Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Vulnerability Assessment Area Territoriale di Ricerca Napoli 1

Angelo Esposito[2], Paolo Vanacore[2], Giuseppe Trerotola[1]

[1]Istituto di Calcolo e Reti ad Alte Prestazioni del Consiglio Nazionale delle Ricerche (ICAR-CNR)

[2]Area Territoriale di Ricerca Napoli 1 (ATdR NA1)

angelo.esposito@cnr.it
paolo.vanacore@cnr.it
giuseppe.trerotola@icar.cnr.it

RT-ICAR-NA-2026-06 **Maggio 2026**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sede di Cosenza, Via P. Bucci Cubo 8/9C, 87036 Rende, Italy, URL: www.icar.cnr.it – Sede di Napoli, Via P. Castellino 111, 80131 Napoli, URL: www.na.icar.cnr.it – Sede di Palermo, Via Ugo La Malfa 153, 90146 Palermo, URL: www.pa.icar.cnr.it

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Vulnerability Assessment Area Territoriale di Ricerca Napoli 1

Angelo Esposito[2], Paolo Vanacore[2], Giuseppe Trerotola[1]

[1]Istituto di Calcolo e Reti ad Alte Prestazioni del Consiglio Nazionale delle Ricerche (ICAR-CNR)

[2]Area Territoriale di Ricerca Napoli 1 (ATdR NA1)

angelo.esposito@cnr.it
paolo.vanacore@cnr.it
giuseppe.trerotola@icar.cnr.it

RT-ICAR-NA-2026-06 **Maggio 2026**

Vulnerability Assessment

Area Territoriale di Ricerca Napoli 1

Abstract

Il presente rapporto tecnico descrive l'analisi, la progettazione e l'implementazione di un sistema di Vulnerability Assessment (VA) all'interno dell'infrastruttura di rete dell'Area Territoriale di Ricerca Napoli 1 (ATdR NA1) del Consiglio Nazionale delle Ricerche (CNR). L'introduzione di una piattaforma di scansione continua risponde alla duplice necessità di gestire in modo proattivo una superficie di attacco intrinsecamente dinamica e di adempiere agli stringenti obblighi di conformità normativa previsti dalla Direttiva NIS2 (UE 2022/2555), che classifica il CNR come Soggetto Essenziale. A seguito di un'analisi comparativa dei principali strumenti di cybersecurity, sia commerciali sia open-source, è stata selezionata la suite Greenbone Vulnerability Management (GVM) / OpenVAS Community Edition, in virtù della sua maturità tecnologica, delle avanzate capacità di reportistica e della garanzia di mantenere la telemetria e i dati sensibili all'interno del perimetro locale (on-premise). Il documento fornisce una guida metodologica ed esecutiva che spazia dall'installazione dell'architettura tramite container Docker su ambiente Ubuntu 24.04 LTS, fino alla disamina dei concetti chiave della piattaforma (Feed, Target, Scan Config, Credenziali e Task). Viene inoltre illustrato l'utilizzo operativo del sistema per il Risk Management attraverso la gestione di report differenziali, inventario degli asset, override e sistemi di alerting automatizzati.

Keywords: Vulnerability Assessment, OpenVAS, Greenbone Vulnerability Management (GVM), Direttiva NIS2, Cybersecurity, Network Security, Risk Management, Compliance IT, Infrastruttura di Rete, Monitoraggio di rete.

Sommario

1. Introduzione

Il presente documento descrive l'analisi, la progettazione e l'implementazione di un sistema di Vulnerability Assessment (VA) all'interno dell'infrastruttura di rete dell'Area Territoriale di Ricerca Napoli 1 (ATdR NA1) del CNR.

L'ATdR NA1 è un raggruppamento di nove Istituti di ricerca e due uffici del Consiglio Nazionale delle Ricerche (CNR), situato nel centro di Napoli, in via Pietro Castellino, nel distretto Vomero-Arenella. L'ATdR NA1 si estende su una superficie di circa 12.000 m^2, con 20.000 m^2 di laboratori e strutture e 5.000 m^2 di terreno, per una superficie totale di circa 25.000 m^2.

Il VA è un processo sistematico e proattivo finalizzato all'identificazione delle vulnerabilità di sicurezza all'interno di un ambiente informatico (sistemi operativi, applicazioni, apparati di rete). A differenza di soluzioni difensive perimetrali (come i Firewall) o reattive (come i sistemi Antivirus, che intervengono a minaccia in corso), l'obiettivo primario dei VA non è mitigare la minaccia, ma fornire una visione dettagliata delle falle di sicurezza, permettendo di pianificare interventi prima che queste possano essere sfruttate da soggetti malevoli.

L'introduzione di una piattaforma di scansione continua delle vulnerabilità risponde a due esigenze fondamentali:

- corretta gestione infrastrutturale;
- conformità normativa alla direttiva NIS2.

Da un punto di vista tecnico e operativo, la rete dell'ATdR NA1 presenta caratteristiche di dinamicità – aggiornamenti, configurazioni e introduzione di nuovi asset – che comportano una dinamicità anche della sua "superficie di attacco". L'adozione di un sistema di VA regolare permette di avere la verifica degli aggiornamenti, la rilevazione di porte esposte accidentalmente, servizi obsoleti, certificati scaduti o credenziali di default attive e, quindi, consente di avere una riduzione dell'esposizione al rischio, con un approccio proattivo.

Dal punto di vista normativo, l'attività mira a contribuire, per l'ATdR NA1, al soddisfacimento dei requisiti introdotti dalla Direttiva NIS2 (UE 2022/2555). In tale contesto, il CNR è stato individuato quale Soggetto Essenziale, comportando obblighi stringenti, sia in termini di resilienza che di gestione del rischio. L'utilizzo di un sistema di VA fornisce, tramite report tracciabili, adeguati strumenti per la valutazione e il monitoraggio della propria infrastruttura.

2. Strumenti software per il Vulnerability Assessment

Nell'ambito delle attività di VA, esiste un'ampia gamma di strumenti, dai tool a riga di comando a piattaforme enterprise complesse.

Tra gli strumenti a riga di comando, è molto diffuso Nmap [1], tool gratuito e open-source (licenza GNU GPLv2). Nmap è spesso utilizzato per la mappatura di rete (Network Discovery) e il port scanning. Nel contesto in esame, è rilevante la funzionalità "Nmap Scripting Engine" (NSE) [7], che consente la definizione di script, in linguaggio Lua, per l'automazione di compiti di network discovery, version detection e vulnerability detection (Figura 1). Nmap, tuttavia, non offre una gestione integrata dei report, un database di vulnerabilità e funzionalità di storicizzazione.

```
paoloaq@Air-di-paolo nmap % nmap -sV --script=vuln -oA report_www_area_na_cnr_it www.area.na.cnr.it
Starting Nmap 7.97SVN ( https://nmap.org ) at 2026-03-17 18:28 +0100
Nmap scan report for www.area.na.cnr.it (140.164.7.19)
Host is up (0.028s latency).
Not shown: 994 closed tcp ports (conn-refused)
PORT    STATE    SERVICE      VERSION
22/tcp  open     ssh          OpenSSH 9.6p1 Ubuntu 3ubuntu13.15 (Ubuntu Linux; protocol 2.0)
| vulners:
|   cpe:/a:openbsd:openssh:9.6p1:
|       PACKETSTORM:179290      10.0    https://vulners.com/packetstorm/PACKETSTORM:179290      *EXPLOIT*
|       1EEC8894-D2F7-547C-827C-915BE866875C    10.0    https://vulners.com/githubexploit/1EEC8894-D2F7-547C-827C-915BE866875C  *EXPLOIT*
|       33D623F7-98E0-5F75-80FA-81AA666D1340    9.8     https://vulners.com/githubexploit/33D623F7-98E0-5F75-80FA-81AA666D1340  *EXPLOIT*
|       F8981437-1287-5B69-93F1-657DFB1DCE59    9.3     https://vulners.com/githubexploit/F8981437-1287-5B69-93F1-657DFB1DCE59  *EXPLOIT*
|       CB2926E1-2355-5C82-A42A-D4F72F114F9B    9.3     https://vulners.com/githubexploit/CB2926E1-2355-5C82-A42A-D4F72F114F9B  *EXPLOIT*
|       8DEE261C-33D4-5057-BA46-E4293B705BAE    9.3     https://vulners.com/githubexploit/8DEE261C-33D4-5057-BA46-E4293B705BAE  *EXPLOIT*
|       6FD8F914-B663-533D-8866-23313FD37804    9.3     https://vulners.com/githubexploit/6FD8F914-B663-533D-8866-23313FD37804  *EXPLOIT*
|       PACKETSTORM:190587      8.1     https://vulners.com/packetstorm/PACKETSTORM:190587      *EXPLOIT*
|       FB2E9ED1-43D7-585C-A197-0D6628B20134    8.1     https://vulners.com/githubexploit/FB2E9ED1-43D7-585C-A197-0D6628B20134  *EXPLOIT*
|       FA3992CE-9C4C-5350-8134-177126E0BD3F    8.1     https://vulners.com/githubexploit/FA3992CE-9C4C-5350-8134-177126E0BD3F  *EXPLOIT*
|       EFD615F0-8F17-5471-AA83-0F491FD497AF    8.1     https://vulners.com/githubexploit/EFD615F0-8F17-5471-AA83-0F491FD497AF  *EXPLOIT*
|       EC20B9C2-6857-5848-848A-A9F430D13EEB    8.1     https://vulners.com/githubexploit/EC20B9C2-6857-5848-848A-A9F430D13EEB  *EXPLOIT*
|       EB13CBD6-BC93-5F14-A210-AC0B5A1D8572    8.1     https://vulners.com/githubexploit/EB13CBD6-BC93-5F14-A210-AC0B5A1D8572  *EXPLOIT*
|       E543E274-C20A-582A-8F8E-F8E3F381C345    8.1     https://vulners.com/githubexploit/E543E274-C20A-582A-8F8E-F8E3F381C345  *EXPLOIT*
|       E34FCCEC-226E-5A46-9B1C-BCD6EF7D3257    8.1     https://vulners.com/githubexploit/E34FCCEC-226E-5A46-9B1C-BCD6EF7D3257  *EXPLOIT*
|       E24EEC0A-40F7-5BBC-9E4D-7B13522FF915    8.1     https://vulners.com/githubexploit/E24EEC0A-40F7-5BBC-9E4D-7B13522FF915  *EXPLOIT*
|       DC1BB99A-8B57-5EE5-9AC4-3D9D59BFC346    8.1     https://vulners.com/githubexploit/DC1BB99A-8B57-5EE5-9AC4-3D9D59BFC346  *EXPLOIT*
|       DA18D761-BB81-54B6-85CB-CFD73CE33621    8.1     https://vulners.com/githubexploit/DA18D761-BB81-54B6-85CB-CFD73CE33621  *EXPLOIT*
|       D8974199-6B08-5895-9610-919F71468F23    8.1     https://vulners.com/githubexploit/D8974199-6B08-5895-9610-919F71468F23  *EXPLOIT*
|       D52370EF-02EE-507D-9212-2D8EA86CBA94    8.1     https://vulners.com/githubexploit/D52370EF-02EE-507D-9212-2D8EA86CBA94  *EXPLOIT*
|       CVE-2024-6387   8.1     https://vulners.com/cve/CVE-2024-6387
|       CFEBF7AF-651A-5302-80B8-F8146D5B33A6    8.1     https://vulners.com/githubexploit/CFEBF7AF-651A-5302-80B8-F8146D5B33A6  *EXPLOIT*
|       C6FB6D50-F71D-5870-B671-D6A09A95627F    8.1     https://vulners.com/githubexploit/C6FB6D50-F71D-5870-B671-D6A09A95627F  *EXPLOIT*
|       C623D558-C162-5D17-88A5-4799A2BEC001    8.1     https://vulners.com/githubexploit/C623D558-C162-5D17-88A5-4799A2BEC001  *EXPLOIT*
|       C5B2D4A1-8C3B-5FF7-B620-EDE207B027A0    8.1     https://vulners.com/githubexploit/C5B2D4A1-8C3B-5FF7-B620-EDE207B027A0  *EXPLOIT*
|       C185263E-3E67-5550-B9C0-AB9C15351960    8.1     https://vulners.com/githubexploit/C185263E-3E67-5550-B9C0-AB9C15351960  *EXPLOIT*
|       BDA609DA-6936-50DC-A325-19FE2CC68562    8.1     https://vulners.com/githubexploit/BDA609DA-6936-50DC-A325-19FE2CC68562  *EXPLOIT*
|       BA3887BD-F579-53B1-A4A4-FF49E953E1C0    8.1     https://vulners.com/githubexploit/BA3887BD-F579-53B1-A4A4-FF49E953E1C0  *EXPLOIT*
|       B1F444E0-F217-5FC0-B266-EBD48589940F    8.1     https://vulners.com/githubexploit/B1F444E0-F217-5FC0-B266-EBD48589940F  *EXPLOIT*
|       92254168-3B26-54C9-B9BE-B4B7563586B5    8.1     https://vulners.com/githubexploit/92254168-3B26-54C9-B9BE-B4B7563586B5  *EXPLOIT*
|       91752937-D1C1-5913-A96F-72F8B8AB4280    8.1     https://vulners.com/githubexploit/91752937-D1C1-5913-A96F-72F8B8AB4280  *EXPLOIT*
|       90104C60-A887-5437-8521-545277685F55    8.1     https://vulners.com/githubexploit/90104C60-A887-5437-8521-545277685F55  *EXPLOIT*
|       89F96BAB-1624-51B5-B09E-E771D918D1E6    8.1     https://vulners.com/githubexploit/89F96BAB-1624-51B5-B09E-E771D918D1E6  *EXPLOIT*
|       81F0C05A-8650-5DE8-97E9-0D89F1807E5D    8.1     https://vulners.com/githubexploit/81F0C05A-8650-5DE8-97E9-0D89F1807E5D  *EXPLOIT*
|       7C7167AF-E780-5506-BEFA-02E5362E8E48    8.1     https://vulners.com/githubexploit/7C7167AF-E780-5506-BEFA-02E5362E8E48  *EXPLOIT*
|       79FE1ED7-EB3D-5978-A12E-AAB1FFECCCAC    8.1     https://vulners.com/githubexploit/79FE1ED7-EB3D-5978-A12E-AAB1FFECCCAC  *EXPLOIT*
|       795762E3-BAB4-54C6-B677-83B0ACC2B163    8.1     https://vulners.com/githubexploit/795762E3-BAB4-54C6-B677-83B0ACC2B163  *EXPLOIT*
|       774022BB-71DA-57C4-9B8F-E21D667DE4BC    8.1     https://vulners.com/githubexploit/774022BB-71DA-57C4-9B8F-E21D667DE4BC  *EXPLOIT*
|       743E5025-3BB8-5EC4-AC44-2AA679730661    8.1     https://vulners.com/githubexploit/743E5025-3BB8-5EC4-AC44-2AA679730661  *EXPLOIT*
|       73A19EF9-346D-5B2B-9792-05D9FE3414E2    8.1     https://vulners.com/githubexploit/73A19EF9-346D-5B2B-9792-05D9FE3414E2  *EXPLOIT*
|       6E81EAE5-2156-5ACB-9046-D792C7FAF698    8.1     https://vulners.com/githubexploit/6E81EAE5-2156-5ACB-9046-D792C7FAF698  *EXPLOIT*
|       6B78D204-22B0-5D11-8A0C-6313958B473F    8.1     https://vulners.com/githubexploit/6B78D204-22B0-5D11-8A0C-6313958B473F  *EXPLOIT*
|       65650BAD-813A-565D-953D-2E7932B26094    8.1     https://vulners.com/githubexploit/65650BAD-813A-565D-953D-2E7932B26094  *EXPLOIT*
|       649197A2-0224-5B5C-9C4E-B5791D42A9FB    8.1     https://vulners.com/githubexploit/649197A2-0224-5B5C-9C4E-B5791D42A9FB  *EXPLOIT*
|       61DDEEE4-2146-5E84-9804-B780AA73E33C    8.1     https://vulners.com/githubexploit/61DDEEE4-2146-5E84-9804-B780AA73E33C  *EXPLOIT*
|       608FA50C-AEA1-5A83-8297-A15FC7D32A7C    8.1     https://vulners.com/githubexploit/608FA50C-AEA1-5A83-8297-A15FC7D32A7C  *EXPLOIT*
|       5D2CB1F8-DC04-5545-8BC7-29EE3DA8890E    8.1     https://vulners.com/githubexploit/5D2CB1F8-DC04-5545-8BC7-29EE3DA8890E  *EXPLOIT*
|       5C81C5C1-22D4-55B3-B843-5A9A60AAB6FD    8.1     https://vulners.com/githubexploit/5C81C5C1-22D4-55B3-B843-5A9A60AAB6FD  *EXPLOIT*
|       53BCD84F-BD22-5C9D-95B6-4B83627AB37F    8.1     https://vulners.com/githubexploit/53BCD84F-BD22-5C9D-95B6-4B83627AB37F  *EXPLOIT*
|       4FB01B00-F993-5CAF-BD57-D7E290D10C1F    8.1     https://vulners.com/githubexploit/4FB01B00-F993-5CAF-BD57-D7E290D10C1F  *EXPLOIT*
|       48603E8E-B170-57FF-85B9-67A7D9504891    8.1     https://vulners.com/githubexploit/48603E8E-B170-57FF-85B9-67A7D9504891  *EXPLOIT*
```

Figura 1 esempio di uso di Nmap Scripting Engine per il rilevamento di vulnerabilità

Tra gli scanner basati su template, una combinazione comune, è Naabu [3] + Nuclei [4]. Entrambi sono strumenti gratuiti, rilasciati con licenze open-source (licenza MIT). Mentre Naabu è utilizzato per il port scanning (Figura 2), Nuclei è impiegato per inviare richieste mirate ad un host, basate su template YAML, al fine di identificare vulnerabilità specifiche (Figura 3). Questa è una soluzione estremamente veloce e personalizzabile, ma orientata più al penetration testing che non alla gestione sistematica, schedulata e reportistica di un'intera rete complessa.

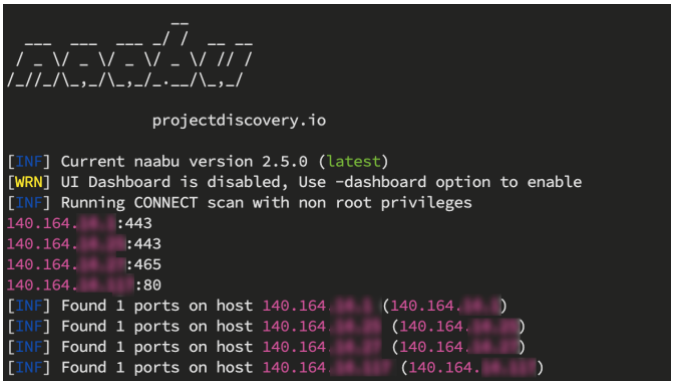

Figura 2 esempio di scansione con Naabu

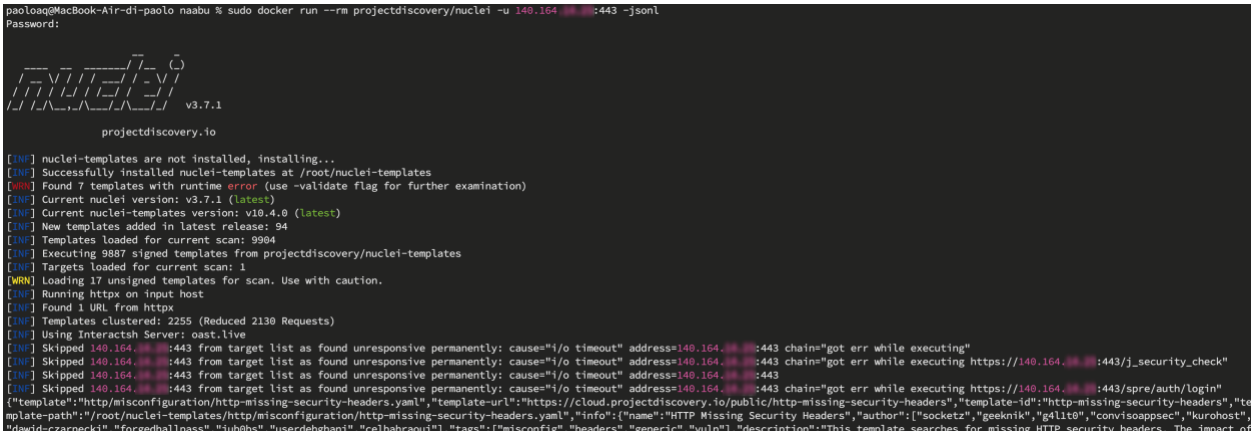

Figura 3 esempio di scansione con Nuclei verso uno specifico host

Come mostrato in Figura 3, l'output del test di vulnerabilità di Nuclei è in formato JSONL. Per una più facile consultazione dell'output, è possibile utilizzare strumenti dedicati, come, ad esempio, Nuclei UI [5] (Figura 4).

A differenza degli strumenti appena descritti, Greenbone Vulnerability Management (GVM)/OpenVAS Community Edition [8] rappresenta una soluzione open-source[1], matura e completa, per il VA. GVM/OpenVAS unisce la profondità di analisi delle suite commerciali (community feed delle vulnerabilità costantemente aggiornati, scansioni multi-subnet, scansioni

[1] OpenVAS Community Feed rilasciato con licenza Open Data Commons Open Database License version 1.0 (ODbLv1) - Vulnerability Tests (VTs) e OpenVAS scanner rilasciati con licenza GNU General Public License Version 2 (GNU GPLv2)

autenticate e non autenticate, pianificazione avanzata tramite Web GUI, ticketing, ...) all'assenza di costi di licenza. Inoltre, l'architettura on-premise garantisce che i dati sensibili sulle eventuali vulnerabilità rilevate risiedano nel perimetro locale.

Tra le soluzioni commerciali di maggior successo, vi sono Tenable Nessus, Qualys Cloud Platform e Rapid7 InsightVM. Se da un lato tali soluzioni offrono interfacce estremamente raffinate, analisi basate su agenti e integrazioni con i sistemi di ticketing, dall'altro presentano costi di licenza significativi (spesso basati sul numero di IP scansionati) e, in alcuni casi, come Qualys, richiedono l'invio dei dati telemetrici dell'infrastruttura verso il cloud del vendor (modello SaaS).

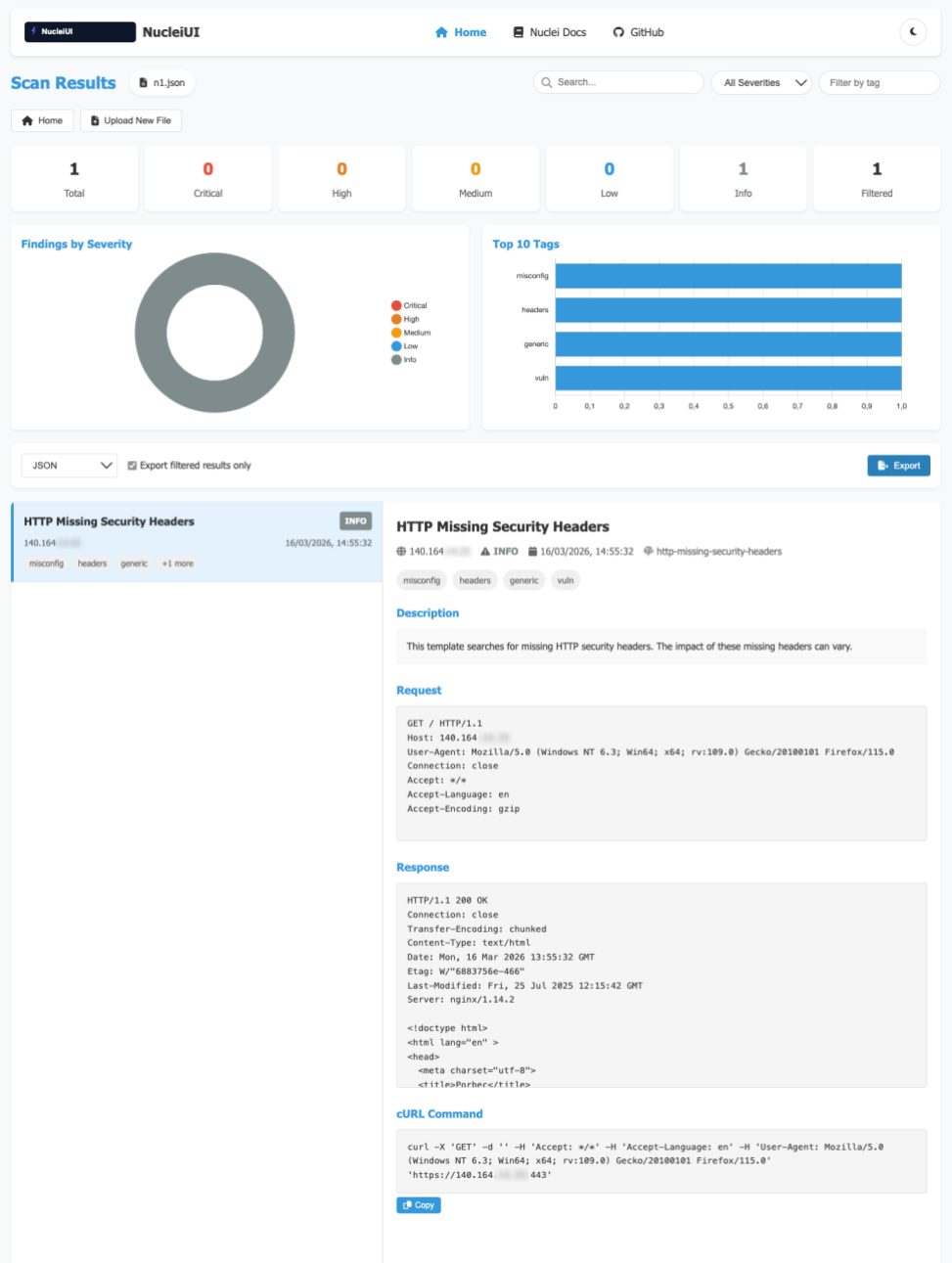

Figura 4 interfaccia web di Nuclei UI

Gli strumenti di VA descritti, come OpenVAS, operano prevalentemente a livello infrastrutturale e di servizio. Per un'analisi approfondita delle applicazioni web, è utile affiancare al VA strumenti di test mirati. Un esempio, gratuito e open-source (licenza Apache 2.0), è OWASP ZAP (Zed Attack Proxy) [9] (Figura 5), che consente di scansionare attivamente il traffico HTTP/HTTPS, effettuando lo spidering (navigazione automatizzata) delle pagine web e iniettando payload mirati per individuare vulnerabilità applicative specifiche (come SQL Injection, Cross-Site Scripting, ecc), in conformità con la top 10 dell'OWASP [10].

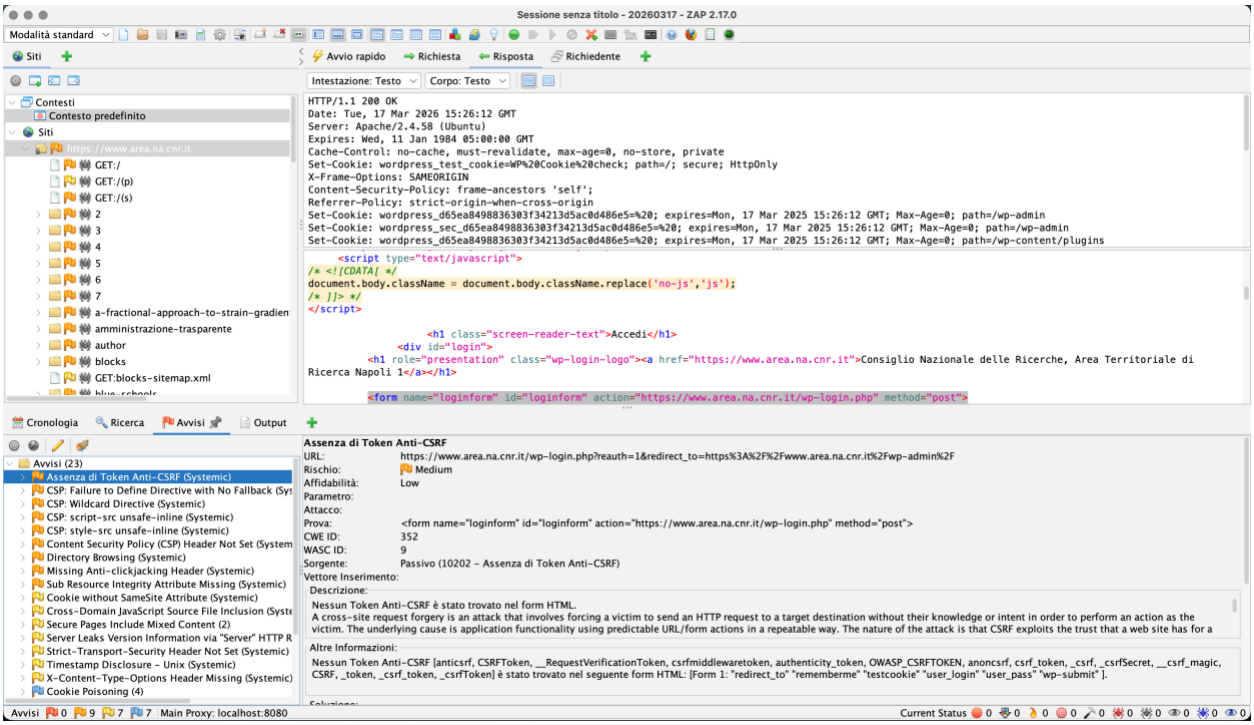

Figura 5 GUI di OWASP ZAP

3. Installazione di Greenbone/OpenVAS

In questo capitolo sono riportati i passi per l'installazione di Docker, Docker Compose e Greenbone/OpenVAS [11] in ambiente Ubuntu 24.04.4 LTS.

- *Installazione di Docker e Docker Compose dal repository ufficiale:*

1. Aggiornamento della lista dei pacchetti

```
sudo apt update
```

2. Installazione dei prerequisiti

```
sudo apt install ca-certificates curl -y
```

3. Creazione della cartella per le chiavi di sicurezza e download della chiave GPG ufficiale di Docker

```
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

4. Aggiunta del repository Docker

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt update
```

5. Installazione di Docker e Docker Compose

```
sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y
```

6. Aggiunta dell'utente corrente al gruppo Docker – per evitare l'uso del comando `sudo` per controllare Docker

```
sudo usermod -aG docker $USER
```

NOTA: verifica dell'installazione di Docker e Docker Compose

```
docker --version
docker compose version
```

- *Installazione di OpenVAS/Greenbone:*

Greenbone rende disponibile uno script bash per seguire l'intero processo di installazione, tramite docker-compose, e l'avvio dei containers. È possibile scaricare ed eseguire lo script, tramite i comandi

```
curl -f -O https://greenbone.github.io/docs/latest/_static/setup-and-start-greenbone-community-
edition.sh && chmod u+x setup-and-start-greenbone-community-edition.sh

./setup-and-start-greenbone-community-edition.sh
```

In alternativa:

1. Creazione directory per i file di configurazione di Greenbone:

```
export DOWNLOAD_DIR=$HOME/greenbone
mkdir -p $DOWNLOAD_DIR
cd $DOWNLOAD_DIR
```

2. Download del file docker-compose:

```
curl -f -O -L https://greenbone.github.io/docs/latest/_static/compose.yaml --output-dir
"$DOWNLOAD_DIR"
```

3. Download delle immagini e avvio dei containers:

```
sudo docker compose -f $DOWNLOAD_DIR/compose.yaml pull
sudo docker compose -f $DOWNLOAD_DIR/compose.yaml up -d
```

È possibile visualizzare i messaggi di log dei container in esecuzione, usando il comando:

```
docker compose -f $DOWNLOAD_DIR/ compose.yaml logs -f
```

4. Per impostazione predefinita, viene creato un utente "admin" con la password "admin". Questa configurazione non è sicura ed è fortemente consigliato di impostare una nuova password. Per aggiornare l'utente "admin" con una password a propria scelta, anziché utilizzare quella generata automaticamente, è possibile utilizzare il seguente comando:

```
docker compose -f $DOWNLOAD_DIR/compose.yml \
    exec -u gvmd gvmd gvmd --user=admin --new-password='<password>'
```

Avviati i container è possibile accedere all'interfaccia web all'indirizzo: https://127.0.0.1 (Figura 6 e Figura 7).

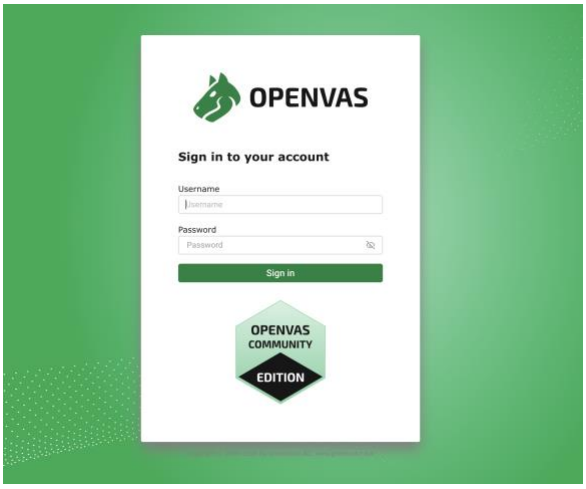

Figura 6 interfaccia web di login di Greenbone/OpenVAS

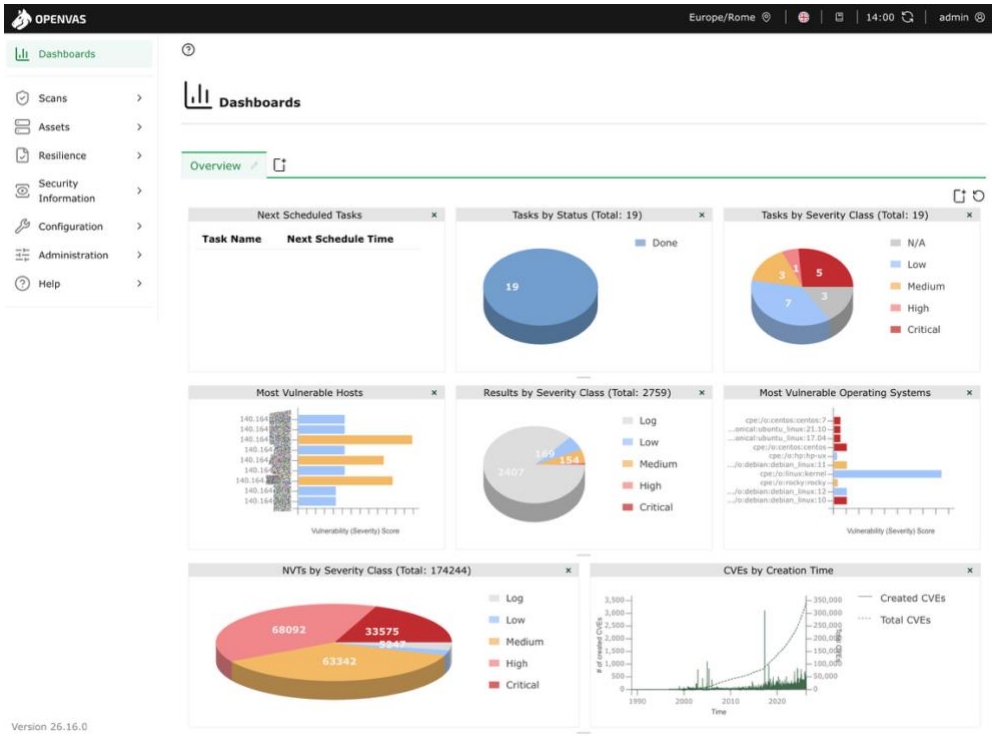

Figura 7 dashboard di Greenbone/OpenVAS

NOTA: nel `docker-compose.yml`, alla versione in cui è redatto il presente documento (22.4), i servizi dello stack adottano una logica di riavvio che non garantisce il corretto ripristino di tutte le funzionalità al riavvio del sistema ospitante. In particolare, per garantire adeguatamente l'avvio dei servizi, è utile modificare i parametri `restart: on-failure` in `restart: unless-stopped`. Per i servizi `gsa` ed `nginx` tale parametro risulta assente e, per quanto sopra esposto, è opportuno impostarlo.

4. Concetti Fondamentali di GVM/OpenVAS

L'uso del software GVM/OpenVAS avviene tramite Web GUI, chiamata Greenbone Security Assistant (GSA), che astrae la complessità del motore OpenVAS, permettendo una più agevole gestione delle scansioni. Nei successivi paragrafi sono riportati i concetti chiave dell'architettura di OpenVAS.

4.1 Feeds

I flussi di dati (feeds) usati da OpenVAS per le scansioni sono i seguenti:

- *NVT* (Network Vulnerability Tests) – test esecutivi lanciati da OpenVAS attivamente contro i *Target* per verificare la presenza di falle. Scritti in *NASL* (Nessus Attack Scripting Language), i NVT definiscono le azioni che uno scanner deve compiere per testare la presenza di una vulnerabilità. L'esecuzione di un NVT dipende, spesso, dal risultato di altri NVT (ad es. un NVT che cerca vulnerabilità su Apache verrà lanciato solo se un NVT precedente ha rilevato che su una determinata porta è in esecuzione un webserver Apache e non Nginx). Gli aggiornamenti, con frequenza quotidiana, sono rilasciati dal team di Greenbone;

- *SCAP* (Security Content Automation Protocol) *Data* – SCAP è uno standard del *NIST* (National Institute of Standards and Technology) usato da Greenbone per catalogare le vulnerabilità e assegnare loro un punteggio di gravità (*CVSS* – Common Vulnerability Scoring System);

- *CERT* (Computer Emergency Response Team) *Data* – feed contenente i bollettini di sicurezza e gli avvisi di allerta emessi da team governativi o istituzionali. Nella Community Edition di OpenVAS, tali dati provengono dal DFN-CERT[2] e dal CERT-Bund[3] del governo tedesco;

- *GVMD* (Greenbone Vulnerability Manager Data*)* – configurazioni di scansione, politiche di conformità, elenchi di porte e formati di report*;*

4.2 Gestione delle scansioni

La gestione delle scansioni si basa sui seguenti concetti chiave:

- *Target* (Figura 8) – perimetro della scansione. Un target rappresenta l'obiettivo di una scansione; può essere un singolo indirizzo IP, un host name, un intervallo di IP, o un'intera subnet (indicata in notazione CIDR). Per ogni target è possibile specificare quali porte e

[2] https://www.dfn-cert.de/
[3] https://www.bsi.bund.de/EN/Themen/Unternehmen-und-Organisationen/Cyber-Sicherheitslage/Reaktion/CERT-Bund/cert-bund_node.html

protocolli da scansionare, ad esempio, tutte le porte TCP e UDP assegnate dalla Internet Assigned Numbers Authority (IANA);

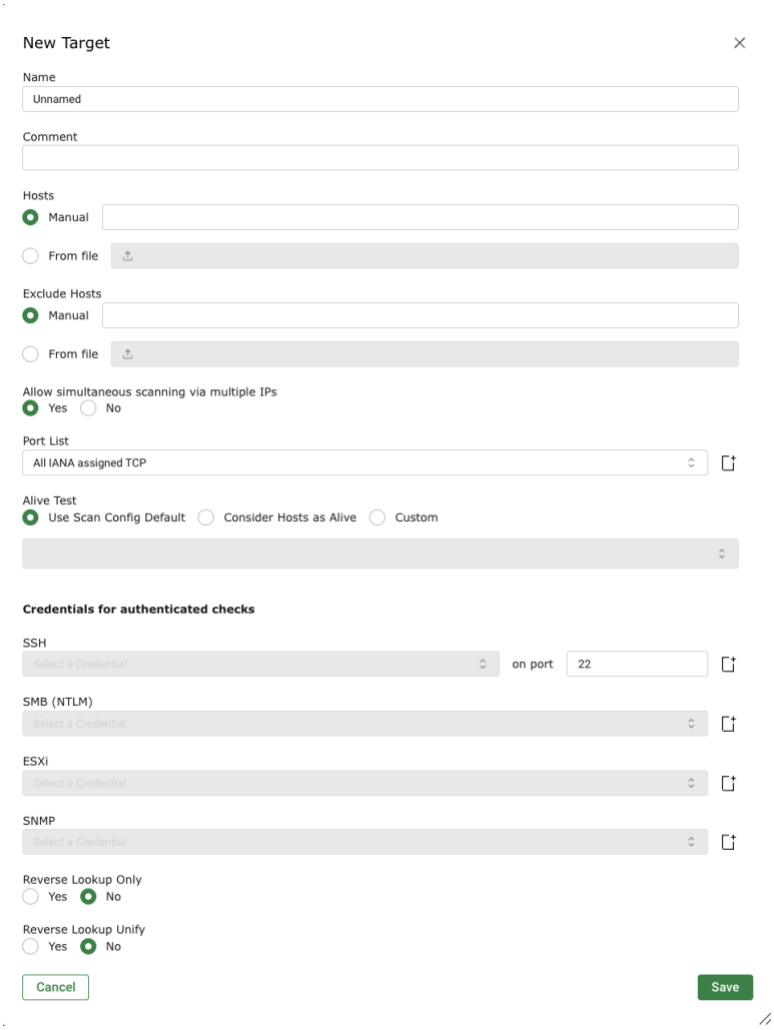

Figura 8 form per la definizione di un Target

- *Scan Config* (Figura 9) – profili di scansione. Definiscono le modalità di analisi dei target, ovvero le famiglie (categorie) dei Network Vulnerability Tests (NVT) e relativi NVT da attivare. I profili sono organizzati in template; ciò consente di definire configurazioni personalizzate, sulla base di template esistenti. Le configurazioni predefinite sono:

    - *empty* – template vuoto, con 0 famiglie e 0 NVT. Utile, come la configurazione *Base*, per la definizione di una configurazione personalizzata;
    - *Base* – template base, con un numero ristretto di NVT (2 famiglie, per un totale di 3 NVT, nella versione 20200827). Il numero di famiglie e di NVT è statico – non è aggiornato automaticamente. Questa configurazione che contiene solo i requisiti

di sistema basilari, affinché il motore OpenVAS possa funzionare – non esegue una scansione di sicurezza;

- *Host Discovery* – configurazione per il discovery degli host connessi. Prevede 2 famiglie e 2 NVT, entrambi statici;
- *System Discovery* – configurazione per il rilevamento dei sistemi operati e tipologia di dispositivi connessi. Contiene 5 famiglie e 30 NVT statici.
- *Discovery* – utile per la scoperta degli host connessi e dei servizi attivi. Il numero di famiglie di NVT (11, nella versione 20201215) è statico, mentre la selezione degli NVT (3324, nella versione 20201215) è dinamica – eventuali nuovi NVT, per le famiglie selezionate, vengono aggiunti e utilizzati automaticamente;
- *Full and fast* – configurazione con il maggior numero di famiglie (60, nella versione 20201215) e NVT (173142, nella versione 20201215). La selezione delle famiglie e di NVT è dinamica;
- *Log4Shell* – configurazione per il rilevamento della vulnerabilità "zero-day" Log4Shell[4]. Contiene 10 famiglie e 29 NVT statici.

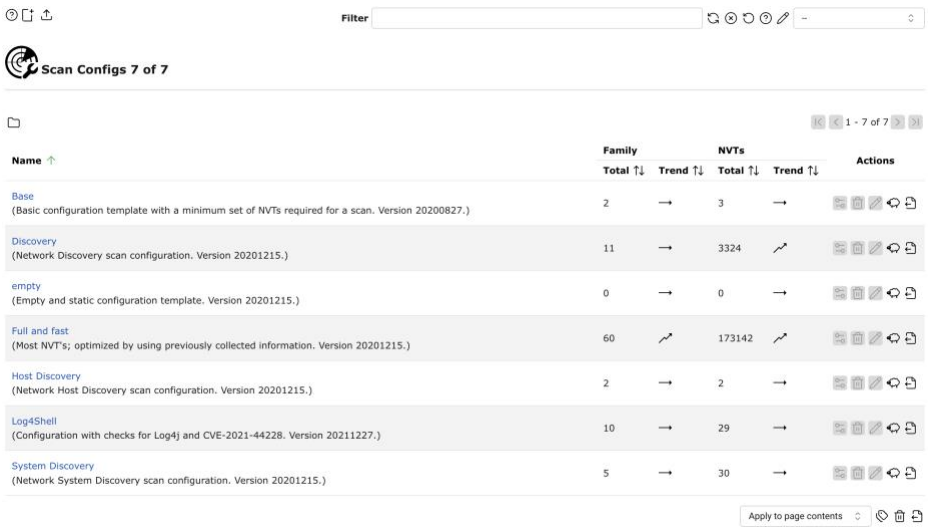

Figura 9 schermata per la configurazione delle scansioni

- *Credentials* (Figura 10) – credenziali di accesso per la scansione autenticata. Consente agli scanner di autenticarsi sugli host per leggere le versioni dei pacchetti software installati al fine

[4] CVE-2021-44228 – https://www.cve.org/CVERecord?id=CVE-2021-44228

ottenere scansione delle vulnerabilità più approfondita. Le tipologie di autenticazione supportate sono: Username e Password; Username e SSH Key; SNMP; S/MIME Certificate; PGP Encryption Key; Password only; Client Certificate;

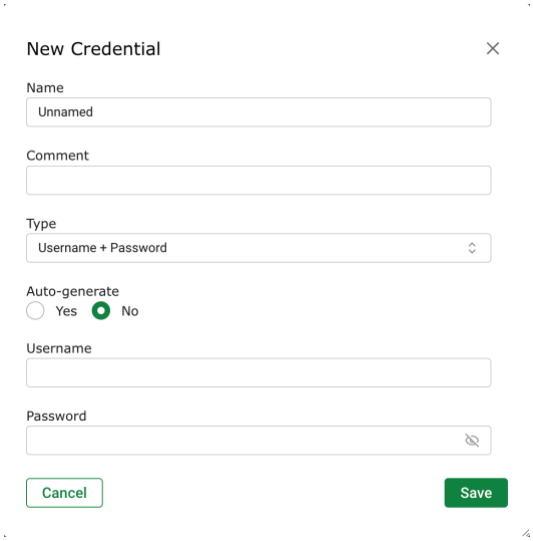

Figura 10 schermata per la definizione di credenziali

- *Task* (Figura 11) – il task consente di definire quando avviare uno Scanner e su quale Target. È possibile scegliere tra lo Scanner OpenVAS e il CVE Scanner. Mentre lo scanner OpenVAS è di tipologia attiva, in quanto interroga i Target, il CVE Scanner è di tipo passivo, verificando i NVT sulla base delle informazioni degli host raccolte da scansioni attive precedenti.

L’esecuzione dei task può avvenire manualmente o possono essere schedulati per essere eseguiti periodicamente.

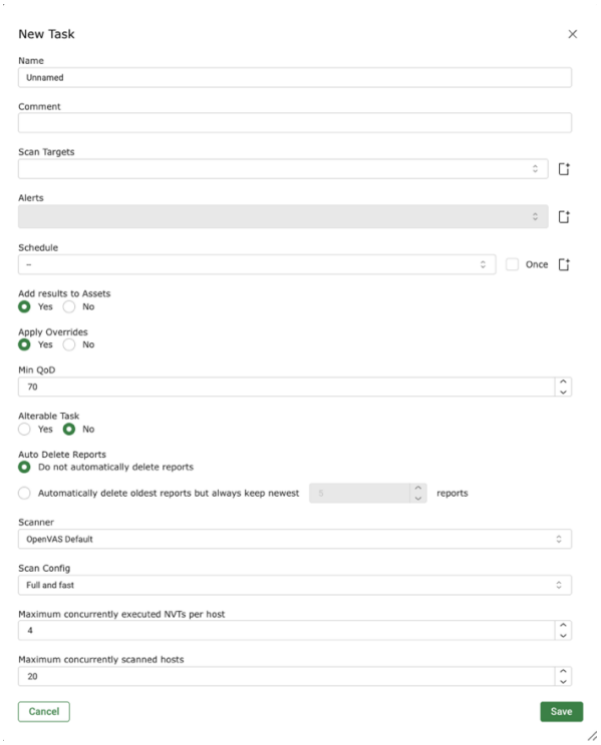

Figura 11 form per la creazione di un nuovo Task

4.3 Report e Gestione del Rischio

Al termine dell’esecuzione dei task, l’analisi dei risultati e la gestione delle vulnerabilità si basano su:

- *Report* (Figura 12) – al termine di ogni task viene generato un report dettagliato delle vulnerabilità rilevate. Le vulnerabilità trovate sono classificate in base ad un punteggio - CVSS (Common Vulnerability Scoring System) - che ne determina il livello di gravità (Low, Medium, High, Critical). Il report include la descrizione della falla, riferimenti diretti (CVE) e, laddove disponibile, la soluzione o le azioni suggerite per la mitigazione del rischio. Greenbone offre inoltre la funzionalità di *Delta Report* (Analisi Differenziale), che permette di confrontare due scansioni eseguite sullo stesso Target in momenti diversi. Confrontando i report, il sistema calcola il "Delta" e classifica ogni vulnerabilità come nuova, risolta o invariata.
  È possibile esportare i report in formato PDF, HTML, CSV, TXT, XML, Anonymous XML.

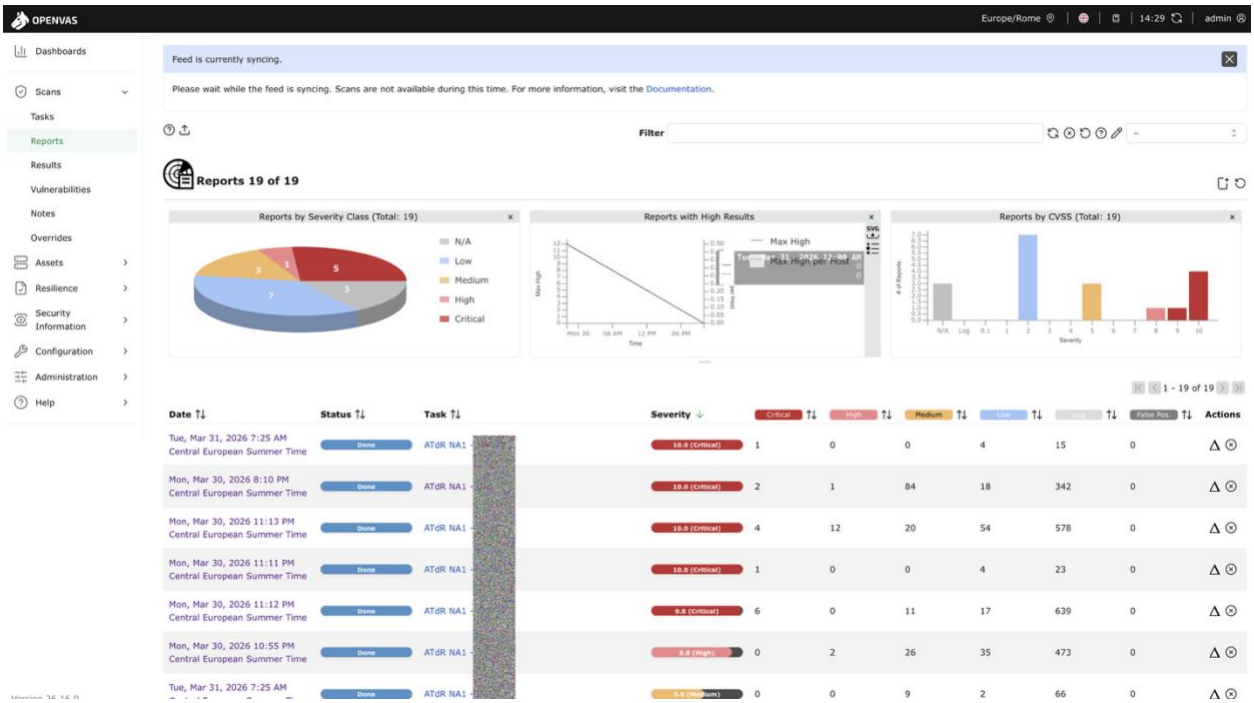

Figura 12 schermata dei report

- *Inventory* (Figura 13, Figura 14, Figura 15) – durante le attività di scansione il sistema memorizza gli host che hanno risposto almeno una volta alle interrogazioni, salvando la data di primo e ultimo rilevamento, il MAC address, l'hardware vendor (*Host Inventory*), i sistemi operativi rilevati (*Operating System Inventory*) e le informazioni sui certificati TLS (*TLS Certificates Inventory*);

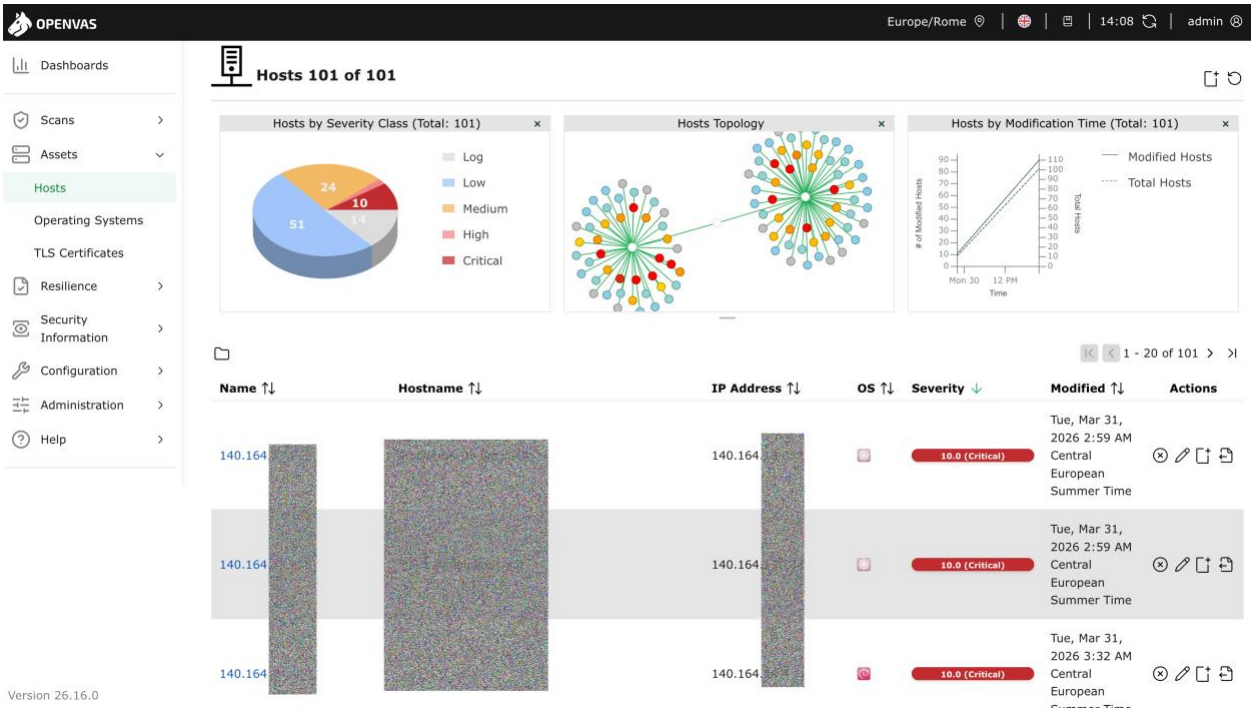

Figura 13 Host Inventory

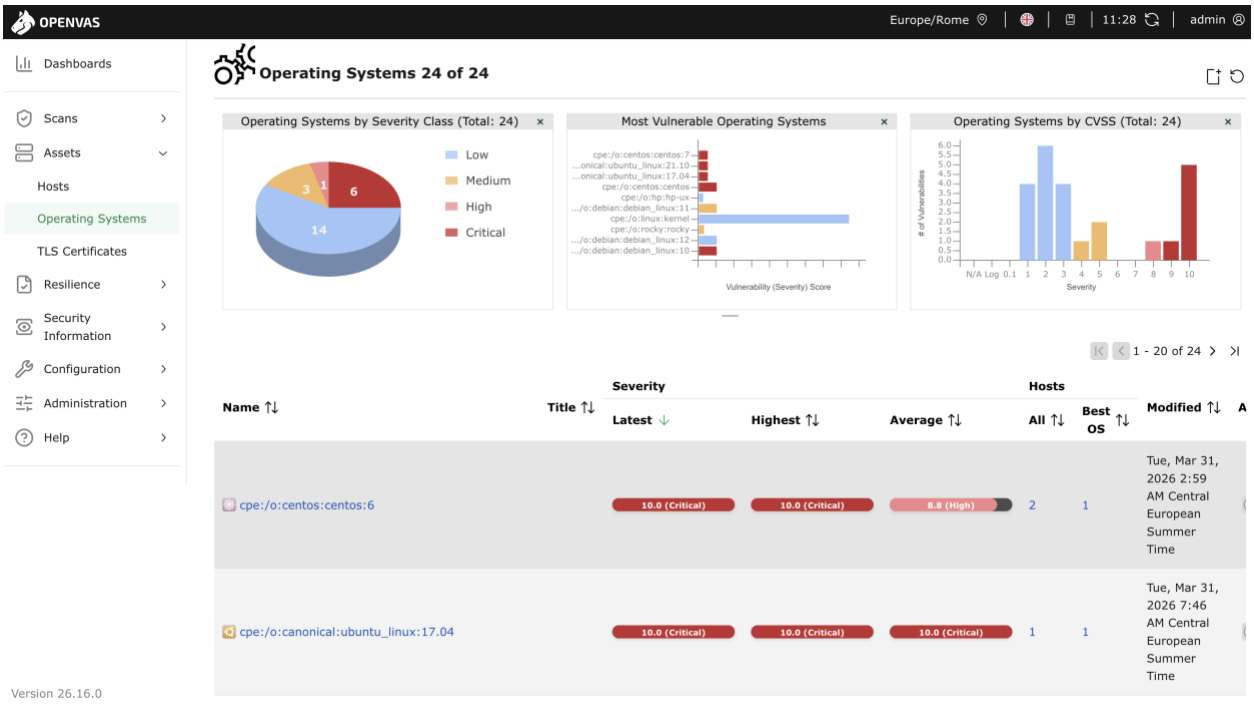

Figura 14 Operating Systems Inventory

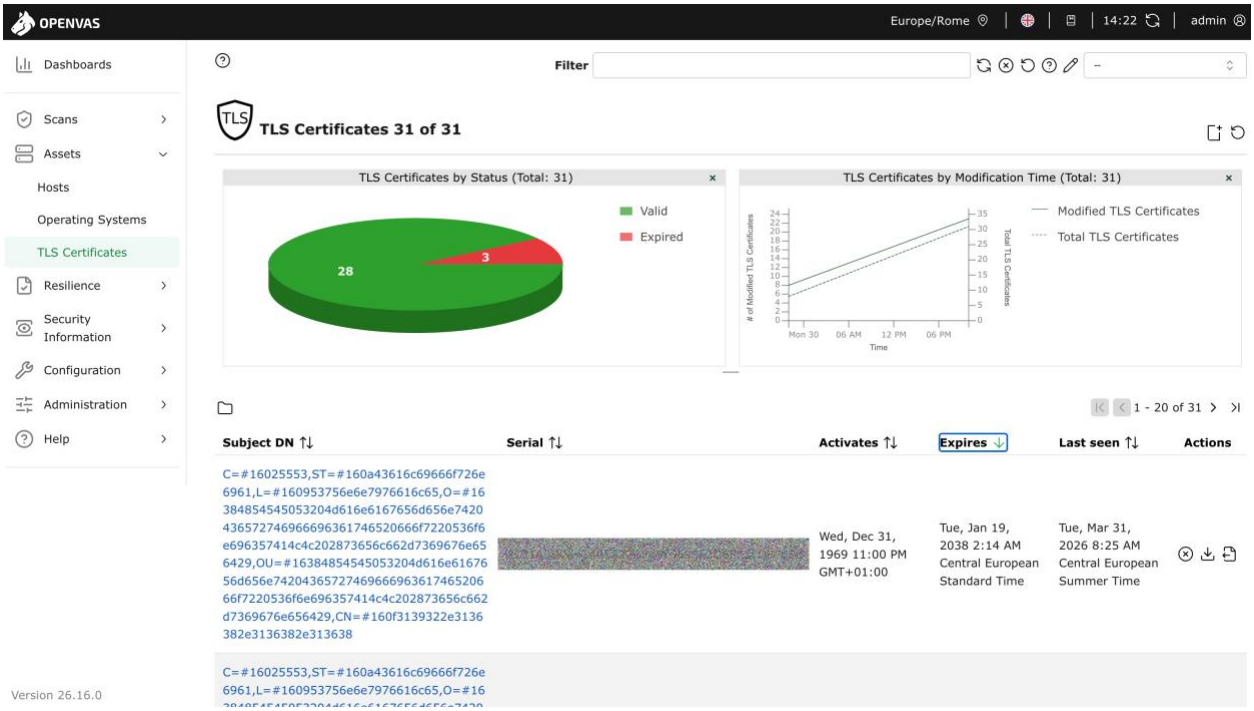

Figura 15 TLS Certificates Inventory

- *Alert* – in Greenbone, gli Alert sono un meccanismo di automazione che permette di eseguire delle azioni a seguito di determinati eventi. La definizione di alert si basa su:
    - *trigger* – evento che attiva l'alert (ad es. il cambio di stato di un task, la variazione di stato di un ticket, …);
    - *condizione/filtro* – condizione che deve essere verificata (ad es. livello di severità, numero di risultati, …);
    - *azione* – azione da compiere (ad es. invio del report via mail, invio di una richiesta HTTP, invio tramite Syslog, …).
- *Overrides* – gli Override permettono di modificare dinamicamente i punteggi di gravità di specifiche vulnerabilità, senza cancellarle. Tale funzionalità è utile per la gestione di *falsi positiv*. La definizione di un overrifde richiede una *motivazione* della sovrascrittura, il *nuovo punteggio* di severità, e lo *scope* (uno specifico IP, una porta, un task, o scope globale) cui si applica l'ovveride.
- *Ticketing* - il sistema di Ticketing integrato in Greenbone permette agli amministratori di legare una vulnerabilità rilevata a un'attività operativa, assegnandola direttamente al team o a una persona incaricata di risolverla. Una funzionalità rilevante riguarda la verifica di

ticket chiusi: se lo scanner non rileva più la vulnerabilità, il ticket viene considerato effettivamente risolto e archiviato con successo; mentre, se viene rilevata ancora la vulnerabilità, il sistema riapre e/o segnala l'incongruenza.

4.4 Conclusioni

L'implementazione del sistema di Vulnerability Assessment basato sulla suite Greenbone/OpenVAS presso l'Area Territoriale di Ricerca Napoli 1 dimostra l'efficacia del deployment di soluzioni enterprise-grade interamente open-source per la protezione di infrastrutture di rete complesse e multi-istituto. L'adozione di questa piattaforma ha permesso di transitare da una postura di sicurezza puramente reattiva o perimetrale a una strategia di difesa proattiva, capace di mappare costantemente gli asset informatici e di identificare tempestivamente servizi obsoleti, certificati scaduti o configurazioni errate prima che possano essere sfruttati da attori malevoli. Oltre all'evidente beneficio tecnico e al significativo abbattimento dei costi di licenza rispetto alle alternative commerciali proprietarie, l'approccio on-premise implementato garantisce la piena sovranità e riservatezza dei dati di vulnerabilità, che rimangono confinati all'interno del perimetro dell'Ente. Sotto il profilo normativo, l'attività svolta costituisce un pilastro fondamentale nel percorso di allineamento e conformità del CNR ai severi requisiti di gestione del rischio e resilienza cyber imposti dalla Direttiva NIS2. Infine, le ottimizzazioni sistemistiche dettagliate nelle appendici — relative alla rotazione dei log, ai livelli di verbosità, all'accesso remoto e alla notifica degli alert via SMTP — assicurano la sostenibilità e la manutenibilità operativa del sistema nel lungo periodo, garantendo all'ATdR NA1 uno strumento scalabile per il monitoraggio continuo della propria postura di sicurezza informatica.

Riferimenti

[1] https://nmap.org/
[2] https://www.gnu.org/licenses/old-licenses/gpl-2.0.html
[3] https://github.com/projectdiscovery/naabu
[4] https://github.com/projectdiscovery/nuclei
[5] https://github.com/queencitycyber/nucleiUI
[6] https://github.com/projectdiscovery/naabu?tab=MIT-1-ov-file#readme
[7] https://nmap.org/man/it/man-nmap-scripting-engine.html
[8] https://www.openvas.org/

[9] https://www.zaproxy.org/
[10] https://owasp.org/Top10/2025/
[11] https://greenbone.github.io/docs/latest/22.4/container/index.html
[12] https://greenbone.github.io/docs/latest/
[13] "*Greenbone Community Edition. Interfaccia web*", Leandro Lanzi, Istituto Nazionale di Fisica Nucleare (INFN) – Commissione Calcolo e Reti. Tutorial days di CCR: Cybersicurezza. Laboratori Nazionali di Frascati, 10-12 ottobre 2022.

Appendice A – livelli di log

Al fine di limitare l'uso dello spazio del disco è possibile modificare il livello di log di OpenVAS. A tal fine, nel file YAML del docker compose, è necessario modificare il parametro `command` del servizio `configure-openvas`:

```
configure-openvas:
    image: greenbone/openvas-scanner:stable
    restart: on-failure
    volumes:
      - openvas_data_vol:/mnt
      - openvas_log_data_vol:/var/log/openvas
    command:
      - /bin/sh
      - -c
      - |
        printf "table_driven_lsc = yes\nopenvasd_server = http://openvasd:80\n" > /mnt/openvas.conf
        sed "s/127/32/" /etc/openvas/openvas_log.conf | sed 's/gvm/openvas/' > /mnt/openvas_log.conf
        chmod 644 /mnt/openvas.conf
        chmod 644 /mnt/openvas_log.conf
```

Individuare la stringa del tipo `"s/127/32/"` e modificare il secondo intero (32, nell'esempio), che indica il livello di log usato. I possibili livelli di log e relativi interi sono:

4	Errors
8	Critical situation
16	Warnings
32	Messages (Livello consigliato per uso standard)
64	Information
128	Debug (Default del compose, verbosità estrema)

Salvato il file `docker-compose.yml`, è necessario ricreare il container di configurazione affinché il nuovo file `openvas_log.conf` venga sovrascritto nei volumi condivisi:

```
docker compose -p greenbone-community-edition down
docker compose -p greenbone-community-edition up -d
```

Appendice B – log rotation

Per limitare l'uso dello spazio disco è possibile configurare `logrotate`, sul sistema host, sui file di log salvati nei volumi. A tal fine è possibile creare un file di configurazione `/etc/logrotate.d/greenbone-docker` il cui contenuto, di esempio, potrebbe essere il seguente:

```
/var/lib/docker/volumes/*_openvas_log_data_vol/_data/*.log
/var/lib/docker/volumes/*_gvmd_data_vol/_data/*.log {
    daily
    rotate 7
    size 500M
    compress
    delaycompress
    missingok
    notifempty
    copytruncate
}
```

Appendice C – accesso remoto alla WebGUI

Per consentire l'accesso alla WebGUI di OpenVAS è necessario modificare le porte di ascolto del servizio `nginx`:

```
vim docker-compose.yml
```

In particolare, è necessario eliminare l'indirizzo IP della macchina host locale dal mapping:

Da:

```
nginx:
    …
    ports:
      - 127.0.0.1:443:443
      - 127.0.0.1:9392:9392
    …
```

Successivamente, è necessario riavviare il servizio applicando le modifiche:

```
docker compose -p greenbone-community-edition down
docker compose -p greenbone-community-edition up -d
```

Appendice D – configurazione SMTP per gli alert

Il container `gvmd` include il client di posta `msmtp`. Per abilitare le notifiche via SMTP, è sufficiente modificare il file `docker-compose.yml`, aggiungendo le opportune variabili d'ambiente del servizio `gvmd`:

```
gvmd:
    …
    environment:
      MTA_HOST: "smtp.server.com"      # Indirizzo del server SMTP
      MTA_PORT: "587"                  # Porta (587 per STARTTLS, 465 per SMTPS)
      MTA_TLS: "on"                    # Abilita la crittografia
      MTA_STARTTLS: "on"               # "on" se la porta è la 587, "off" se è la 465
```

```
      MTA_AUTH: "on"                    # Abilita l'autenticazione
      MTA_USER: "email@dominio.com"     # nome utente SMTP
      MTA_PASSWORD: "Password"          # password dell'utente SMTP
      MTA_FROM: "email@dominio.com"     # Indirizzo mittente
      MTA_TLS_CERTCHECK: "on"           # Verifica certificato del server (consigliato "on")
```

Aggiunte le variabili d'ambiente è necessario riavviare il servizio:

```
docker compose -p greenbone-community-edition down
docker compose -p greenbone-community-edition up -d
```

Se necessario, è possibile visualizzare i log di `msmtp` con il comando:

```
docker logs greenbone-community-edition-gvmd-1 | grep -i msmtp
```