ISTITUTO NAZIONALE DI FISICA NUCLEARE

SEZIONE DI FIRENZE

INFN-22-02/CCR

6 dicembre 2021
CCR-59/2021/P

INSTALLAZIONE, CONFIGURAZIONE E MANUTENZIONE DI UN SERVER PER SCANSIONI DI VULNERABILITÀ NON INVASIVE

Leandro Lanzi[1]

[1)] *INFN Sezione di Firenze, Dip. Fisica, Polo Scientifico di Sesto F.no, Univ. di Firenze, I-50019 Sesto Fiorentino (FI), Italia*

Abstract

L'obiettivo di questa guida è fornire un supporto da un punto di vista sistemistico per l'installazione, la corretta configurazione e la manutenzione di uno o più server per effettuare scansioni di vulnerabilità non invasive tramite il framework open source GreenboneVulnerability Manager.

Published by
Laboratori Nazionali di Frascati

1 Introduzione

Con la *circolare n. 2 del 18 aprile 2017* [1], l'Agenzia per l'Italia digitale (di seguito anche AgID), dando seguito alla Direttiva del Presidente del Consiglio dei Ministri del 1^o agosto 2015 che imponeva l'adozione di standard minimi di prevenzione e reazione ad eventi cibernetici[1], ha indicato alle pubbliche amministrazioni le misure minime (di seguito anche MM) per la sicurezza ICT,[2] dirette a contrastare le minacce più comuni e frequenti cui sono soggetti i sistemi informativi. Fra le altre cose, le MM di AgID richiedono quanto segue[3]

- ABSC ID 4.1.1: "Ad ogni modifica significativa della configurazione eseguire la ricerca delle vulnerabilità su tutti i sistemi in rete con strumenti automatici che forniscano a ciascun amministratore di sistema report con indicazioni delle vulnerabilità più critiche";

- ABSC ID 4.4.1: "Assicurare che gli strumenti di scansione delle vulnerabilità utilizzati siano regolarmente aggiornati con tutte le più rilevanti vulnerabilità di sicurezza".

Nel *Disciplinare per l'uso delle risorse informatiche INFN* [3] approvato con delibera del Consiglio Direttivo n. 15443 [4] viene indicato che "Gli Utenti che hanno privilegi di amministratore sui loro sistemi (p.e. laptop) sono tenuti a prendere visione dei relativi documenti con le Norme d'uso, in attuazione della Circolare AgID 18/04/2017 n. 2/2017, e a seguirne scrupolosamente le indicazioni".

Nel documento INFN relativo all'implementazione delle misure minime di sicurezza nell'INFN [2] è riportato che "Il gruppo Auditing esegue scansioni trimestrali con Nessus[4] su tutti i nodi visibili all'esterno delle LAN. Lo stesso gruppo ha preparato una distribuzione Linux, adattata alle nostre necessità (INFNKALI[5]) che gli amministratori locali utilizzeranno per le scansioni all'interno delle loro reti locali" e che "INFNKALI viene aggiornata quotidianamente".

La distribuzione INFNKALI è stata mantenuta aggiornata dal sottoscritto giornalmente dal 2017 al 2020 ma, visto che le scansioni di sicurezza sono diventate un'attività da ripetere costantemente, risulta più efficiente utilizzare una procedura di installazione che permetta di avere uno strumento duraturo e semplice da gestire per compiere questo lavoro.

[1] Il termine "cibernetici" è quello che viene riportato talvolta in leggi, regolamenti e norme e fa riferimento al termine inglese "cyber" usato nel contesto della "cyber security".

[2] Con ICT (information communication technology) si intendono le tecnologie dell'informazione e della comunicazione cioè l'insieme dei metodi e delle tecniche utilizzate nella trasmissione, ricezione ed elaborazione di dati e informazioni (tecnologie digitali comprese).

[3] ABSC: AgID Basic Security Control(s) Id Number.

[4] Nessus di Tenable è un software proprietario per vulnerability assessment [10]. Come descritto in seguito nella sezione 3.1 a pagina 4, da un fork di Nessus è nato GNessUs che poi è diventato OpenVAS (Open Vulnerability Assessment System).

[5] Questa distribuzione è stata anche indicata col nome di KalINFN [5].

L'obiettivo di questa guida è fornire un supporto da un punto di vista sistemistico per l'installazione, la corretta configurazione e la manutenzione di uno o più server per effettuare scansioni di vulnerabilità non invasive così come previsto dalla normativa attuale e dai regolamenti dell'Ente.

2 Scelta del software

Esistono vari software per compiere scansioni di vulnerabilità. Fra i più famosi ci sono: Tenable Nessus, Metasploit Framework[6], Intruder[7], Greenbone Vulnerability Manager. Esistono anche software per compiere scansioni di vulnerabilità su tipologie di bersagli specifici come ad esempio i siti web, quali Nikto[8], Netsparker[9], Acunetix[10], ThreatSpotter[11].

Questi software possono essere open source o proprietari e talvolta ne esistono versioni community[12] e trial [13].

Non è stato scelto Metasploit Framework perché è indicato più per compiere penetration testing che non vulnerability scan.

Le versioni trial non sono state considerate perché si cerca una soluzione stabile e duratura[14].

Il software proprietario non è stato preso in considerazione soprattutto per i costi eccessivi che una licenza a pagamento comporterebbe per il suo utilizzo nella rete INFN.

[6]Metasploit Framework [11] è uno strumento open source per lo sviluppo e l'esecuzione di exploits ai danni di una macchina remota che fa parte del progetto Metasploit di Rapid7.

[7]Intruder [12] è un software proprietario per scansioni di vulnerabilità

[8]Nikto [13] è un software open source per scansioni di vulnerabilità specifiche per server web.

[9]Netsparker di Invicti [14] è un software proprietario per scansioni di vulnerabilità specifiche di tipo XSS, SQL Injection e altre vulnerabilità nelle applicazioni web.

[10]Netsparker di Invicti [15] è un software proprietario per scansioni di vulnerabilità specifiche per server web.

[11]ThreatSpotter [16] è uno scanner di vulnerabilità indirizzato agli ambienti cloud come Amazon Web Services (AWS), Microsoft Azure, e Google Cloud Platform (GCP).

[12]Per versione community di un software si intende una versione gratuita liberamente utilizzabile di un software che viene distribuito anche in una versione enterprise con licenza a pagamento. Solitamente le versioni community hanno alcune funzionalità in meno rispetto alle versioni enterprise e mancano di assistenza.

[13]Per versione trial di un software si intende una versione gratuita liberamente utilizzabile per un limitato periodo di tempo (tipicamente una settimana o un mese). Solitamente allo scadere del periodo di trial il software smette di funzionare ed è necessario passare alla versione enterprise con licenza a pagamento.

[14]Come indicato di seguito in 3.2 a pagina 5, esistono due versioni enterprise di GVM con licenze a pagamento, cioè GPE (Greenbone Professional Edition) e GCS (Greenbone Cloud Services) per le quali esistono le versioni trial . Per quanto riguarda GCS la sua versione trial ha una durata di soli 14 giorni e permette la scansione di soli 22 IP. La versione trial di GPE è nominata Greenbone Security Manager Trial (GSM Trial), ha alcune funzionalità ridotte, per esempio, non include la gestione dei certificati TLS, utilizza comunque i Greenbone Community Feed come GVM ed inoltre è disponibile solo come macchina virtuale VMware Workstation Player, VMware Workstation Pro e Oracle Virtual Box. Anche se non ci sono limiti di tempo all'utilizzo di questo secondo prodotto, risulta più indicato un uso su un PC personale, appunto per una semplice prova trattandosi di un prodotto "Trial", piuttosto che su un server per una configurazione stabile e duratura.

È stato scelto di concentrarsi sulla versione community di Greenbone Vulnerability Manager, precedentemente conosciuto come OpenVAS, perché si tratta di un software open source, è supportato da una comunità numerosa, ha una licenza che ne permette un uso adatto alle esigenze dell'Ente. Inoltre è disponibile liberamente una copiosa e puntuale documentazione che ne descrive nel dettaglio, soprattutto da un punto di vista tecnico e pratico, ogni aspetto, caratteristica e funzionalità[15].

Inoltre la scelta di software open source completamente free risulta in pieno accordo con la missione ed i principi di un Ente Pubblico di Ricerca quale è l'INFN evitando di cadere in insidiose operazioni, note come vendor lock-in, messe in atto sempre più frequentemente da parte dei produttori di software con licenze a pagamento.

3 Descrizione di Greenbone Vulnerability Manager

Greenbone Vulnerability Manager (GVM) è un framework open source che utilizza diversi servizi e strumenti per effettuare scansioni di vulnerabilità e gestirne la sanificazione.

3.1 Brevissima storia del progetto GVM/OpenVAS

Inizialmente GVM è stato sviluppato con il nome di OpenVAS e prima ancora col nome GNessUs, come fork del software Nessus dopo che i suoi sviluppatori (Tenable Network Security) avevano trasformato in software proprietario il codice iniziale open source nell'ottobre del 2005.

Quando il progetto OpenVAS è stato lanciato, consisteva solo in un motore per la scansione delle vulnerabilità.

Poco dopo è stata fondata Greenbone Networks col fine di sviluppare e offrire supporto professionale al prodotto. Greenbone Networks ha iniziato a guidare lo sviluppo di OpenVAS, ha aggiunto diversi componenti e lo ha trasformato in un software completo e articolato di gestione delle vulnerabilità mantenendo i valori del software libero.

Dopo il rilascio della versione 9 di OpenVAS, il progetto open source è stato rinominato Greenbone Vulnerability Management (GVM). A partire da GVM 10, il termine OpenVAS viene utilizzato solo per la componente scanner come era all'inizio del progetto.

CRITERIA	GSE	GPE	GCS
Setup	Individual selection of operating system and hardware Built on own responsibility or installation of community packages	Turnkey solution Simple and uncomplicated setup within shortest time	Simple account registration, and configuration within shortest time
Feed Compatibility	Established on own responsibility	Assured with SLA	Assured with SLA
Performance	Optimized on own responsibility	Optimized for hardware	Variable according to requirements
Backup/Recovery	Solved individually	Integrated	Integrated
Fixes/Improvements	Managed on own responsibility	Assured with SLA	Assured with SLA
Support	Via (external) community on voluntary basis	Assured with SLA	Assured with SLA
Software Updates	Manual source build updates and manual migration of data	Regularly and seamlessly	Continuously

Figura 1: Confronto fra le caratteristiche di distribuzione, implementazioni e supporto delle tre versioni di GVM: GSE, GPE e GCS [6].

3.2 Versioni disponibili

GVM è rilasciato sotto forma di sorgenti open source completamente liberi col nome di Greenbone Source Edition (GSE).

Parallelamente a GSE viene mantenuta una versione Enterprise, Greenbone Professional Edition (GPE), e una versione Cloud, Greenbone Cloud Services (GCS), entrambe con licenze a pagamento.

Nella figura 1 sono confrontate le tre versioni in termini di distribuzione (delivery), implementazione (deployment) e supporto (support) mentre in figura 2 a pagina 6 in termini di caratteristiche e funzionalità (features). [6]

3.3 Architettura del framework

Nella figura 3 a pagina 7 è riportato uno schema che descrive com'è organizzato il framework GVM.

[15]Oltre alla documentazione specifica per la versione open source, disponibile su `github` alla pagina `https://greenbone.github.io/docs/index.html`, è utilizzabile tutta la documentazione della versione enterprise, ovviamente solo per la parte comune alla versione open source, accessibile alla pagina `https://docs.greenbone.net/`. In particolare la parte riguardante la documentazione delle API (Application Programming Interface) nella sezione "API Documentation" della pagina indicata sopra, soprattutto per quanto riguarda GMP (Greenbone Management Protocol), è particolarmente utile per gestire le scansioni da riga di comando tramite comandi di shell o script python. Anche il forum `https://community.greenbone.net/` è spesso utile per risolvere eventuali problemi che si possono presentare.

CRITERIA	GSE	GPE	GCS
Possibilities for Updates & Feed	Only Greenbone Community Feed	Daily automatic Possible via GSM configurable sync ports, redundant proxy servers, USB or FTP Airgap, or GSM master	Daily automatic
System Update	Dependent on distribution or on own responsibility	Contains security updates Update from any version to latest release possible Grace periods for EoL and LTS Migration of data and configurations between appliances and versions	Automatic Continuous security and platform updates
Protocols	Configured and set up on own responsibility	NTP, GMP, OSP, HTTPS, SSH,SNMPv2, SNMP, Syslog, IPv6, LDAP, RADIUS and more	NTP, GMP, HTTPS, SSH, SNMPv2, SNMP, Syslog, LDAP, RADIUS and more
Integrations and Connectors	Not available	Different vendors like PaloAlto, Fortinet, Cisco FireSight, Nagios, Splunk, Verinice and more	RESTful API for all functionalities
Backup/Recovery	Solved individually	Backup for user data, system data via LVM, transfer via SCP or USB	Automatic
Alerts/Schedules	Configured on own responsibility via operating system	Via e-mail, HTTP, SMS, connector to a SIEM or ticket system and more Complete scheduling possible	Via e-mail, Slack or Microsoft Teams
Scan Architecture	Not available	Master/sensor, Airgap inside of high security zones	Cloud scanner, gateway components for internal scans

Figura 2: Confronto fra le funzionalità (features) delle tre versioni di GVM: GSE, GPE e GCS [6].

GVM è raggruppato in tre parti principali:

- due applicazione di scansione (OpenVAS e OSP Scanner),
- il demone Greenbone Vulnerability Manager Daemone (gvmd),
- l'applicazione Greenbone Security Assistant (GSA) con il demone Greenbone Security Assistant Daemon (gsad)

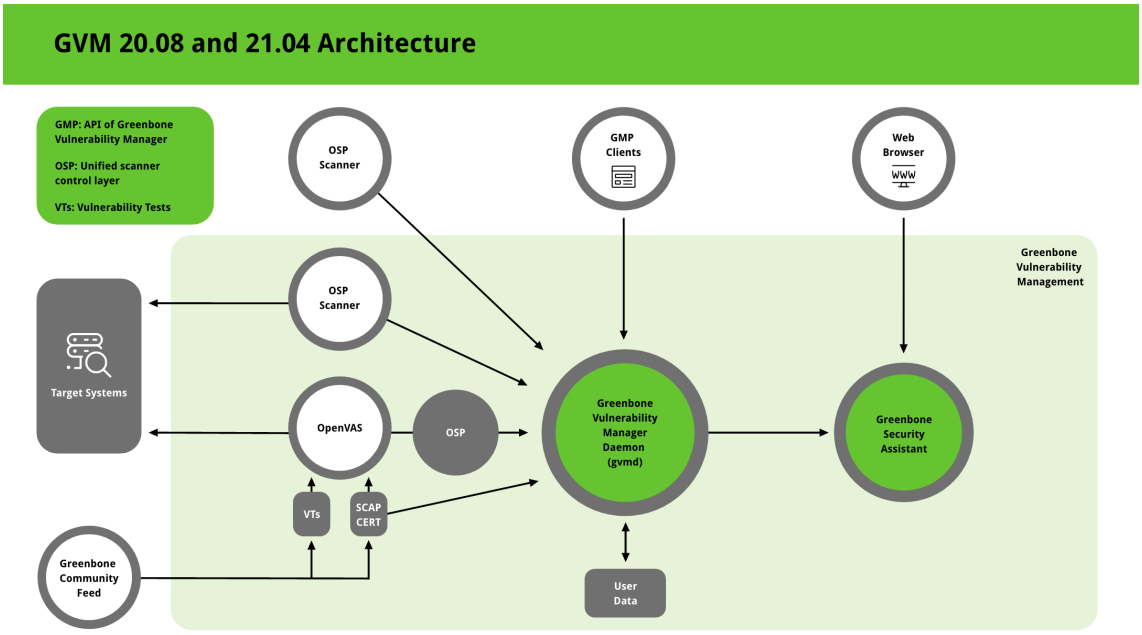

Figura 3: Architettura di GVM [7].

3.3.1 Greenbone Vulnerability Manager Daemon (gvmd)

Il Greenbone Vulnerability Manager Daemon (gvmd) è il servizio centrale che coordina tutti i servizi che compongono il framework dai processi di scansione, ai processi di controllo dei test di vulnerabilità e di accesso ai dati tramite il protocollo stateless Greenbone Management Protocol (GMP) basato interamente su messaggi XML.

gvmd controlla lo scanner OpenVAS tramite il protocollo Open Scanner Protocol (OSP).

gvmd controlla anche un database SQL (PostgreSQL) dove sono memorizzate tutte le configurazioni e i dati dei risultati delle scansioni.

Anche la gestione degli utenti è demandata a gvmd compreso il controllo dei permessi e degli accessi con gruppi e ruoli.

Infine il servizio ha un sistema interno di runtime per compiti ed eventi programmati.

3.3.2 Greenbone Security Assistant (GSA)

Il Greenbone Security Assistant (GSA) è l'interfaccia web di GVM con cui gli utenti possono controllare le scansioni e accedere alle informazioni sulle vulnerabilità.

GSA si connette a gvmd tramite il server web Greenbone Security Assistant Daemon (gsad) per fornire un'applicazione web completa per la gestione delle vulnerabilità.

La comunicazione avviene utilizzando il Greenbone Management Protocol (GMP) con cui l'utente può anche comunicare direttamente utilizzando diversi strumenti da riga di comando.

3.3.3 *OpenVAS Scanner e OSP Scanner*

Lo scanner principale di GVM è OpenVAS Scanner: si tratta di un motore di scansione completo che esegue test di vulnerabilità (VT) sui sistemi che si intende analizzare (sistemi target).

Per compiere questo lavoro, utilizza feed aggiornati quotidianamente e completi:

- i Greenbone Security Feed (GSF) commerciali, completi, senza nessuna limitazione;
- i Greenbone Community Feed (GCF) disponibile gratuitamente.

Per un confronto fra GSF e GCF, che spesso viene affrontata in modo non corretto da molti sostenitori del software commerciale, si riporta un'unica frase del white paper di Greenbone Networks GmbH [8] dove sono analizzate nel dettaglio tutte le differenze fra GSE, GPE e GCS: "All in all, the Community Feed encompasses about 30% less VTs than the professional feed"[16].

Oltre ad OpenVAS gli utenti possono sviluppare un proprio motore di scansione (OSP Scanner) e connetterlo a GVM utilizzando il framework generico `OSPd` che permette appunto a GVM di interagire con scanner diversi tramite il protocollo OSP (Open Scanner Protocol) basato su messaggi XML.

4 Installazione di GVM da sorgenti e configurazione iniziale

La procedura di seguito descritta è da considerarsi un aggiornamento di quanto riportato in [9] in cui viene proposto l'uso di uno script appositamente sviluppato per installare la versione precedente di GVM (versione 20.08) su un server con la versione precedente di Debian GNU/Linux (versione 10, buster).

Per l'installazione ci si riferisce ad una macchina (fisica o virtuale o ad un container linux[17]) con le seguenti caratteristiche.

- Almeno 4 GB di RAM (meglio 8 GB).
- Almeno 2 CPU (meglio 4 CPU).
- Almeno 15 GB di spazio disco (meglio 30 GB): partendo da una distribuzione base di Debian Bullseye (368 MB) alla fine dell'installazione di GVM lo spazio disco occupato complessivamente dal sistema operativo e da GVM è circa 8.2 GB.
- Un'installazione base di Debian GNU/Linux Bullseye (v.11).

[16] "Tutto sommato, il Community Feed comprende circa il 30% in meno di VT rispetto al feed professionale"

[17] Per container Linux non si intende un docker container ma piuttosto un classico LXC.

Si installa la versione 21.04 di GVM che, alla data di stesura del presente documento, è la versione stabile. Trattandosi potenzialmente di un ambiente multi-utente, in accordo con quanto previsto dalle MM di AgID, si eseguono tutte le operazioni che richiedono permessi amministrativi col comando `sudo`.

La scelta di non utilizzare docker-container è dovuta al fatto che, data la complessità e la peculiarità dei VT usati per le scansioni, si preferisce non inserire livelli di virtualizzazione troppo elevati fra l'hardware di rete e il sistema che gestisce le scansioni per non rischiare di ridurre l'efficacia dei VT.

Tutte le operazioni necessarie a compiere l'installazione da sorgenti di GVM 21.04 sono riportate nell'appendice 7.1 a pagina 13.

Per semplificare le operazioni di installazione e configurazione è disponibile uno script automatico scaricabile tramite `git` dal repository del software INFN seguendo le indicazioni riportate in[18]:
`https://baltig.infn.it/llanzi/ossoverde_2021`.

Alla fine dell'installazione è possibile accedere all'interfaccia di GVM collegandosi con un browser web tramite il protocollo HTTPS alla porta 443 del server.

Le credenziali dell'utente di amministrazione per accedere all'interfaccia web sono quelle scelte in fase di installazione con la procedura riportata nella sezione 7.3.5 a pagina 29.

5 Interfaccia web di GVM

Una volta completata l'installazione di tutti i pacchetti e la configurazione iniziale, la gestione di GVM e delle scansioni viene effettuata quasi completamente attraverso l'interfaccia web che permette, per esempio, di compiere le seguenti azioni:

- gestione degli utenti, gruppi, ruoli e permessi;
- configurazione dei tipi di scansione, delle porte e dei protocolli da usare, dei bersagli da sottoporre a scansione;
- gestione delle operazioni di scansione;
- gestione dei risultati delle scansioni e dei report.

6 Manutenzione di GVM

Per ottenere risultati utili dalle scansioni di vulnerabilità è necessario mantenere aggiornati gli strumenti di scansione.

[18] Prima di poter scaricare lo script è necessario richiedere l'accesso al progetto `ossoverde_2021` con una mail a `leandro.lanzi@fi.infn.it`.

Si devono quindi costantemente aggiornare i test di vulnerabilità (Network Vulnerability Test, NVT) e altre informazioni utilizzate dallo scanner di vulnerabilità (OpenVAS) tramite i GCF (vedi sezione 3.3.3 a pagina 8).

I GCF sono organizzati secondo le seguenti 4 tipologie, sono aggiornati separatamente e sono consultabili tramite l'interfaccia web (Administration → Feed Status).

- NVT Feed: relativi ai Network Vulnerability Test.
- SCAP Feed: relativi a CVE (Common Vulnerabilities and Exposures), CPE (Common Platform Enumeration) e OVAL (Open Vulnerability and Assessment Language)[19].
- CERT Feed: relativi ai bollettini di CERT-Bund (Computer Emergency Response Team of the German Federal Office for Information Security) e di DFN-CERT (CERT responsabile di centinaia di università associate con la German Research and Education Network.
- GVMD_DATA: relativi a parametri, configurazioni e dati di funzionamento di GVM:
  - Compliance Policies,
  - Port Lists,
  - Report Formats,
  - Scan Config.

6.1 Operazioni di Manutenzione

L'aggiornamento dei GFC è un operazione semplice ma al tempo stesso molto delicata perché da un lato molte operazioni avvengono in background, con tempi anche dell'ordine di qualche decina di minuti, dall'altro alcune di queste operazioni devono essere eseguite in un ben preciso ordine[20] e necessitano che tutte le operazioni precedenti si siano concluse. Per questo l'utente non può semplicemente eseguire in sequenza i comandi di aggiornamento ma deve assicurarsi che tutte le operazioni, anche quelle in background, si siano concluse prima di andare oltre nella procedura. L'osservazione dell'output dei comandi e dei file di log permette di avere informazioni specifiche sull'andamento degli aggiornamenti. Per ciascuno dei comandi di aggiornamento descritti di seguito si indicherà cosa ricercare nei file di log per assicurarsi dell'esito dei singoli comandi e per procedere alle fasi successive dell'aggiornamento[21].

[19] Per un approfondimento su CVE, CPE e OVAL consultare in appendice la sezione 7.4 a pagina 34.

[20] In particolare l'aggiornamento dei CERT feed deve essere effettuata successivamente al compimento della procedura di aggiornamento dei SCAP feed.

[21] Qualora si decida di affidare l'aggiornamento dei GFC a comandi in cron, si consiglia di utilizzare un unico script nel quale si attenda almeno 30 minuti fra il termine dell'esecuzione di un comando di aggiornamento e l'inizio del successivo

- Aggiornamento dei NVT Feed

  ```
  sudo -u gvm greenbone-nvt-sync
  ```

  L'output del comando dà indicazioni se vengono scaricati nuovi file. In caso di aggiornamenti, nel file di log `/var/log/gvm/openvas.log` compare una voce del tipo "`Updated NVT cache from version YYYYMMDDHHMM to YYYYMMDDHHMM`", mentre nel file di log `/var/log/gvm/gvmd.log` compaiono varie voci e la conclusione del processo di aggiornamento è segnalata con "`update_nvt_cache_retry: rebuild successful`".

- Aggiornamento dei SCAP Feed

  ```
  sudo -u gvm greenbone-feed-sync --type SCAP
  ```

  L'output del comando dà indicazioni se vengono scaricati nuovi file. In tal caso nel file di log `/var/log/gvm/gvmd.log` compaiono varie voci che riportano "`Updating`". Al termine della procedura di aggiornamento in background nel file di log compare la voce "`update_scap_end: Updating SCAP info succeeded`". Nel caso invece che non venga scaricato nessun file, nel log non si osserva nessuna voce.

- Aggiornamento dei CERT Feed

  ```
  sudo -u gvm greenbone-feed-sync --type CERT
  ```

  L'output del comando dà indicazioni se vengono scaricati nuovi file, mentre nel file di log `/var/log/gvm/gvmd.log` compaiono varie voci e la conclusione del processo di aggiornamento è segnalata con "`sync_cert: Updating CERT info succeeded.`".

- Aggiornamento dei GVM_DATA Feed

  ```
  sudo -u gvm greenbone-feed-sync --type GVMD_DATA
  ```

  L'output del comando dà indicazioni se vengono scaricati nuovi file ma nei log, pur comparendo alcune notifiche in caso di aggiornamenti particolari, non viene data comunicazione della conclusione del processo di aggiornamento. Trattandosi comunque dell'ultima operazione di aggiornamento non è particolarmente critico determinare l'istante preciso in cui termina.

Se si dispone di più server con GVM che escono sulla Internet pubblica con lo stesso gateway è necessario prestare particolare attenzione che queste operazioni di aggiornamento non si sovrappongano: come viene ricordato a conclusione di ogni aggiornamento[22] se si effettuano più aggiornamenti simultaneamente si rischia di essere temporaneamente bloccati.

Il download, cioè la sincronizzazione locale, dei Feed avviene in modo incrementale di default ma, nel caso questa operazione non vada a buon fine, si può forzare la sincronizzazione completa dei feed aggiungendo l'opzione `--rsync` nei quattro comandi riportati sopra.

[22] "Only one sync per time, otherwise the source ip will be temporarily blocked."

7 Appendice

7.1 Installazione di GVM

7.1.1 Promemoria

Di seguito è riportata la procedura di installazione di tutto il framework GVM utilizzando un utente che, usando `sudo`, può eseguire comandi sia come amministratore che come utente `gvm`. Se durante le operazioni di installazione si esegue il `logout` dell'utente, al successivo `login` è necessario ridefinire le seguenti variabili.

```
export PATH=$PATH:/sbin
export BASE_DIR=/usr/src/ossoverde
export SOURCE_DIR=$BASE_DIR/source
export BUILD_DIR=$BASE_DIR/build
export INSTALL_DIR=$BASE_DIR/install
export INSTALL_PREFIX=/usr/local
export GVM_VERSION=21.4.3
export GVM_LIBS_VERSION=$GVM_VERSION
export GVMD_VERSION=21.4.4
export GSA_VERSION=$GVM_VERSION
export OPENVAS_SMB_VERSION=21.4.0
export OPENVAS_SCANNER_VERSION=$GVM_VERSION
```

7.1.2 Prerequisiti

- Il pacchetto `sudo` deve essere installato e l'utente che esegue l'installazione deve appartenere al gruppo `sudo`. Nel caso queste condizioni non siano già verificate eseguire da utente amministratore i seguenti comandi dove si indica con `install_user` l'username dell'utente che effettuerà l'installazione.

  ```
  adduser install_user
  usermod -aG sudo install_user
  ```

- Eseguire il login con l'utente con cui si desidera compiere l'installazione cioè, riferendoci al punto precedente, con l'equivalente dell'utente `install_user`.

- Visualizzazione dell'`username` dell'utente con cui si esegue l'installazione e della sua `PATH`

  ```
  echo $USER
  echo $PATH
  ```

- Nel caso non sia giù presente in `PATH`, è utile avere `/sbin` nella `PATH` dell'utente

```
export PATH=$PATH:/sbin
```

- Creazione dell'utente `gvm` e del gruppo `gvm`.

  ```
  sudo useradd -r -M -U -G sudo -s /bin/bash gvm
  ```

- Configurare l'utente con cui si esegue l'installazione in modo che appartenga al gruppo `gvm`.

  ```
  sudo usermod -aG gvm $USER
  ```

- Per rendere effettive le modifiche relative al gruppo è necessario fare il `logout` e nuovamente il `login` dell'utente con cui si effettua l'installazione.

- Definizione dei percorsi da utilizzare per l'installazione.

  - Directory dove scaricare e compilare i sorgenti dei vari software: `/usr/src/ossoverde`.

    ```
    export BASE_DIR=/usr/src/ossoverde
    sudo mkdir -p $BASE_DIR
    ```

  - Directory dove scaricare i sorgenti: `/usr/src/ossoverde/source`.

    ```
    export SOURCE_DIR=$BASE_DIR/source
    sudo mkdir -p $SOURCE_DIR
    ```

  - Directory dove compilare i sorgenti: `/usr/src/ossoverde/build`.

    ```
    export BUILD_DIR=$BASE_DIR/build
    sudo mkdir -p $BUILD_DIR
    ```

  - Directory dove creare i file per l'installazione del software: `/usr/src/ossoverde/install`.

    ```
    export INSTALL_DIR=$BASE_DIR/install
    sudo mkdir -p $INSTALL_DIR
    ```

- Definizione del PATH dove installare i pacchetti[23] (scelta dell'`Install Prefix`): `/usr/local`.

  ```
  export INSTALL_PREFIX=/usr/local
  ```

- Settare i permessi di scrittura e di accesso al gruppo `gvm` sulle directory create

[23] Si sceglie quella di sistema in modo da non dover aggiungere altri PATH, per esempio, alle librerie.

```
sudo chgrp -R gvm $BASE_DIR
sudo chmod -R g+rwX $BASE_DIR
```

- Installazione dei pacchetti utili alla compilazione di tutti i software che compongono il framework

```
sudo apt update


sudo apt install -y \
build-essential \
curl \
cmake \
pkg-config \
python3 \
python3-pip \
gnupg \
doxygen \
git \
clang-format
```

- Importazione delle chiavi pubbliche di Greenbone per validare i pacchetti che verranno scaricati.

```
curl -O https://www.greenbone.net/GBCommunitySigningKey.asc
gpg --import GBCommunitySigningKey.asc
gpg --edit-key 9823FAA60ED1E580
```

  - All'interno della console di gpg digitare il comando: `trust`.

    ```
    trust
    ```

  - Scegliere l'opzione: `5 = I trust ultimately`.

    ```
    5
    ```

  - Alla domanda `Do you really want to set this key to ultimate trust? (y/N)` rispondere con `y`.

    ```
    y
    ```

  - Uscire dalla console di gpg con il comando: `quit`.

    ```
    quit
    ```

- Scegliere la versione di GVM da installare. Alla data di stesura del presente documento la versione è la: 21.4.3.

```
export GVM_VERSION=21.4.3
```

7.1.3 Installazione di tutte le compontenti del framework GVM

- **gvm-libs**

```
# Setting the gvm-libs version to use
export GVM_LIBS_VERSION=$GVM_VERSION

# Required (and optional) dependencies for gvm-libs
sudo apt install -y \
  libglib2.0-dev \
  libgpgme-dev \
  libgnutls28-dev \
  uuid-dev \
  libssh-gcrypt-dev \
  libhiredis-dev \
  libxml2-dev \
  libpcap-dev \
  libnet1-dev \
  libldap2-dev \
  libradcli-dev

# Downloading the gvm-libs sources

curl -f -L \
https://github.com/greenbone/gvm-libs/ar
   chive/refs/tags/v$GVM_LIBS_VERSION.tar.gz \
-o $SOURCE_DIR/gvm-libs-$GVM_LIBS_VERSION.tar.gz

curl -f -L \
https://github.com/greenbone/gvm-libs/releases/dow
   nload/v$GVM_LIBS_VERSION/gvm-libs-$GVM_LIBS_VERSION.tar.gz.asc \
-o $SOURCE_DIR/gvm-libs-$GVM_LIBS_VERSION.tar.gz.asc

# Verifying the source file
gpg --verify $SOURCE_DIR/gvm-libs-$GVM_LIBS_VERSION.tar.gz.asc \
$SOURCE_DIR/gvm-libs-$GVM_LIBS_VERSION.tar.gz

# Output should be similar to
# gpg: Signature made Mon 11 Oct 2021 02:39:46 PM UTC
# gpg:                using RSA key
                      8AE4BE429B60A59B311C2E739823FAA60ED1E580
```

```
# gpg: Good signature from "Greenbone Community Feed integrity key"
      [ultimate]

# Extracting the tarball
tar -C $SOURCE_DIR -xvzf $SOURCE_DIR/gvm-libs-$GVM_LIBS_VERSION.tar.gz

# Building gvm-libs
mkdir -p $BUILD_DIR/gvm-libs
cd $BUILD_DIR/gvm-libs
cmake $SOURCE_DIR/gvm-libs-$GVM_LIBS_VERSION \
  -DCMAKE_INSTALL_PREFIX=$INSTALL_PREFIX \
  -DCMAKE_BUILD_TYPE=Release \
  -DSYSCONFDIR=/etc \
  -DLOCALSTATEDIR=/var \
  -DGVM_PID_DIR=/run/gvm
make -j$(nproc)

#Installing gvm-libsexport
make DESTDIR=$INSTALL_DIR install
sudo cp -rv $INSTALL_DIR/* /
rm -rf $INSTALL_DIR/*
```

- **gvmd**

```
#Setting the gvmd version to use
export GVMD_VERSION=21.4.4

#Required dependencies for gvmd
sudo apt install -y \
  libglib2.0-dev \
  libgnutls28-dev \
  libpq-dev \
  postgresql-server-dev-13 \
  libical-dev \
  xsltproc \
  rsync \
  texlive-latex-extra \
  texlive-fonts-recommended \
  xmlstarlet \
  zip \
  rpm \
```

```
  fakeroot \
  dpkg \
  nsis \
  gnupg \
  gpgsm \
  wget \
  sshpass \
  openssh-client \
  socat \
  snmp \
  python3 \
  smbclient \
  python3-lxml \
  gnutls-bin \
  xml-twig-tools \
  xmltoman

# Downloading the gvmd sources

curl -f -L \
https://github.com/greenbone/gvmd/archive/refs/tags/v$GVMD_VERSION.tar.gz \
-o $SOURCE_DIR/gvmd-$GVMD_VERSION.tar.gz

curl -f -L \
https://github.com/greenbone/gvmd/re
   leases/download/v$GVMD_VERSION/gvmd-$GVMD_VERSION.tar.gz.asc \
-o $SOURCE_DIR/gvmd-$GVMD_VERSION.tar.gz.asc

#Verifying the source file
gpg --verify $SOURCE_DIR/gvmd-$GVMD_VERSION.tar.gz.asc \
$SOURCE_DIR/gvmd-$GVMD_VERSION.tar.gz

# Output should be similar to
gpg: Signature made Tue 12 Oct 2021 06:59:23 AM UTC
gpg:                using RSA key
                    8AE4BE429B60A59B311C2E739823FAA60ED1E580
gpg: Good signature from "Greenbone Community Feed integrity key"
     [ultimate]

# Extracting the tarball
```

```
tar -C $SOURCE_DIR -xvzf $SOURCE_DIR/gvmd-$GVMD_VERSION.tar.gz

# Building gvmd
mkdir -p $BUILD_DIR/gvmd
cd $BUILD_DIR/gvmd
cmake $SOURCE_DIR/gvmd-$GVMD_VERSION \
  -DCMAKE_INSTALL_PREFIX=$INSTALL_PREFIX \
  -DCMAKE_BUILD_TYPE=Release \
  -DLOCALSTATEDIR=/var \
  -DSYSCONFDIR=/etc \
  -DGVM_DATA_DIR=/var \
  -DGVM_RUN_DIR=/run/gvm \
  -DOPENVAS_DEFAULT_SOCKET=/run/ospd/ospd-openvas.sock \
  -DGVM_FEED_LOCK_PATH=/var/lib/gvm/feed-update.lock \
  -DSYSTEMD_SERVICE_DIR=/lib/systemd/system \
  -DDEFAULT_CONFIG_DIR=/etc/default \
  -DLOGROTATE_DIR=/etc/logrotate.d
```

L'ultimo comando può uscire con il seguente errore

```
export OPENVAS_SCANNER_VERSION=$GVM_VERSION
-- Looking for PostgreSQL...
CMake Error at /usr/share/cmake-3.18/Modules
/FindPackageHandleStandardArgs.cmake:165 (message):
  Could NOT find PostgreSQL (missing: PostgreSQL_TYPE_INCLUDE_DIR)
  (found version "13.5")
Call Stack (most recent call first):
  /usr/share/cmake-3.18/Modules/FindPackageHandleStandardArgs.cmake:458
  (_FPHSA_FAILURE_MESSAGE)
  /usr/share/cmake-3.18/Modules/FindPostgreSQL.cmake:247
  (find_package_handle_standard_args)
  src/CMakeLists.txt:43 (find_package)

-- Configuring incomplete, errors occurred!
```

dovuto al fatto che non viene riconosciuta la versione 13.5 di PostgreSQL. Modificare le seguenti righe del file `/usr/share/cmake-3.18/Modules/FindPostgreSQL.cmake` da

```
set(PostgreSQL_KNOWN_VERSIONS ${PostgreSQL_ADDITIONAL_VERSIONS}
    "12" "11" "10" "9.6" "9.5" "9.4" "9.3" "9.2" "9.1" "9.0" "8.4" "8.3"
    "8.2" "8.1" "8.0")
```

a

```
set(PostgreSQL_KNOWN_VERSIONS ${PostgreSQL_ADDITIONAL_VERSIONS}
    "13.5" "13" "12" "11" "10" "9.6" "9.5" "9.4" "9.3" "9.2" "9.1" "9.0"
    "8.4" "8.3" "8.2" "8.1" "8.0")
```

Dopo la modifica eseguire nuovamente `cmake`

```
# Building gvmd
cd $BUILD_DIR/gvmd
cmake $SOURCE_DIR/gvmd-$GVMD_VERSION \
  -DCMAKE_INSTALL_PREFIX=$INSTALL_gpg \
  --verify $SOURCE_DIR/gsa-$GSA_VERSION.tar.gz.asc \
  $SOURCE_DIR/gsa-$GSA_VERSION.tar.gzPREFIX \
  -DCMAKE_BUILD_TYPE=Release \
  -DLOCALSTATEDIR=/var \
  -DSYSCONFDIR=/etc \
  -DGVM_DATA_DIR=/var \
  -DGVM_RUN_DIR=/run/gvm \
  -DOPENVAS_DEFAULT_SOCKET=/run/ospd/ospd-openvas.sock \
  -DGVM_FEED_LOCK_PATH=/var/lib/gvm/feed-update.lock \
  -DSYSTEMD_SERVICE_DIR=/lib/systemd/system \
  -DDEFAULT_CONFIG_DIR=/etc/default \
  -DLOGROTATE_DIR=/etc/logrotate.d
```

e completare l'installazione con i comandi seguenti.

```
# ... building gvmd
make -j$(nproc)

#Installing gvmd
make DESTDIR=$INSTALL_DIR install
sudo cp -rv $INSTALL_DIR/* /
rm -rf $INSTALL_DIR/*
```

- **gsa**

```
# Setting the GSA version to use
export GSA_VERSION=$GVM_VERSION

# Required dependencies for gsad
sudo apt install -y \
```

```
  libmicrohttpd-dev \
  libxml2-dev \
  libglib2.0-dev \
  libgnutls28-dev

# Required dependencies for GSA
sudo apt install -y \
  nodejs \
  yarnpkg

# Downloading the gsa sources
curl -f -L \
 https://github.com/greenbone/gsa/archive/refs/tags/v$GSA_VERSION.tar.gz \
 -o $SOURCE_DIR/gsa-$GSA_VERSION.tar.gz
curl -f -L \
 https://github.com/greenbone/gsa/releases/download/v$GSA_VERSION/
 gsa-$GSA_VERSION.tar.gz.asc \
 -o $SOURCE_DIR/gsa-$GSA_VERSION.tar.gz.asc

# Verifying the source files
gpg --verify $SOURCE_DIR/gsa-$GSA_VERSION.tar.gz.asc \
 $SOURCE_DIR/gsa-$GSA_VERSION.tar.gz

# Output should be similar to
# gpg: Signature made Tue 12 Oct 2021 02:27:25 PM UTC
# gpg:                using RSA key
                      8AE4BE429B60A59B311C2E739823FAA60ED1E580
# gpg: Good signature from "Greenbone Community Feed integrity key"
       [ultimate]

# Extracting the tarball
tar -C $SOURCE_DIR -xvzf $SOURCE_DIR/gsa-$GSA_VERSION.tar.gz

# Building gsa
mkdir -p $BUILD_DIR/gsa
cd $BUILD_DIR/gsa
cmake $SOURCE_DIR/gsa-$GSA_VERSION \
  -DCMAKE_INSTALL_PREFIX=$INSTALL_PREFIX \
  -DCMAKE_BUILD_TYPE=Release \
  -DSYSCONFDIR=/etc \
```

```
  -DLOCALSTATEDIR=/var \
  -DGVM_RUN_DIR=/run/gvm \
  -DGSAD_PID_DIR=/run/gvm \
  -DLOGROTATE_DIR=/etc/logrotate.d
make -j$(nproc)

# Installing gsa
make DESTDIR=$INSTALL_DIR install
sudo cp -rv $INSTALL_DIR/* /
rm -rf $INSTALL_DIR/*
```

- **openvas-smb**

```
# Setting the openvas-smb version to use
export OPENVAS_SMB_VERSION=21.4.0

# Required dependencies for openvas-smb
sudo apt install -y \
  gcc-mingw-w64 \
  libgnutls28-dev \
  libglib2.0-dev \
  libpopt-dev \
  libunistring-dev \
  heimdal-dev \
  perl-base

# Downloading the openvas-smb sources
curl -f -L \
 https://github.com/greenbone/openvas-smb/archive/refs/tags/
 v$OPENVAS_SMB_VERSION.tar.gz \
 -o $SOURCE_DIR/openvas-smb-$OPENVAS_SMB_VERSION.tar.gz
curl -f -L \
 https://github.com/greenbone/openvas-smb/releases/download/
 v$OPENVAS_SMB_VERSION/openvas-smb-$OPENVAS_SMB_VERSION.tar.gz.asc \
 -o $SOURCE_DIR/openvas-smb-$OPENVAS_SMB_VERSION.tar.gz.asc

# Verifying the source file
gpg --verify $SOURCE_DIR/openvas-smb-$OPENVAS_SMB_VERSION.tar.gz.asc \
 $SOURCE_DIR/openvas-smb-$OPENVAS_SMB_VERSION.tar.gz

# Output should be similar to
```

```
# gpg: Signature made Fri 25 Jun 2021 06:36:43 AM UTC
# gpg:                using RSA key
                      8AE4BE429B60A59B311C2E739823FAA60ED1E580
# gpg: Good signature from "Greenbone Community Feed integrity key"
       [ultimate]

# Extracting the tarball
tar -C $SOURCE_DIR -xvzf \
 $SOURCE_DIR/openvas-smb-$OPENVAS_SMB_VERSION.tar.gz

# Building openvas-smb
mkdir -p $BUILD_DIR/openvas-smb
cd $BUILD_DIR/openvas-smb
cmake $SOURCE_DIR/openvas-smb-$OPENVAS_SMB_VERSION \
  -DCMAKE_INSTALL_PREFIX=$INSTALL_PREFIX \
  -DCMAKE_BUILD_TYPE=Release
make -j$(nproc)

# Installing openvas-smb
make DESTDIR=$INSTALL_DIR install
sudo cp -rv $INSTALL_DIR/* /
rm -rf $INSTALL_DIR/*
```

- **openvas-scanner**

```
# Setting the openvas-scanner version to use
export OPENVAS_SCANNER_VERSION=$GVM_VERSION

# Required dependencies for openvas-scanner
sudo apt install -y \
  bison \
  libglib2.0-dev \
  libgnutls28-dev \
  libgcrypt20-dev \
  libpcap-dev \
  libgpgme-dev \
  libksba-dev \
  rsync \
  nmap \
  python3-impacket \
  libsnmp-dev
```

```
# Downloading the openvas-scanner sources
curl -f -L \
 https://github.com/greenbone/openvas-scanner/archive/refs/tags/
 v$OPENVAS_SCANNER_VERSION.tar.gz -o $SOURCE_DIR/openvas-scanner-
 $OPENVAS_SCANNER_VERSION.tar.gz
curl -f -L \
 https://github.com/greenbone/openvas-scanner/releases/download/
 v$OPENVAS_SCANNER_VERSION/openvas-scanner-
 $OPENVAS_SCANNER_VERSION.tar.gz.asc \
 -o $SOURCE_DIR/openvas-scanner-
 $OPENVAS_SCANNER_VERSION.tar.gz.asc

# Verifying the source file
gpg --verify \
 $SOURCE_DIR/openvas-scanner-$OPENVAS_SCANNER_VERSION.tar.gz.asc \
 $SOURCE_DIR/openvas-scanner-$OPENVAS_SCANNER_VERSION.tar.gz

# Output should be similar to
# gpg: Signature made Mon 11 Oct 2021 03:37:56 PM UTC
# gpg:                using RSA key
                      8AE4BE429B60A59B311C2E739823FAA60ED1E580
# gpg: Good signature from "Greenbone Community Feed integrity key"
                      [ultimate]

# Extracting the tarball
tar -C $SOURCE_DIR -xvzf \
 $SOURCE_DIR/openvas-scanner-$OPENVAS_SCANNER_VERSION.tar.gz

# Building openvas-scanner
mkdir -p $BUILD_DIR/openvas-scanner
cd $BUILD_DIR/openvas-scanner
cmake $SOURCE_DIR/openvas-scanner-$OPENVAS_SCANNER_VERSION \
  -DCMAKE_INSTALL_PREFIX=$INSTALL_PREFIX \
  -DCMAKE_BUILD_TYPE=Release \
  -DSYSCONFDIR=/etc \
  -DLOCALSTATEDIR=/var \
  -DOPENVAS_FEED_LOCK_PATH=/var/lib/openvas/feed-update.lock \
  -DOPENVAS_RUN_DIR=/run/ospd
make -j$(nproc)
```

```
# Installing openvas-scanner
make DESTDIR=$INSTALL_DIR install
sudo cp -rv $INSTALL_DIR/* /
rm -rf $INSTALL_DIR/*
```

- **ospd-openvas**

```
#Setting the ospd and ospd-openvas versions to use
export OSPD_VERSION=21.4.4
export OSPD_OPENVAS_VERSION=$GVM_VERSION

#Required dependencies for ospd-openvas
sudo apt install -y \
  python3 \
  python3-pip \
  python3-setuptools \
  python3-packaging \
  python3-wrapt \
  python3-cffi \
  python3-psutil \
  python3-lxml \
  python3-defusedxml \
  python3-paramiko \
  python3-redis

# Downloading the ospd sources
curl -f -L \
 https://github.com/greenbone/ospd/archive/refs/tags/
 v$OSPD_VERSION.tar.gz \
 -o $SOURCE_DIR/ospd-$OSPD_VERSION.tar.gz
curl -f -L \
 https://github.com/greenbone/ospd/releases/download/v$OSPD_VERSION/
 ospd-$OSPD_VERSION.tar.gz.asc \
 -o $SOURCE_DIR/ospd-$OSPD_VERSION.tar.gz.asc

# Downloading the ospd-openvas sources
curl -f -L \
 https://github.com/greenbone/ospd-openvas/archive/refs/tags/
 v$OSPD_OPENVAS_VERSION.tar.gz \
 -o $SOURCE_DIR/ospd-openvas-$OSPD_OPENVAS_VERSION.tar.gz
```

```
curl -f -L \
 https://github.com/greenbone/ospd-openvas/releases/download/
 v$OSPD_OPENVAS_VERSION/ospd-openvas-$OSPD_OPENVAS_VERSION.tar.gz.asc \
 -o $SOURCE_DIR/ospd-openvas-$OSPD_OPENVAS_VERSION.tar.gz.asc

# Verifying the source files
gpg --verify $SOURCE_DIR/ospd-$OSPD_VERSION.tar.gz.asc \
 $SOURCE_DIR/ospd-$OSPD_VERSION.tar.gz
gpg --verify $SOURCE_DIR/ospd-openvas-$OSPD_OPENVAS_VERSION.tar.gz.asc \
 $SOURCE_DIR/ospd-openvas-$OSPD_OPENVAS_VERSION.tar.gz

# Output should be similar to
# gpg: Signature made Tue 12 Oct 2021 12:05:33 PM UTC
# gpg:                using RSA key
                      8AE4BE429B60A59B311C2E739823FAA60ED1E580
# gpg: Good signature from "Greenbone Community Feed integrity key"
       [ultimate]

# Extracting the tarball
tar -C $SOURCE_DIR -xvzf $SOURCE_DIR/ospd-$OSPD_VERSION.tar.gz
tar -C $SOURCE_DIR -xvzf \
 $SOURCE_DIR/ospd-openvas-$OSPD_OPENVAS_VERSION.tar.gz

# Installing ospd
cd $SOURCE_DIR/ospd-$OSPD_VERSION
python3 -m pip install . --prefix=$INSTALL_PREFIX --root=$INSTALL_DIR

# Installing ospd-openvas
cd $SOURCE_DIR/ospd-openvas-$OSPD_OPENVAS_VERSION
python3 -m pip install . --prefix=$INSTALL_PREFIX --root=$INSTALL_DIR \
 --no-warn-script-location

# Finishing installation
sudo cp -rv $INSTALL_DIR/* /
rm -rf $INSTALL_DIR/*
```

- **gvm-tools**

```
# Required dependencies for gvm-tools
sudo apt install -y \
  python3 \
```

```
    python3-pip \
    python3-setuptools \
    python3-packaging \
    python3-lxml \
    python3-defusedxml \
    python3-paramiko

# Installing gvm-tools for the current user
# python3 -m pip install --user gvm-tools

# Installing gvm-tools system-wide
python3 -m pip install --prefix=$INSTALL_PREFIX --root=$INSTALL_DIR \
 --no-warn-script-location gvm-tools
sudo cp -rv $INSTALL_DIR/* /
rm -rf $INSTALL_DIR/*
```

7.2 Configurazione dell'utente per poter utilizzare GVM

Aggiornare la PATH dell'utente per poter eseguire sia i comandi di amministrazione che di gestione di GVM, per esempio, aggiungendo al file `.bashrc`

```
export PATH=$PATH:/usr/local/sbin:/sbin
```

7.3 Configurazione di GVM

7.3.1 Installazione e configurazione di Redis Data Store

```
# Installing the Redis server
sudo apt install -y redis-server

# Adding configuration for running the Redis server for the scanner
sudo cp $SOURCE_DIR/openvas-scanner-$GVM_VERSION/config/redis-openvas.conf \
 /etc/redis/
sudo chown redis:redis /etc/redis/redis-openvas.conf
echo "db_address = /run/redis-openvas/redis.sock" | sudo tee -a \
 /etc/openvas/openvas.conf

## start redis with openvas config
sudo systemctl start redis-server@openvas.service

## ensure redis with openvas config is started on every system startup
sudo systemctl enable redis-server@openvas.service
```

```
#Adding the gvm user to the redis group
sudo usermod -aG redis gvm
```

7.3.2 Impostazione dei permessi

```
# Creating /run/gvm
sudo mkdir /run/gvm

# Adjusting directory permissions
sudo chown -R gvm:gvm /var/lib/gvm
sudo chown -R gvm:gvm /var/lib/openvas
sudo chown -R gvm:gvm /var/log/gvm
sudo chown -R gvm:gvm /run/gvm
sudo chmod -R g+srw /var/lib/gvm
sudo chmod -R g+srw /var/lib/openvas
sudo chmod -R g+srw /var/log/gvm

# Adjusting gvmd permissions
sudo chown gvm:gvm /usr/local/sbin/gvmd
sudo chmod 6750 /usr/local/sbin/gvmd

# Adjusting feed sync script permissions
sudo chown gvm:gvm /usr/local/bin/greenbone-nvt-sync
sudo chmod 740 /usr/local/sbin/greenbone-feed-sync
sudo chown gvm:gvm /usr/local/
Open Vulnerability and Assessment Languagesbin/greenbone-*-sync
sudo chmod 740 /usr/local/sbin/greenbone-*-sync
```

7.3.3 Impostazione di sudo per le scansioni

Tramite il comando

```
sudo visudo
```

aggiungere le seguenti righe al file di configurazione di sudo

```
# allow users of the gvm group run openvas
%gvm ALL = NOPASSWD: /usr/local/sbin/openvas
```

7.3.4 Installazione e configurazione di PostgreSQL

```
# Installing the PostgreSQL server
sudo apt install -y postgresql
```

```
# Starting the PostgreSQL database server
sudo systemctl start postgresql@13-main

# Setting up PostgreSQL user and database
sudo -u postgres bash
createuser -DRS gvm
createdb -O gvm gvmd
exit

# Setting up database permissions and extensions
sudo -u postgres bash
psql gvmd
create role dba with superuser noinherit;
grant dba to gvm;

create extension "uuid-ossp";
create extension "pgcrypto";
exit
exit
```

7.3.5 Definizione dell'utente amministratore di GVM

```
#Creating an administrator user with provided password
gvmd --create-user=admin --password=<password>
```

7.3.6 Definizione dell'utente per l'importazione dei feed

```
gvmd --modify-setting 78eceaec-3385-11ea-b237-28d24461215b \
 --value `gvmd --get-users --verbose | grep admin | awk '{print $2}'`
```

7.3.7 Download dei feed

```
#Syncing VTs processed by the scanner
sudo -u gvm greenbone-nvt-sync

#Syncing the data processed by gvmd
## scap
sudo -u gvm greenbone-feed-sync --type SCAP
## cert
sudo -u gvm greenbone-feed-sync --type CERT
##gvm_data
```

```
sudo -u gvm greenbone-feed-sync --type GVMD_DATA
```

7.3.8 Configurazione dei servizi con `systemd`

```
# Systemd service file for ospd-openvas
cat << EOF > $BUILD_DIR/ospd-openvas.service
[Unit]
Description=OSPd Wrapper for the OpenVAS Scanner (ospd-openvas)
Documentation=man:ospd-openvas(8) man:openvas(8)
After=network.target networking.service redis-server@openvas.service
Wants=redis-server@openvas.service
ConditionKernelCommandLine=!recovery

[Service]
Type=forking
User=gvm
Group=gvm
RuntimeDirectory=ospd
RuntimeDirectoryMode=2775
PIDFile=/run/ospd/ospd-openvas.pid
ExecStart=/usr/local/bin/ospd-openvas --unix-socket \
 /run/ospd/ospd-openvas.sock --pid-file /run/ospd/ospd-openvas.pid \
 --log-file /var/log/gvm/ospd-openvas.log --lock-file-dir /var/lib/openvas \
 --socket-mode 0o770
SuccessExitStatus=SIGKILL
Restart=always
RestartSec=60

[Install]
WantedBy=multi-user.target
EOF

sudo cp $BUILD_DIR/ospd-openvas.service /etc/systemd/system/

# Systemd service file for gvmd
cat << EOF > $BUILD_DIR/gvmd.service
[Unit]
Description=Greenbone Vulnerability Manager daemon (gvmd)
After=network.target networking.service postgresql.service \
 ospd-openvas.service
Wants=postgresql.service ospd-openvas.service
```

```
Documentation=man:gvmd(8)
ConditionKernelCommandLine=!recovery

[Service]
Type=forking
User=gvm
Group=gvm
PIDFile=/run/gvm/gvmd.pid
RuntimeDirectory=gvm
RuntimeDirectoryMode=2775
ExecStart=/usr/local/sbin/gvmd --osp-vt-update=/run/ospd/ospd-openvas.sock \
 --listen-group=gvm
Restart=always
TimeoutStopSec=10

[Install]
WantedBy=multi-user.target
EOF

sudo cp $BUILD_DIR/gvmd.service /etc/systemd/system/

# Systemd service file for gsad
cat << EOF > $BUILD_DIR/gsad.service
[Unit]
Description=Greenbone Security Assistant daemon (gsad)
Documentation=man:gsad(8) https://www.greenbone.net
After=network.target gvmd.service
Wants=gvmd.service

[Service]
Type=forking
User=gvm
Group=gvm
PIDFile=/run/gvm/gsad.pid
ExecStart=/usr/local/sbin/gsad --listen=0.0.0.0 --port=9392 --http-only
Restart=always
TimeoutStopSec=10

[Install]
WantedBy=multi-user.target
```

```
Alias=greenbone-security-assistant.service
EOF

sudo cp $BUILD_DIR/gsad.service /etc/systemd/system/

# Making systemd aware of the new service files
sudo systemctl daemon-reload

# Ensuring services are run at every system startup
sudo systemctl enable ospd-openvas
sudo systemctl enable gvmd
sudo systemctl enable gsad
```

7.3.9 Prima verifica di funzionamento dell'interfaccia web

Per verificare il funzionamento dell'interfaccia web è possibile collegarsi (per il momento usando il protocollo HTTP, a breve utilizzando HTTPS) tramite un browser web alla porta 9392

```
http://indirizzo_del_server:9392
```

7.3.10 Creazione di certificati self-signed e passaggio ad HTTPS

- Verifica dei certificati presenti tramite il comando

  ```
  sudo -u gvm gvm-manage-certs -V
  ```

  Che dovrebbe rendere un output simile al seguente, non essendo ancora stati generati i certificati.

  ```
  ERROR: Directory for keys (/var/lib/gvm/private/CA) not found!
  ERROR: Directory for certificates (/var/lib/gvm/CA) not found!
  ERROR: CA key not found in /var/lib/gvm/private/CA/cakey.pem
  ERROR: CA certificate not found in /var/lib/gvm/CA/cacert.pem
  ERROR: CA certificate failed verification,
         see /tmp/tmp.xKlj9933jr/gvm-manage-certs.log for details.
         Aborting.
  find: ‘/var/lib/gvm/CA’: No such file or directory

  ERROR: Your GVM certificate infrastructure did NOT pass validation.
         See messages above for details.
  ```

- Crezione dei certificati col comando

```
sudo -u gvm gvm-manage-certs -a
```

- Verificando nuovamenti i certificati presenti col comando

  ```
  sudo -u gvm gvm-manage-certs -V
  ```

  si ottiene qualcosa di simile al seguente output

  ```
  OK: Directory for keys (/var/lib/gvm/private/CA) exists.
  OK: Directory for certificates (/var/lib/gvm/CA) exists.
  OK: CA key found in /var/lib/gvm/private/CA/cakey.pem
  OK: CA certificate found in /var/lib/gvm/CA/cacert.pem
  OK: CA certificate verified.
  OK: Certificate /var/lib/gvm/CA/servercert.pem verified.
  OK: Certificate /var/lib/gvm/CA/clientcert.pem verified.

  OK: Your GVM certificate infrastructure passed validation.
  ```

  I file che contengono il certificato del server e la chiave privata sono i seguenti:

  - certificato del server:
    `/var/lib/gvm/CA/servercert.pem`
  - chiave privata del certificato del server:
    `/var/lib/gvm/private/CA/serverkey.pem`

- Per poter utilizzare il protocollo HTTPS per l'interfaccia web modificare le seguenti righe del file `/etc/systemd/system/gsad.service` da

  ```
  [Service]
  Type=forking
  User=gvm
  Group=gvm
  PIDFile=/run/gvm/gsad.pid
  ExecStart=/usr/local/sbin/gsad --listen=0.0.0.0 --port=9392 --http-only
  ```

  a

  ```
  [Service]
  Type=forking
  #User=gvm
  #Group=gvm
  PIDFile=/run/gvm/gsad.pid
  ```

```
ExecStart=/usr/local/sbin/gsad --listen=0.0.0.0 --port=443 \
 -c /var/lib/gvm/CA/servercert.pem \
 -k /var/lib/gvm/private/CA/serverkey.pem \
 --drop-privileges=gvm
```

e riavviare il servizio con i segueti comandi

```
sudo systemctl daemon-reload
sudo systemctl restart gsad.service
```

- Per verificare il funzionamento dell'interfaccia web col protocollo HTTPS, collegarsi tramite un browser web alla seguente URL:

```
https://indirizzo_del_server
```

7.4 SCAP Feed: CVE, CPE ed OVAL

7.4.1 CVE

Un elenco CVE (Common Vulnerabilities and Exposures) è un elenco di falle nella sicurezza informatica divulgato al pubblico. Quando si fa riferimento a un CVE, si intende in genere una falla di sicurezza a cui è stato assegnato un numero identificativo (ID) CVE.

Gli avvisi di sicurezza emessi da fornitori e ricercatori menzionano quasi sempre almeno un ID CVE. I CVE aiutano i professionisti IT a coordinare le iniziative per assegnare le priorità e risolvere le vulnerabilità, con l'obiettivo di rendere i sistemi informatici più sicuri.

Il sistema è supervisionato da MITRE Corporation e supportato dall'Agenzia per la sicurezza informatica e delle infrastrutture che fa capo al Dipartimento della sicurezza interna degli Stati Uniti.

Le voci incluse nell'elenco CVE sono concise e non contengono dati tecnici o informazioni sui rischi, sulla loro portata o sulle possibili correzioni. Queste informazioni sono contenute in altri database, ad esempio l'U.S. National Vulnerability Database (NVD), il CERT/CC Vulnerability Notes Database e altri elenchi gestiti e aggiornati da fornitori e altre organizzazioni. Data la presenza di numerosi sistemi di informazione, gli ID CVE costituiscono una modalità attendibile per distinguere una falla di sicurezza specifica dalle altre.

7.4.2 CPE

Il dizionario CPE (Common Platform Enumeration) è uno schema di denominazione strutturato per sistemi informatici, software e pacchetti. Basato sulla sintassi generica per gli Uniform Resource Identifiers (URI), CPE include un formato di nome formale, un

metodo per controllare i nomi rispetto a un sistema, e un formato di descrizione per legare testo e test a un nome. Il dizionario CPE viene aggiornato ogni notte quando vengono aggiunte modifiche o nuovi nomi.

A partire da dicembre 2009, il National Vulnerability Database accetta contributi al Dizionario Ufficiale CPE.

Il Dizionario CPE ospitato e mantenuto al NIST può essere utilizzato da organizzazioni non governative su base volontaria e non è soggetto a copyright negli Stati Uniti.

7.4.3 OVAL

OVAL (Open Vulnerability and Assessment Language) rappresenta un tentativo delle comunità internazionali che si occupano di sicurezza per standardizzare il modo di valutare e segnalare lo stato di sicurezza delle macchine dei sistemi informatici. OVAL include un linguaggio per codificare i dettagli del sistema e un assortimento di repository di contenuti tenuti in tutta la comunità.

Gli strumenti e i servizi che usano OVAL forniscono informazioni accurate, coerenti e concretamente utilizzabili da poter essere utilizzate nelle fasi di vulnerability assestment.

L'uso di OVAL fornisce anche metriche affidabili e riproducibili per la sicurezza delle informazioni e permette l'interoperabilità e l'automazione tra numerosi strumenti e servizi di sicurezza.

Riferimenti bibliografici

[1] Agenzia per l'Italia Digitale *circolare 18 aprile 2017, n. 2/2017* in sostituzione della *circolare n. 1/2017 del 17 marzo 2017, recante: Misure minime di sicurezza ICT per le pubbliche amministrazioni. (Direttiva del Presidente del Consiglio dei ministri 1º agosto 2015) (17A03060), GU n.103 del 5-5-2017*, Vigente al 5-5-2017.

[2] *Implementazione delle misure minime di sicurezza nell'INFN*, 20 dicembre 2017.

[3] *Disciplinare per l'uso delle risorse informatiche INFN* del 24/01/2020.

[4] Delibera n. 15442 del Consiglio Direttivo dell'INFN del 28/02/2020 *Approvazione modifiche al Disciplinare per l'uso delle risorse informatiche dell'INFN*.

[5] Leandro Lanzi *KALINFN - Una distribuzione live per scansioni*, *Corso di formazione sulla Sicurezza Informatica* previsto dal programma formativo della Commissione Nazionale per la Formazione INFN, INFN Milano, 13-14 novembre 2017. `https://agenda.infn.it/event/14102/contributions/23006/`

[6] `https://www.greenbone.net/en/product-comparison`

[7] https://greenbone.github.io/docs/_images/gvm-architecture.png

[8] https://www.greenbone.net/wp-content/uploads/Solution_Comparison_EN.pdf

[9] Leandro Lanzi *GVM+SCANBOT*, Workshop CCR 24-28/05/2021.
https://agenda.infn.it/event/25889/contributions/135733/

[10] https://www.tenable.com/products/nessus

[11] https://www.metasploit.com/

[12] https://www.intruder.io/

[13] https://github.com/sullo/nikto

[14] https://www.netsparker.com/

[15] https://www.acunetix.com/

[16] https://threatspotter.com/