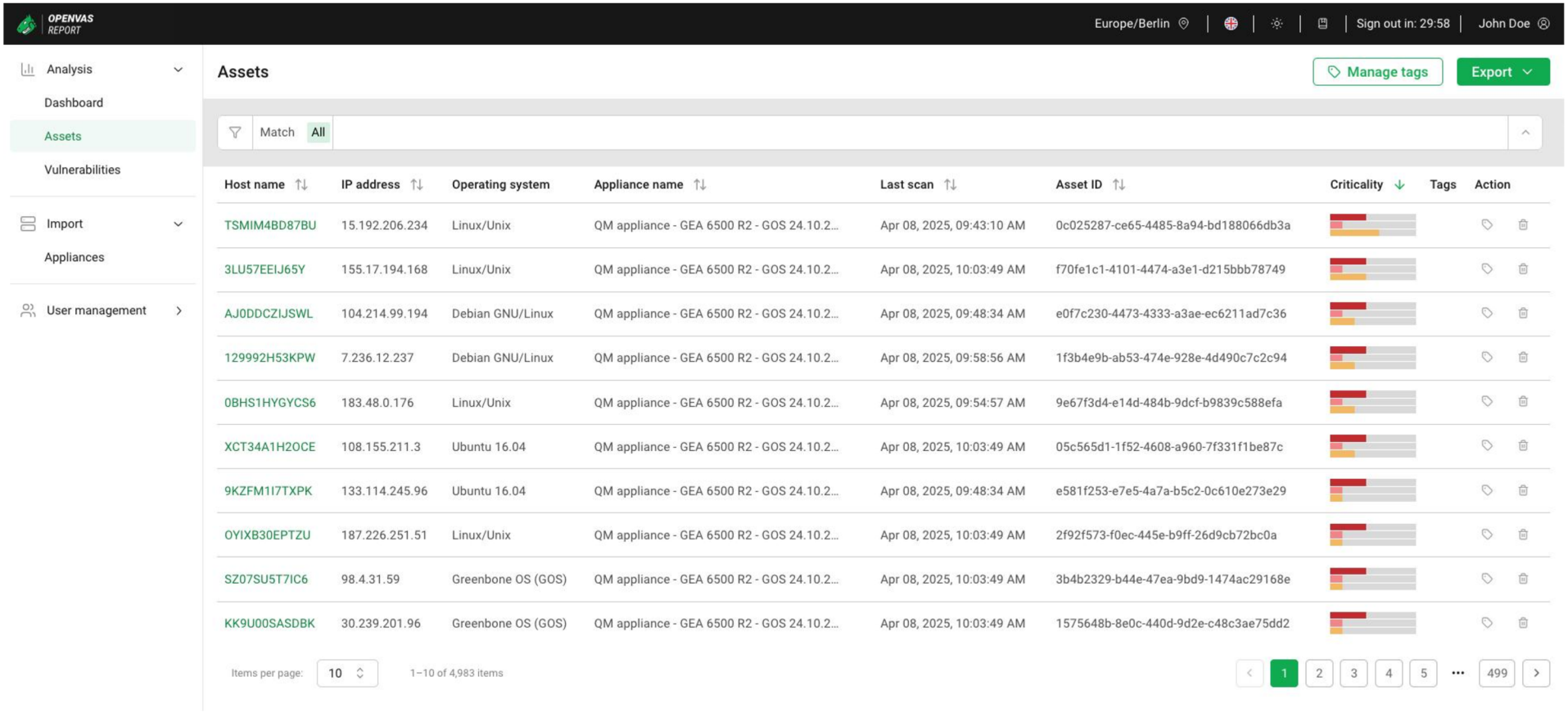

Figure 13 - OpenVAS Asset details

Source: https://docs.greenbone.net/OPENVAS-REPORT-Manual/en/_images/asset-details.png

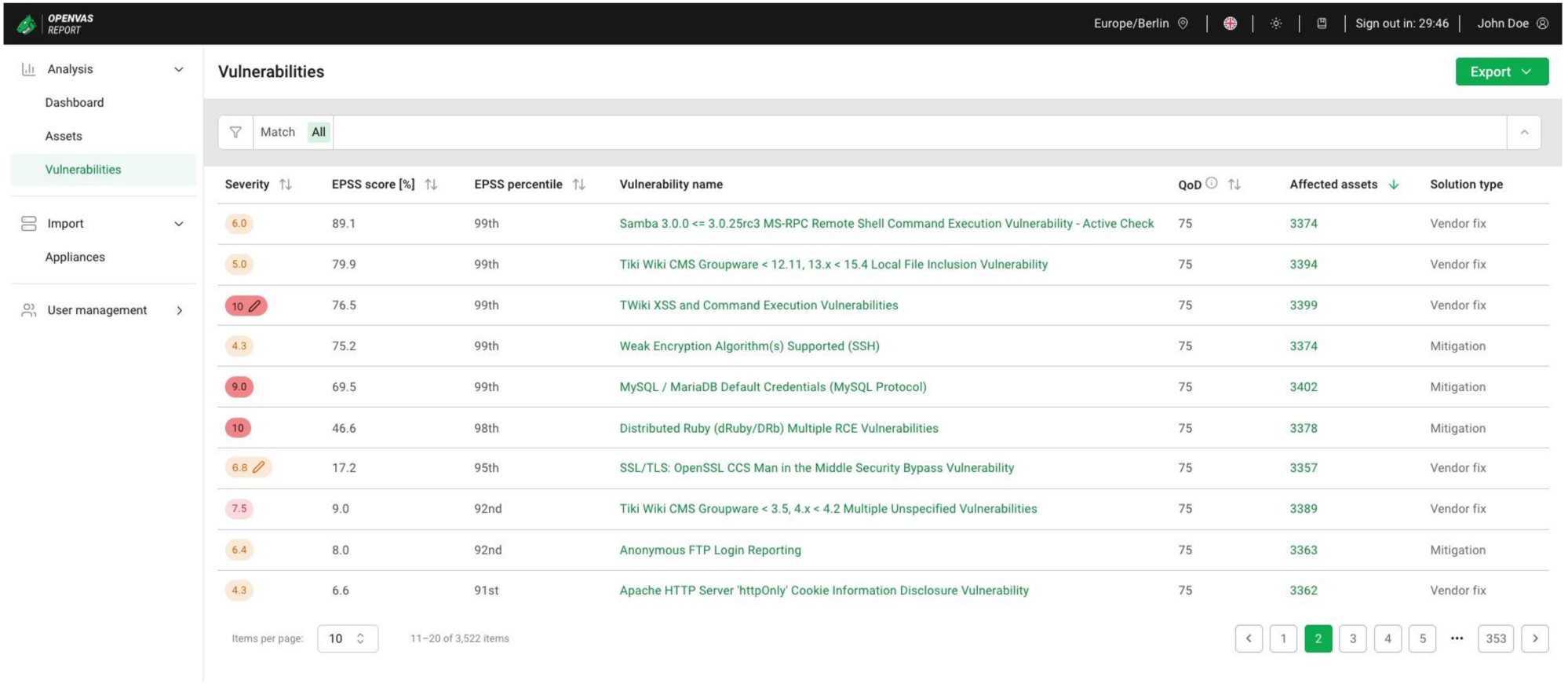

Figure 14 - OpenVAS vulnerabilities page

Source: https://docs.greenbone.net/OPENVAS-REPORT-Manual/en/_images/vulnerabilities-overview.png

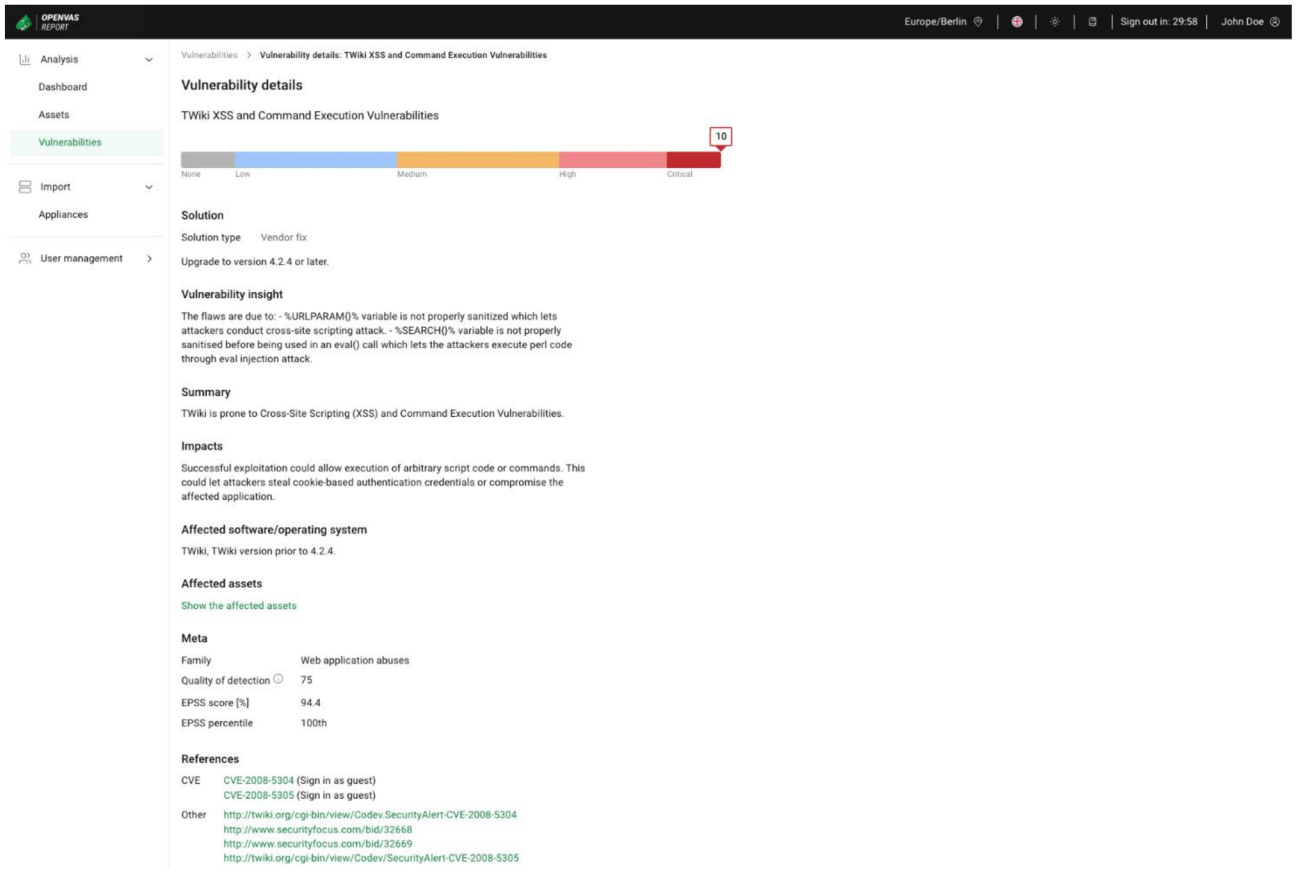

Figure 15 - OpenVAS vulnerability details

Source: https://docs.greenbone.net/OPENVAS-REPORT-Manual/en/_images/vulnerability-details.png

I.3. Automated vulnerability assessment - Nuclei

Nuclei is a fast and flexible open-source vulnerability scanner developed by ProjectDiscovery. It allows security professionals and system administrators to automatically detect vulnerabilities using YAML-based templates, making it highly customizable and scriptable for integration into CI/CD pipelines, asset discovery or routine assessments.

Unlike traditional scanners, Nuclei is template-driven and focuses on speed, modularity, and low false-positive rates.

Documentation:

- https://github.com/projectdiscovery/nuclei
- https://projectdiscovery.io/nuclei

Important Directories and Files

- *~/.local/nuclei-templates/*: Default location for downloaded templates
- *~/.config/nuclei/*: Stores metadata like update status, cache

Core Concept: Templates

At the heart of Nuclei lies the template system. Template structure:

- A unique ID for the template
- Essential information and metadata relevant to the template
- The designated protocol, such as HTTP, DNS, File, etc.
- Details specific to the chosen protocol, like the requests made in the HTTP protocol
- A series of matchers to ascertain the presence of findings
- Necessary extractors for data retrieval from the results

```
acrapx ❯❯ cat CVE-2025-0107.yaml
id: CVE-2025-0107

info:
  name: Palo Alto Networks Expedition - OS Command Injection
  author: iamnooooob,pdresearch
  severity: critical
  description: |
    An OS command injection vulnerability in Palo Alto Networks Expedition enables an unauthenticated attacker to run arbitrary OS commands as the www-data user in Expedition, which results in the
words, device configurations, and device API keys for firewalls running PAN-OS software.
  reference:
    - https://security.paloaltonetworks.com/PAN-SA-2025-0001
    - https://ssd-disclosure.com/ssd-advisory-palo-alto-expedition-rce-regionsdiscovery/
    - https://nvd.nist.gov/vuln/detail/CVE-2025-0107
  classification:
    cvss-metrics: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
    cvss-score: 9.8
    epss-score: 0.22791
    epss-percentile: 0.95559
  metadata:
    verified: true
    max-request: 1
    shodan-query: title:"Expedition"
    fofa-query: title=="Expedition Project"
  tags: cve,cve2025,rce,paloalto,expedition

http:
  - raw:
      - |
        GET /API/regionsDiscovery.php?master=spark%3A%2F%2F{{interactsh-url}}:443&mask=26&project=your_project&devices=device1%2Cdevice2&mtserver=127.0.0.1%3A3306&mtuser=root&mtpassword=paloalto&ta
arquetPath=%2Ftmp&timezone=Europe%2FHelsinki&mlserver=127.0.0.1&debug=false&initDate=2023-01-01&endDate=2023-01-31 HTTP/1.1
        Host: {{Hostname}}

    matchers-condition: and
    matchers:
      - type: word
        part: body
        words:
          - 'msg":"Started'
          - '"success":true'
        condition: and

      - type: word
        part: interactsh_protocol
        words:
          - "dns"
# digest: 4a0a00473045022052d4c16374943beb476a248bc8f298132222dbcc5c7378cacc96abdd87a13c7202210094f8be4d4533a0c6989e8b589337d6a6e34e17104d5f31f336eaf9994984899f:922c64590222798bb761d5b6d8e72950
```

Figure 16 - Nuclei template example

Nuclei allows you to write your own templates to scan for internal vulnerabilities or undocumented services specific to your environment.

Nuclei installation

```
#First install go
go install -v github.com/projectdiscovery/nuclei/v3/cmd/nuclei@latest

cp /root/go/bin/nuclei /usr/local/go/bin/
```

Update Nuclei binary

Keeping Nuclei and its templates up to date is essential to ensure accurate and comprehensive vulnerability scanning.

```
nuclei -update # Updates the Nuclei engine (binary) to the latest version.
```

Incremental template update

```
nuclei -ut # Incrementally updates only the changed or new templates from the official repository.
```

Full reset of templates (hard refresh)

```
# Deletes the current templates and clones a fresh copy of the official repository.
rm -rf ~/nuclei-templates/ 2>/dev/null
git clone https://github.com/projectdiscovery/nuclei-templates ~/nuclei-templates/
```


Common Nuclei Usage Scenarios

The nuclei CLI tool provides flexible scanning capabilities across various targets and vulnerability categories. Below are typical usage patterns to perform CVE scans, target internal assets, apply custom templates, filter by severity, or control output format and rate:

```
# Scan a single target URL using all templates from default location
nuclei -u https://example.com

# Scan a single target URL using all CVE templates
nuclei -u https://example.com -t cves/

# Use custom internal templates on a target inside the local infrastructure
nuclei -u https://internal.app.local -t /opt/nuclei/custom-templates/

# Scan a list of targets for high and critical severity vulnerabilities
nuclei -list targets.txt -t vulnerabilities/ -severity critical,high

# Detect information exposure vulnerabilities and save output in JSON
nuclei -u https://example.com -t exposures/ -json -o exposures.json

# Perform DNS-related checks on multiple domains with a request rate limit
nuclei -list domains.txt -t dns/ -rate-limit 25
```


Nuclei scan example

```
acrapx ❯❯ nuclei -u https://google.fr

                     __     _
   ____  __  _______/ /__  (_)
  / __ \/ / / / ___/ / _ \/ /
 / / / / /_/ / /__/ /  __/ /
/_/ /_/\__,_/\___/_/\___/_/   v3.4.7

                projectdiscovery.io

[WRN] Found 7 templates with runtime error (use -validate flag for further examination)
[INF] Current nuclei version: v3.4.7 (latest)
[INF] Current nuclei-templates version: v10.2.5 (latest)
[WRN] Scan results upload to cloud is disabled.
[INF] New templates added in latest release: 45
[INF] Templates loaded for current scan: 8233
[INF] Executing 8031 signed templates from projectdiscovery/nuclei-templates
[WRN] Loading 202 unsigned templates for scan. Use with caution.
[INF] Targets loaded for current scan: 1
[INF] Templates clustered: 1769 (Reduced 1662 Requests)
[INF] Using Interactsh Server: oast.me
[missing-sri] [http] [info] https://www.google.com/ ["https://www.gstatic.com/og/_/js/k=og.asy.en_US.0PIB_wh3sEw.2019.O/rt=j/m=_ac,_awd,ada,lldp,qads,abld/exm=/d=1/ed=1/rs=AA2YrTs88ik
=ll_tdm,adcgm3,ll_fw,abld/excm=/d=1/ed=1/ct=zgms/rs=AA2YrTvGYOqmiGMk_fDQHBe8bqDJs4CmqA"]
[weak-cipher-suites:tls-1.0] [ssl] [low] google.fr:443 ["[tls10 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA]"]
[tls-version] [ssl] [info] google.fr:443 ["tls10"]
[deprecated-tls:tls_1.1] [ssl] [info] google.fr:443 ["tls11"]
[weak-cipher-suites:tls-1.1] [ssl] [low] google.fr:443 ["[tls11 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA]"]
[deprecated-tls:tls_1.0] [ssl] [info] google.fr:443 ["tls10"]
[tls-version] [ssl] [info] google.fr:443 ["tls11"]
[tls-version] [ssl] [info] google.fr:443 ["tls12"]
[tls-version] [ssl] [info] google.fr:443 ["tls13"]
[security-txt] [http] [info] https://www.google.fr/.well-known/security.txt ["https://g.co/vulnz","mailto:security@google.com"]
[rdap-whois:registrantEmail] [http] [info] https://rdap.nic.fr/domain/google.fr ["dns-admin@google.com"]
[rdap-whois:registrantPhone] [http] [info] https://rdap.nic.fr/domain/google.fr ["tel:+353.14361000"]
[rdap-whois:registrantAddress] [http] [info] https://rdap.nic.fr/domain/google.fr ["70 Sir John Rogerson's Quay","Dublin","2","IE"]
[rdap-whois:nameServers] [http] [info] https://rdap.nic.fr/domain/google.fr ["ns2.google.com","ns3.google.com","ns4.google.com","ns1.google.com"]
[rdap-whois:expirationDate] [http] [info] https://rdap.nic.fr/domain/google.fr ["2025-12-30T17:16:48Z"]
[rdap-whois:registrantName] [http] [info] https://rdap.nic.fr/domain/google.fr ["Google Ireland Holdings Unlimited Company"]
[rdap-whois:registrantCountry] [http] [info] https://rdap.nic.fr/domain/google.fr ["IE"]
[rdap-whois:secureDNS] [http] [info] https://rdap.nic.fr/domain/google.fr ["false"]
[rdap-whois:status] [http] [info] https://rdap.nic.fr/domain/google.fr ["server update prohibited","server transfer prohibited","server delete prohibited","server recover prohibited"]
[rdap-whois:registrationDate] [http] [info] https://rdap.nic.fr/domain/google.fr ["2000-07-26T22:00:00Z"]
[rdap-whois:lastChangeDate] [http] [info] https://rdap.nic.fr/domain/google.fr ["2024-12-03T11:13:05.110779Z"]
[rdap-whois:registrantOrg] [http] [info] https://rdap.nic.fr/domain/google.fr ["Google Ireland Holdings Unlimited Company"]
[form-detection] [http] [info] https://www.google.com/
[dmarc-detect] [dns] [info] _dmarc.google.fr [""v=DMARC1; p=reject; rua=mailto:mailauth-reports@google.com""]
[caa-fingerprint] [dns] [info] google.fr ["pki.goog"]
[nameserver-fingerprint] [dns] [info] google.fr ["ns1.google.com.","ns3.google.com.","ns4.google.com.","ns2.google.com."]
[spf-record-detect] [dns] [info] google.fr ["v=spf1 -all""]
[txt-fingerprint] [dns] [info] google.fr [""v=spf1 -all""]
[mx-fingerprint] [dns] [info] google.fr ["0 smtp.google.com."]
[ssl-issuer] [ssl] [info] google.fr:443 ["Google Trust Services"]
[ssl-dns-names] [ssl] [info] google.fr:443 ["*.google.fr","google.fr"]
[wildcard-tls] [ssl] [info] google.fr:443 ["CN: *.google.fr","SAN: [*.google.fr google.fr]"]
[INF] Scan completed in 2m. 32 matches found.
```

Figure 17 - Nuclei scan example on https://google.fr

I.4. Linux Package Manager

Package managers are central to Linux system maintenance. They automate the installation, upgrade, configuration, and removal of software. However, as they operate with elevated privileges, they are critical to attack surfaces. This section focuses on the apt package manager used in Debian-based systems and explains how it ensures software integrity and authenticity.

APT Source Files: Format and Parameters

APT relies on repository definitions to locate and authenticate packages. These definitions are stored in source files which describe the repository URI, distribution suite, component sections, architectures, and associated GPG keys.

APT reads its software source configuration from the following locations:

- */etc/apt/sources.list*: Main file listing repository URIs.
- */etc/apt/sources.list.d/*.list*: Supplementary list files for modular configuration.

There are two supported file formats:

- Legacy *.list* format (single-line entries)
- Modern *.sources* format (key–value entries)

Legacy .list format example

```
deb http://deb.debian.org/debian bookworm main contrib non-free
deb-src http://deb.debian.org/debian bookworm main contrib non-free
```


Modern .sources format

These files follow a *key: value* format and allow fine-grained configuration of repository properties.

- Types

Specifies the type of packages to retrieve:

- deb – Binary packages (default for installation)
- deb-src – Source code packages (used for audit or recompilation)

- URIs

Defines the location of the repository:

- Supports http://, https://, ftp://, file://, cdrom:
- Multiple URIs can be listed

- Suites

Indicates the distribution release:

- Named versions: bookworm, jammy, etc.
- Aliases: stable, testing, unstable
- Special: *-security, *-updates, *-backports

- Components

Defines package categories:

- Debian: main, contrib, non-free, non-free-firmware
- Ubuntu: main, universe, restricted, multiverse

- Architectures

Specifies target CPU architectures:

- Common values: amd64, i386, arm64, riscv64

- Signed-By

Specifies the GPG key used to verify the repository signature:

  - Replaces global apt-key
  - Increases repository isolation and trust control

Avoid legacy apt-key. Always use Signed-By for key scoping.

Modern .source format example

```
# /etc/apt/sources.list.d/ubuntu.sources

Types: deb
URIs: http://archive.ubuntu.com/ubuntu
Suites: noble noble-updates noble-backports
Components: main restricted universe multiverse
Signed-By: /usr/share/keyrings/ubuntu-archive-keyring.gpg

Types: deb
URIs: http://security.ubuntu.com/ubuntu
Suites: noble-security
Components: main restricted universe multiverse
Signed-By: /usr/share/keyrings/ubuntu-archive-keyring.gpg
```

Integrity and Authenticity of Linux Software Packages

Ensuring the integrity and authenticity of packages is a core security feature of modern Linux distributions. APT uses digital signatures based on PGP (see: Erreur ! Source du renvoi introuvable.) to verify that packages have not been tampered with and originate from trusted sources.

Principle of Package Signing

Each package or repository is signed by its maintainer using a PGP private key. The verification process works as follows:

- Signing:
  - The package maintainer generates a digital signature using a private key.
  - This signature is based on a hash of the package content.
- Verification:
  - The user's system stores the corresponding public key.
  - When a package is downloaded, APT verifies the signature using this key.

If the computed hash matches the one from the signature, the package is considered valid.

This process ensures:

- The package has not been altered (integrity)
- The package comes from a trusted source (authenticity)

Key Locations and Trust

- */etc/apt/trusted.gpg.d/*: Legacy location for trusted public keys (deprecated)
- */usr/share/keyrings/* or */etc/apt/keyrings/*: Recommended for scoped GPG keys used with Signed-By

The use of /etc/apt/trusted.gpg.d/ is discouraged due to security risks associated with its automatic trust of any added keys, potentially leading to the installation of malicious software. Instead, /usr/share/keyrings/ or /etc/apt/keyrings/ are recommended as they allow for better key management and explicit trust associations with specific repositories, enhancing security and maintainability.

Key Management and *signed-by*

Important: *The legacy tool apt-key is deprecated.*

Instead of trusting all keys system-wide, modern best practice is to use the *signed-by* directive in *.sources* files.

This binds a specific repository to a specific GPG key, preventing cross-repo trust issues.

- Example:

```
Signed-By: /usr/share/keyrings/ubuntu-archive-keyring.gpg
```

Security Implications

If signature verification fails:

- The package will not be installed
- APT will return an error and abort the operation

This prevents:

- Man-in-the-middle attacks
- Malicious repository injection
- Compromised package deployment

APT Tools: apt, apt-cache, and apt-file

APT (Advanced Package Tool) provides several command-line tools to interact with Debian-based package systems. Each has a specific role in managing, querying, and exploring packages.

apt: The High-Level Package Manager

Introduced as a user-friendly frontend, apt combines functionalities from apt-get and apt-cache.

- Main uses:
  - Installing, updating, and removing packages
  - Refreshing the package index
  - Displaying useful information

```
apt update                    # Refresh package lists
apt upgrade                   # Upgrade all installed packages
apt full-upgrade              # Upgrade, including packages requiring dependency changes
apt install nmap              # Install a package
apt remove apache2            # Uninstall a package (keep config files)
apt purge apache2             # Uninstall and remove configuration files
apt list --installed          # List installed packages
apt show openssh-server       # Show package details
```

apt-cache: Package Metadata Lookup

apt-cache allows detailed exploration of the local package index without modifying the system. It is useful for searching and inspecting metadata:

- Dependencies
- Reverse dependencies
- Package versions and descriptions

```
apt-cache search firewall          # Search for packages by keyword
apt-cache show ufw                 # Show metadata for a specific package
apt-cache depends nginx            # List package dependencies
apt-cache rdepends python3         # List reverse dependencies
```


apt-file: Search Contents of Packages

apt-file allows you to search for files contained within packages, even if those packages are not installed. It is mainly used for finding which package provides a specific command or configuration file. It is useful to verify that a file won't be deleted or modified during an update.

```
apt-file update          # Update its internal database
apt-file search zlib.so  # Find the package providing this binary
apt-file list nginx      # List all files provided by nginx
```


I.5. Centralized Patch & Endpoint Management

In a professional environment, centralized patch and endpoint management systems are essential for maintaining system integrity, security compliance, and operational continuity across all devices. While these two responsibilities, patch management and endpoint management, can be handled by separate tools, some platforms combine both capabilities into a unified solution. These tools allow administrators to remotely manage software updates, deploy configurations, and monitor endpoints from a unified console.

ManageEngine Endpoint Central

ManageEngine Endpoint Central is a unified endpoint management (UEM) solution developed by Zoho Corporation. It provides a centralized platform for managing, securing, and auditing workstations and servers, whether Linux, Windows, or macOS, from a single web interface.

Although originally focused on Windows fleets, Endpoint Central has evolved to support Linux systems, offering key functionalities for patching, asset tracking, and remote administration.

Documentation: https://www.manageengine.com/fr/desktop-central/

Key Features for Linux Systems

- Patch Management
  - Automatically scans for missing patches and applies OS-level updates using native tools (apt, yum, dnf, etc.).
  - Schedule patch deployment by group or device
  - Define test and production deployment rings
  - Track patch success/failure from a dashboard
- Software Inventory and License Tracking
  - Maintains a detailed software inventory of installed packages and versions, useful for compliance audits and license management.
- Remote Control and Shell Access
  - Offers secure remote terminal sessions to Linux servers and desktops for troubleshooting or configuration tasks.
- Custom Script Deployment
  - Supports pushing and executing shell scripts on selected devices or groups, making it ideal for configuration management tasks.

- Reporting and Compliance Dashboards

- Generate reports for:
  - Systems missing critical updates
  - Patch deployment status
  - System health and compliance metrics

Architecture Overview

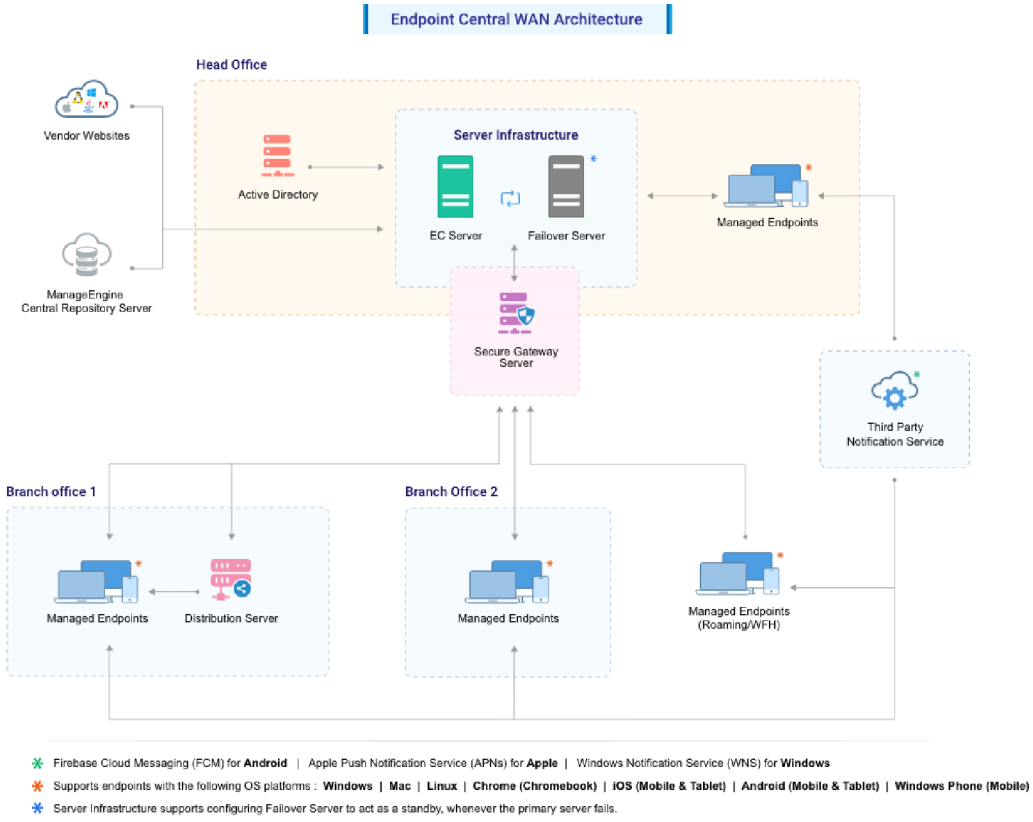

Figure 18 - Endpoint Centrale architecture overview

Source : https://www.manageengine.com/products/desktop-central/images/endpoint-central-wan-architecture.png

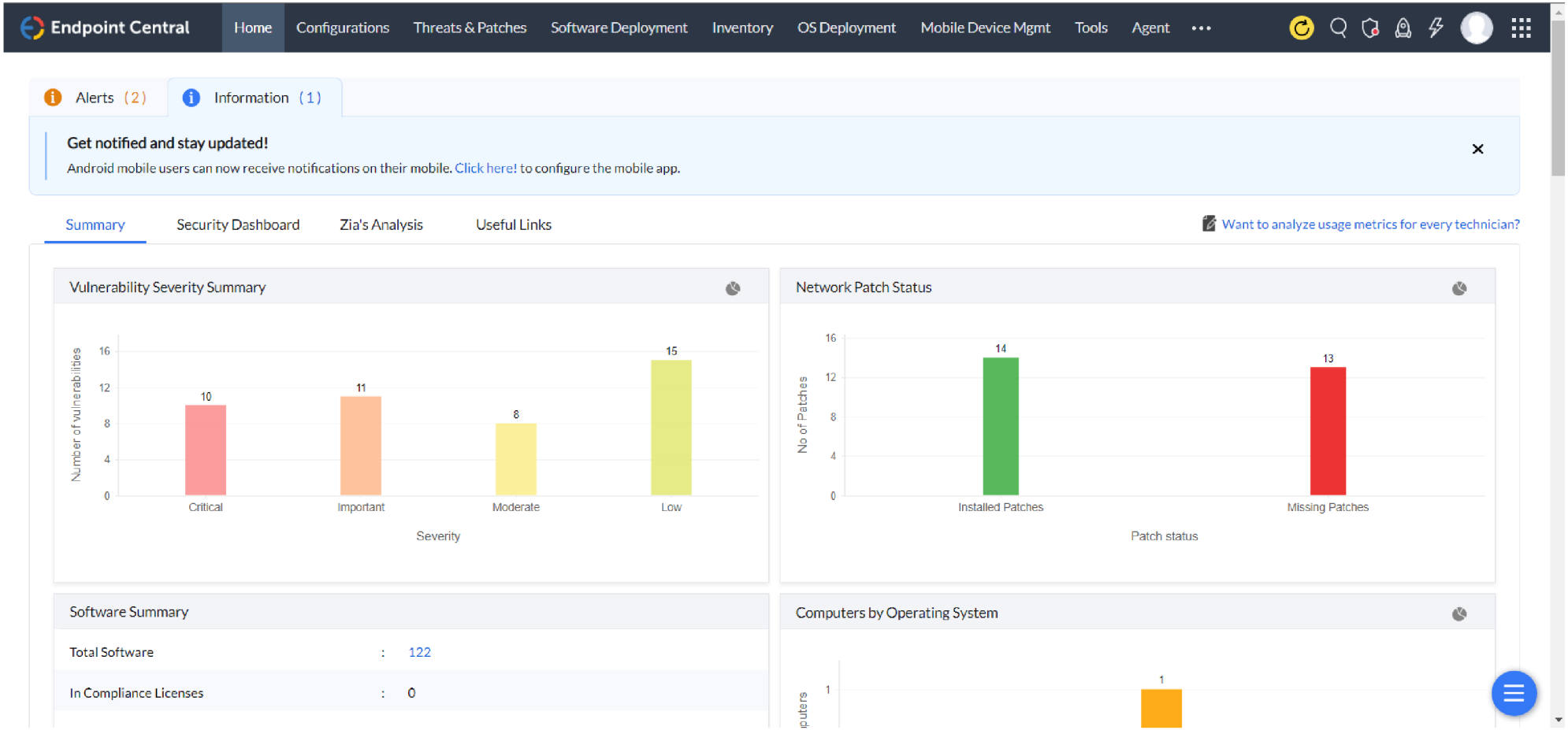

Figure 19 - Endpoint Central interface

Source: https://www.manageengine.com/products/desktop-central/help/images/ec-console-3.png

Landscape: Ubuntu Systems Management at Scale

Landscape is Canonical's centralized management tool designed to administer, monitor, and automate tasks across large fleets of Ubuntu machines. It is particularly suited for environments that require centralized package management, compliance, and reporting, whether on-premises, in the cloud, or in hybrid infrastructures.

Documentation: https://ubuntu.com/landscape

Key Features

- Patch & Package Management
  - Landscape lets administrators view, schedule, and apply updates and security patches across all registered Ubuntu systems.
  - Approve or reject updates manually or automatically
  - Group systems by role or environment (e.g., dev, prod)
  - Schedule updates outside business hours
- System Inventory and Monitoring
  - Hardware specs and software inventory
  - Available updates and upgrade history
  - System performance (CPU, memory, disk usage)
- User and Role Management
  - Supports role-based access control (RBAC) for delegating management responsibilities across teams.
- Script Automation
  - Enables batch execution of custom shell scripts across selected machines, useful for enforcing configuration consistency.
- Compliance & Reporting
  - Landscape allows you to generate reports on package status, update history, and compliance posture for audits.

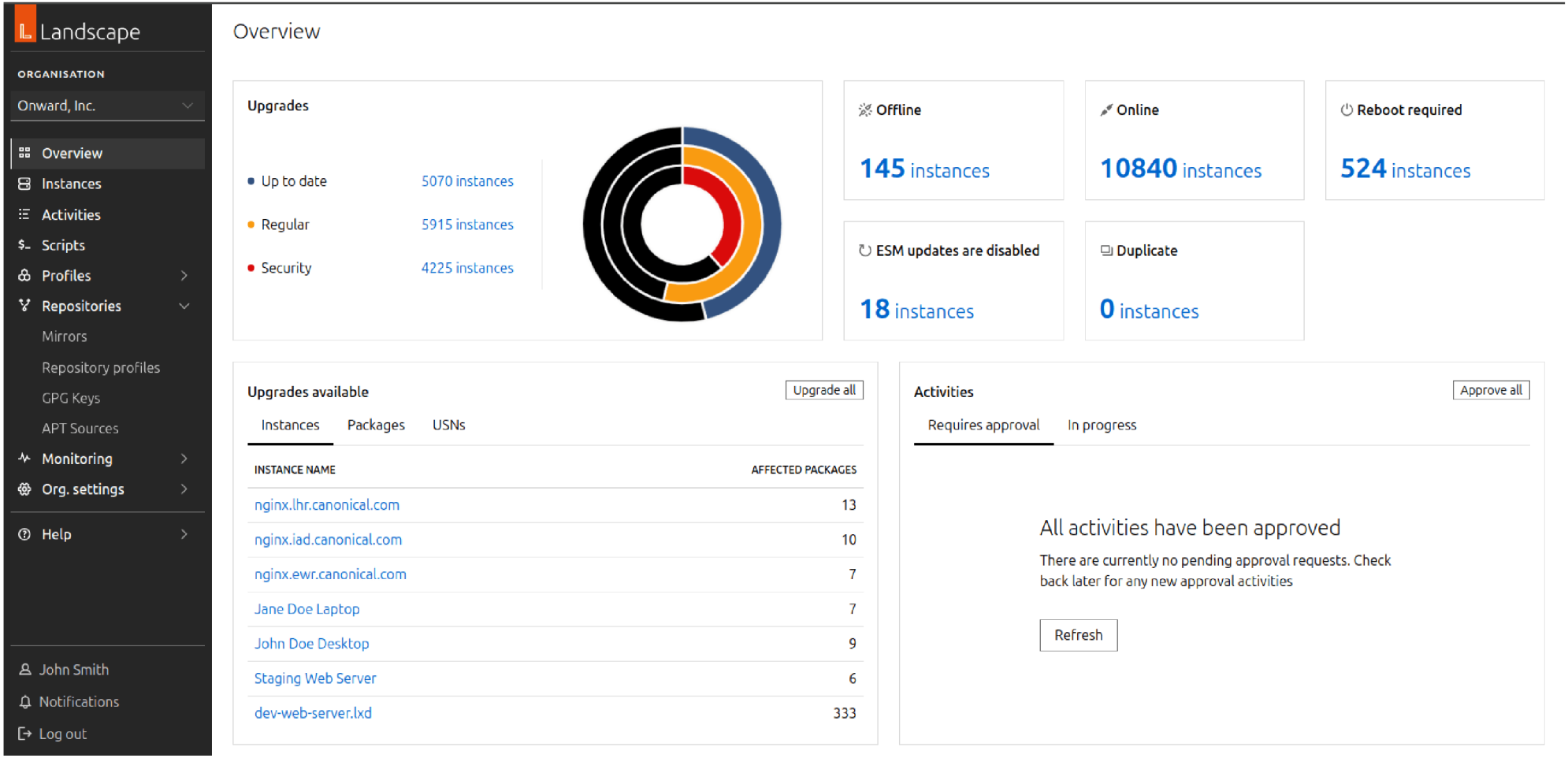

Figure 20 - Landscape interface

Source: https://canonical.com/blog/canonical-releases-landscape-24-04-lts

II. Further Resources and Recommended Readings

Web site

- https://wiki.ubuntu.com/Security/Features
- https://cyber.gouv.fr/publications/recommandations-de-securite-relatives-un-systeme-gnulinux
- https://blog.stephane-robert.info/
- https://downloads.cisecurity.org/
- https://www.it-connect.fr/cours-tutoriels/administration-systemes/linux/
- https://github.com/imthenachoman/How-To-Secure-A-Linux-Server
- https://www.linuxfoundation.org/lf-security

Playground and labs

- https://webvm.io/
- https://labex.io/projects/category/linux

Books

Ward, B. (2021). How Linux Works, 3rd Edition: What Every Superuser Should Know. San Francisco, CA: No Starch Press.

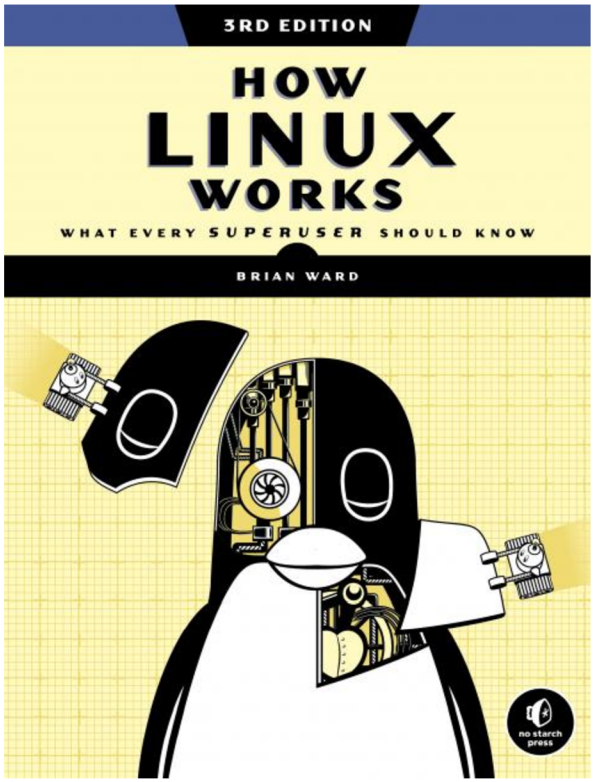

Figure 21 - How Linux Works, 3rd Edition: What Every Superuser Should Know

Source : https://nostarch.com/howlinuxworks3