OpenVAS Technical Segment

Paul Asadoorian

Security Weekly

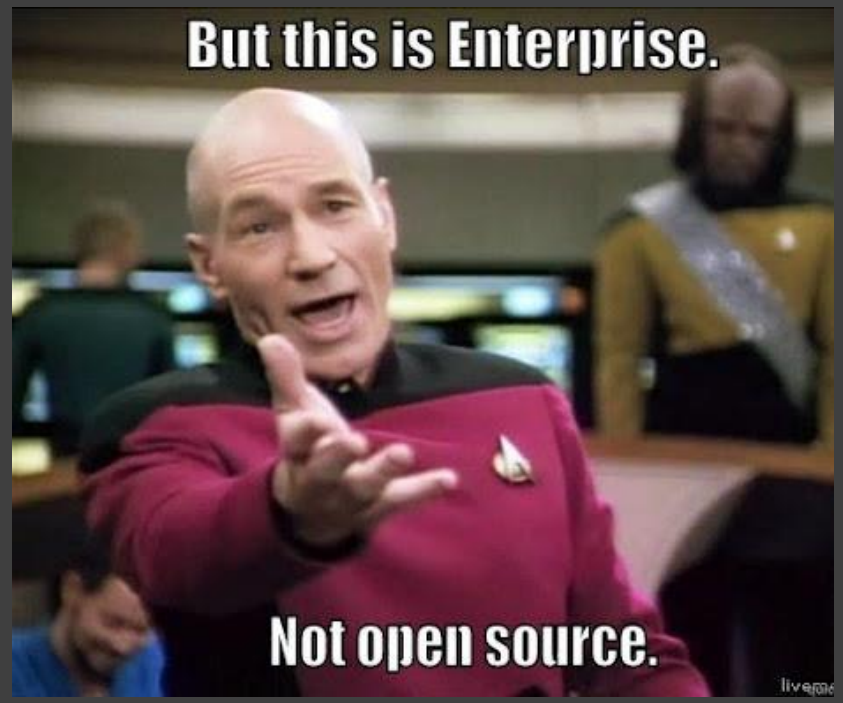
What Is The Problem You Are Trying To Solve?

- You need a vulnerability scanner, but you are very short on budget but flush with time

- You want to build a vulnerability scanner into your environment and not pay license fees from a commercial company

- You desire ultimate control and flexibility
  - By compiling it from source, you also gain an understanding of how it works in order to troubleshoot it more easily

- You need to setup a vulnerability scanner to make a point at your job that:
  - You have vulnerable systems
  - A vulnerability scanner can help identify vulnerabilities in those systems
  - Your patch management program is inaccurate or teams are over confident



History

- In 2005, Tenable decided that Nessus version 3 would be closed-source and that Nessus version 2 would remain under GPL
  - FYI, I joined Tenable in March 2009
- Several forks appeared shortly thereafter, most notable was OpenVAS (Open Vulnerability Assessment System)
- Greenbone Networks, a German-based company, spearheaded the effort to continue developing OpenVAS
- Greenbone Security Manager (GSM) appeared in 2010, furthering the company's efforts to turn an open-source vulnerability scanner into a vulnerabilty management platform

OpenVAS: Many Different Components Today

gvm-libs

release v21.4.2 | Doc Coverage 87% | PASSED

This is the libraries module for the Greenbone Vulnerability Management Solution.

It is used for the Greenbone Security Manager appliances and provides various functionalities to support the integrated service daemons.

OpenVAS

release v21.4.2 | Doc Coverage unknown | Build and Test passing

This is the Open Vulnerability Assessment Scanner (OpenVAS) of the Greenbone Vulnerability Management (GVM) Solution.

It is used for the Greenbone Security Manager appliances and is a full-featured scan engine that executes a continuously updated and extended feed of Network Vulnerability Tests (NVTs).

openvas-smb

release v21.4.0 | PASSED

This is the `smb` module for the OpenVAS Scanner. It includes libraries (`openvas-wmiclient` / `openvas-wincmd`) to interface with Microsoft Windows Systems through the Windows Management Instrumentation API and a `winexe` binary to execute processes remotely on that system.

OpenVAS: More Components

ospd

release v21.4.3 | pypi v21.4.3 | Scrutinizer 6.13 | codecov 75% | PASSED

ospd is a base class for scanner wrappers which share the same communication protocol: OSP (Open Scanner Protocol). OSP creates a unified interface for different security scanners and makes their control flow and scan results consistently available under the central Greenbone Vulnerability Manager service.

OSP is similar in many ways to GMP (Greenbone Management Protocol): XML-based, stateless and non-permanent connection.

The design supports wrapping arbitrary scanners with same protocol OSP, sharing the core daemon options while adding scanner specific parameters and options.

Greenbone Vulnerability Manager

release v21.4.3 | Doc Coverage 91% | PASSED

The Greenbone Vulnerability Manager is the central management service between security scanners and the user clients.

It manages the storage of any vulnerability management configurations and of the scan results. Access to data, control commands and workflows is offered via the XML-based Greenbone Management Protocol (GMP). Controlling scanners like *OpenVAS* is done via the Open Scanner Protocol (OSP).

ospd-openvas

release v21.4.2 | pypi v21.4.2 | codecov 75% | Build and test Python package passing

This is an OSP server implementation to allow GVM to remotely control OpenVAS, see https://github.com/greenbone/openvas.

Once running, you need to configure OpenVAS for the Greenbone Vulnerability Manager, for example via the web interface Greenbone Security Assistant. Then you can create scan tasks to use OpenVAS.

Greenbone Security Assistant

release v20.8.3 | codecov 59% | Build and test C passing | Build and test JavaScript passing

The Greenbone Security Assistant is the web interface developed for the Greenbone Security Manager appliances.

It connects to the Greenbone Vulnerability Manager **GVM** to provide a full-featured user interface for vulnerability management.

Greenbone Security Assistant consists of

- GSA - The webpage written in React

and

- GSAD - The HTTP server talking to the GVM daemon

Three Ways To Get It

- You can download a pre-configured VM from Greenbone (registration required)

- There are several open-source projects that have made available Docker container builds. Only one that I really love, and referenced for this segment:
  - https://github.com/immauss/openvas/

- You can compile all the components from source (This is tedious and time-consuming, but a great learning experience!)
  - Also, much less time consuming as I will show you how (based on this outstanding guide: https://makyotox.medium.com/install-openvas-greenbone-20-08-on-ubuntu-20-04-8082c7eab67c
  - Even with the above guide, there are several tweaks as the project changed since that was published

The Setup

- VirtualBox (I used the packages in the Ubuntu Repo, this also works with Detection Lab)
  - **FYI, I rarely do this, favoring installing from source or official repos (for Docker, Golang, Ruby, NodeJS, etc...)**
- Ubuntu Server 20.04
- Install VirtualBox Tools: https://linuxize.com/post/how-to-install-virtualbox-guest-additions-in-ubuntu/
- I gave it 2 cores, 30GB of disk, and 16GB of RAM (You could probably get away with less, but more is actually more in this case)
  - **FYI, 10GB of disk is NOT enough**

Part 1 – Install Prerequisites

```
# curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | apt-key add -
# echo "deb https://dl.yarnpkg.com/debian/ stable main" | tee /etc/apt/sources.list.d/yarn.list
# apt-get update
# apt-get install \
bison \
clang-format \
cmake \
curl \
doxygen \
flex \
g++ \
gcc \
gcc-mingw-w64 \
gettext \
git \
gnutls-bin \
heimdal-dev \
heimdal-dev \
libgcrypt20-dev \
libglib2.0-dev \
libgnutls28-dev \
libgpgme-dev \
libhiredis-dev \
libical-dev \
libksba-dev \
libldap2-dev \
libmicrohttpd-dev \
libnet1-dev \
```

```
libpcap-dev \
libpopt-dev \
libpq-dev \
libradcli-dev \
libsnmp-dev \
libssh-gcrypt-dev \
libunistring-dev \
libxml2-dev \
nmap \
perl-base \
pkg-config \
postgresql-server-dev-12 \
python3-defusedxml \
python3-dev \
python3-lxml \
python3-paramiko \
python3-pip \
python3-polib \
python3-setuptools \
redis \
redis-server \
texlive-fonts-recommended \
texlive-latex-extra \
uuid-dev \
xmltoman \
xml-twig-tools \
zlib1g-dev \
yarn \
postgresql \
postgresql-contrib \
postgresql-server-dev-all \
xsltproc -y
```

Part 2 – Add gvm user and install database

```
# useradd -r -d /opt/gvm -c "GVM User" -s /bin/bash gvm
# mkdir /opt/gvm
# chown gvm:gvm /opt/gvm
# The GVM and OpenVAS packages will install files in standard locations, which requires root access. This was a change made
a few months ago. I gave the gvm user full sudo access and # set a password:
# passwd gvm
# EDITOR=/usr/bin/vi visudo

# sudo -Hiu postgres
$ createuser gvm
$ createdb -O gvm gvmd
$ psql gvmd
=> create role dba with superuser noinherit;
=> grant dba to gvm;
=> create extension "uuid-ossp";
=> \q
$ exit

$ systemctl restart postgresql
$ systemctl enable postgresql
$ cp /etc/environment /etc/environment.bck
$ echo
PATH=\"/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/opt/gvm/bin:/opt/
gvm/sbin:/opt/gvm/.local/bin\" > /etc/environment

$ echo "/opt/gvm/lib" > /etc/ld.so.conf.d/gvm.conf
```

Part 3 – Clone The Repos

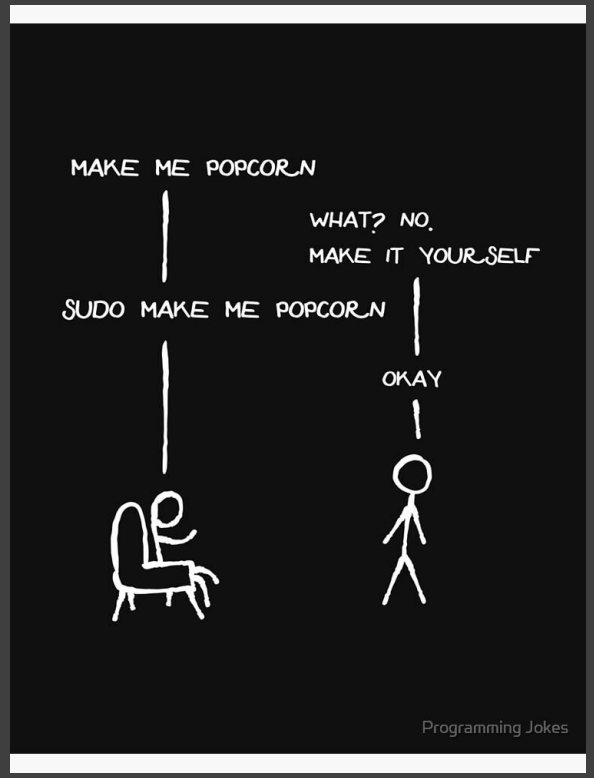
```
# Build and Install GVM 20.08 (The stable version, newer version is still in
# beta/alpha, don't attempt it, unless you really want to work for it!)
# su - gvm
$ mkdir /tmp/gvm-source
$ cd /tmp/gvm-source
$ git clone -b gvm-libs-20.08 https://github.com/greenbone/gvm-libs.git
$ git clone https://github.com/greenbone/openvas-smb.git
$ git clone -b openvas-20.08 https://github.com/greenbone/openvas.git
$ git clone -b ospd-20.08 https://github.com/greenbone/ospd.git
$ git clone -b ospd-openvas-20.08 https://github.com/greenbone/ospd-openvas.git
$ git clone -b gvmd-20.08 https://github.com/greenbone/gvmd.git
$ git clone -b gsa-20.08 https://github.com/greenbone/gsa.git
$ export PKG_CONFIG_PATH=/opt/gvm/lib/pkgconfig:$PKG_CONFIG_PATH
```

Part 4 – Build Some Of The Source

```
# For each of the directories and code you just pulled down, basically
# run these commands in each directory:

$ cd gvm-libs
$ mkdir build
$ cd build
$ cmake .. -DCMAKE_INSTALL_PREFIX=/opt/gvm
$ make
$ sudo make install

# Execute the same commands in the following directories:
$ cd ../../openvas-smb/
$ cd ../../openvas
```



Also note, the install prefix is set to /opt/gvm, this really just makes you feel good about installing everything in /opt/gvm. In fact, a recent change to the Greenbone repos shows the developers ignoring the prefix for select files. Up to you to figure out which ones. (Don't worry, I figured it out and documented it here). **This is also why you need to run "sudo make install".**

Part 5 – More Config & Setup

```
# ldconfig
# cp /tmp/gvm-source/openvas/config/redis-openvas.conf /etc/redis/
# chown redis:redis /etc/redis/redis-openvas.conf
# echo "db_address = /run/redis-openvas/redis.sock" > /opt/gvm/etc/openvas/openvas.conf
# chown gvm:gvm /opt/gvm/etc/openvas/openvas.conf
# usermod -aG redis gvm
# echo "net.core.somaxconn = 1024" >> /etc/sysctl.conf
# echo 'vm.overcommit_memory = 1' >> /etc/sysctl.conf
# sysctl -p
# cat <<EOT > /etc/systemd/system/disable_thp.service
[Unit]
Description=Disable Kernel Support for Transparent Huge Pages (THP)
[Service]
Type=simple
ExecStart=/bin/sh -c "echo 'never' > /sys/kernel/mm/transparent_hugepage/enabled && echo 'never' >
/sys/kernel/mm/transparent_hugepage/defrag"
[Install]
WantedBy=multi-user.target
EOT
# systemctl daemon-reload
# systemctl enable --now disable_thp
# systemctl enable --now redis-server@openvas
# echo "gvm ALL = NOPASSWD: /opt/gvm/sbin/openvas" > /etc/sudoers.d/gvm
# sed
's/Defaults\s.*secure_path=\"\/usr\/local\/sbin:\/usr\/local\/bin:\/usr\/sbin:\/usr\/bin:\/sbin:\/bin:\/snap\/bin\"/Defaul
ts
secure_path=\"\/usr\/local\/sbin:\/usr\/local\/bin:\/usr\/sbin:\/usr\/bin:\/sbin:\/bin:\/snap\/bin:\/opt\/gvm\/sbin\"/g'
/etc/sudoers | EDITOR='tee' visudo
# echo "gvm ALL = NOPASSWD: /opt/gvm/sbin/gsad" >> /etc/sudoers.d/gvm
```

https://community.greenbone.net/t/too-many-open-files-stack-smashing-detected/4087
https://www.mail-archive.com/openvas-discuss@wald.intevation.org/msg11729.html

Part 6 – Update Plugins (NVTs) and More Building

```
# First, change some permissions. We'll run this command again as well to get any changes:

# chown -R gvm:gvm /etc/openvas /opt/gvm /var/lib/openvas /var/log/gvm /etc/gvm /var/lib/gvm

# Update NVTs, back to gvm user
# su - gvm
$ greenbone-nvt-sync
$ sudo openvas --update-vt-info

# Build and Install Greenbone Vulnerability Manager
$ export PKG_CONFIG_PATH=/opt/gvm/lib/pkgconfig:$PKG_CONFIG_PATH
$ cd /tmp/gvm-source/gvmd
$ mkdir build
$ cd build
$ cmake .. -DCMAKE_INSTALL_PREFIX=/opt/gvm
$ make
$ sudo make install

# Build and Install Greenbone Security Assistant
$ cd ../../gsa
$ mkdir build
$ cd build
$ cmake .. -DCMAKE_INSTALL_PREFIX=/opt/gvm
$ make
$ sudo make install
```

https://www.incredibuild.com/blog/cmake-vs-make

Part 7 – More Updating & File Permission Fixes

```
# Update GVM CERT and SCAP data from the feed servers;
$ greenbone-scapdata-sync -rsync    # This takes a really long time, go get some coffee or tea or whatever
$ greenbone-certdata-sync --rsync
$ greenbone-feed-sync --type GVMD_DATA
$ gvm-manage-certs -a
# Build and Install OSPd and OSPd-OpenVAS
$ export PKG_CONFIG_PATH=/opt/gvm/lib/pkgconfig:$PKG_CONFIG_PATH
$ mkdir -p /opt/gvm/lib/python3.8/site-packages/
$ export PYTHONPATH=/opt/gvm/lib/python3.8/site-packages
$ cd /tmp/gvm-source/ospd
$ python3 setup.py install --prefix=/opt/gvm
$ cd ../ospd-openvas
$ python3 setup.py install --prefix=/opt/gvm
# More clean-up as the OpenVAS devs changed the location of files to "be more standard". You could also use softlinks.
$ mkdir -p /opt/gvm/var/log/gvm
$ mkdir -p /opt/gvm/var/lib/gvm/private/CA/
$ cp /var/lib/gvm/private/CA/serverkey.pem /opt/gvm/var/lib/gvm/private/CA/
$ sudo chown -R gvm:gvm /opt/gvm
$ mkdir /opt/gvm/var/lib/gvm/CA
$ cp /var/lib/gvm/CA/servercert.pem /opt/gvm/var/lib/gvm/CA/
$ mkdir /opt/gvm/var/lib/gvm/CA
$ cp /var/lib/gvm/CA/servercert.pem /opt/gvm/var/lib/gvm/CA/
$ sudo mkdir /run/gvm
$ sudo chown gvm:gvm /run/gvm
# Now you can start OpenVAS Scanner, GSA and GVM services
$ /usr/bin/python3 /opt/gvm/bin/ospd-openvas \
--pid-file /opt/gvm/var/run/ospd-openvas.pid \
--log-file /opt/gvm/var/log/gvm/ospd-openvas.log \
--lock-file-dir /opt/gvm/var/run -u /opt/gvm/var/run/ospd.sock
$ gvmd --osp-vt-update=/opt/gvm/var/run/ospd.sock
$ sudo gsad
$ ps aux | grep -E "ospd-openvas|gsad|gvmd" | grep -v grep   # This should list all required running processes
```

Part 8 – Create systemd Services

```
# Now, as root, create the services in systemd:
# cat <<EOT > /etc/systemd/system/openvas.service
[Unit]
Description=Control the OpenVAS service
After=redis.service
After=postgresql.service
[Service]
ExecStartPre=-rm -rf /opt/gvm/var/run/ospd-openvas.pid /opt/gvm/var/run/ospd.sock /opt/gvm/var/run/gvmd.sock
Type=simple
User=gvm
Group=gvm
Environment=PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/opt/gvm/bin:/op
t/gvm/sbin:/opt/gvm/.local/bin
Environment=PYTHONPATH=/opt/gvm/lib/python3.8/site-packages
ExecStart=/usr/bin/python3 /opt/gvm/bin/ospd-openvas \
--pid-file /opt/gvm/var/run/ospd-openvas.pid \
--log-file /opt/gvm/var/log/gvm/ospd-openvas.log \
--lock-file-dir /opt/gvm/var/run -u /opt/gvm/var/run/ospd.sock
RemainAfterExit=yes
[Install]
WantedBy=multi-user.target
EOT
# systemctl daemon-reload
# systemctl start openvas
# systemctl status openvas
# systemctl enable openvas
```

```
# Create GSA Service Unit file
# cat <<EOT > /etc/systemd/system/gsa.service
[Unit]
Description=Control the OpenVAS GSA service
After=openvas.service
[Service]
Type=simple
User=gvm
Group=gvm
Environment=PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/opt/gvm/bin:/opt/gvm/sbin:/opt/gvm/.local/bin
Environment=PYTHONPATH=/opt/gvm/lib/python3.8/site-packages
ExecStart=/usr/bin/sudo /opt/gvm/sbin/gsad --mlisten=0.0.0.0 --mport=9392
RemainAfterExit=yes
[Install]
WantedBy=multi-user.target
EOT
# cat <<EOT > /etc/systemd/system/gsa.path
[Unit]
Description=Start the OpenVAS GSA service when gvmd.sock is available
[Path]
PathChanged=/opt/gvm/var/run/gvmd.sock
Unit=gsa.service
[Install]
WantedBy=multi-user.target
EOT
# Create GVM Service unit file
# cat <<EOT > /etc/systemd/system/gvm.service
[Unit]
Description=Control the OpenVAS GVM service
After=openvas.service
[Service]
Type=simple
User=gvm
Group=gvm
Environment=PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/opt/gvm/bin:/opt/gvm/sbin:/opt/gvm/.local/bin
Environment=PYTHONPATH=/opt/gvm/lib/python3.8/site-packages
ExecStart=/opt/gvm/sbin/gvmd --osp-vt-update=/opt/gvm/var/run/ospd.sock --listen=0.0.0.0 --port=9392
RemainAfterExit=yes
[Install]
WantedBy=multi-user.target
EOT
# cat <<EOT > /etc/systemd/system/gvm.path
[Unit]
Description=Start the OpenVAS GVM service when opsd.sock is available
[Path]
PathChanged=/opt/gvm/var/run/ospd.sock
Unit=gvm.service
[Install]
WantedBy=multi-user.target
EOT
# systemctl daemon-reload
# systemctl enable --now openvas
# systemctl enable --now gvm.{path,service}
# systemctl enable --now gsa.{path,service}
```

Part 9 – Create More systemd Services..

Part 10 – Create Scanners & Accounts

```
# Create GVM Scanner
# sudo -Hiu gvm gvmd --create-scanner="SW OpenVAS Scanner" --scanner-type="OpenVAS" --scanner-host=/opt/gvm/var/run/ospd.sock
# sudo -Hiu gvm gvmd --get-scanners

# SCANNER_UUID=$(sudo -Hiu gvm gvmd --get-scanners | grep OpenVAS | cut -f1 -d" ")
# sudo -Hiu gvm gvmd --modify-scanner=$SCANNER_UUID --scanner-host=/opt/gvm/var/run/ospd.sock
# sudo -Hiu gvm gvmd --verify-scanner=$SCANNER_UUID

# Create OpenVAS (GVM) Admin User
# sudo -Hiu gvm gvmd --create-user gvmadmin --password="secret"

# If you ever want to change the password:
# sudo -Hiu gvm gvmd --user=gvmadmin --new-password="newsecret"

# Add the user as import feed owner
# USER_UUID=$(sudo -Hiu gvm gvmd --get-users --verbose | cut -f2 -d" ")
# sudo -Hiu gvm gvmd --modify-setting 78eceaec-3385-11ea-b237-28d24461215b --value $USER_UUID
```

Log files for troubleshooting:

- /opt/gvm/var/log/gvm/gsad.log
- /opt/gvm/var/log/gvm/ospd-openvas.log
- /var/log/gvm/gvmd.log
- /var/log/gvm/openvas.log

Part 11 – (Finally) Rejoice When/If It Works!