Command Line Inteface (CLI) di gvmd

Tutte le operazioni disponibili nell'interfaccia web possono essere eseguite anche da riga di comando tramite la CLI (Command Line Interface) messa a disposizione da GCE.
Si può accedere alla CLI tramite un apposito container (`gvm-tools`) oppure direttamente tramite un socket del container `gvmd`.

Accesso alla CLI tramite il container `gvm-tools`

Esecuzione ed accesso al container

```
docker-compose -f $GCE_CONTAINER_DIR/docker-compose.yml -p greenbone-
community-edition run --rm gvm-tools
```

Verifica di funzionamento della CLI: all'interno della sessione sul container `gvm-tools`, eseguire il seguente comando per visualizzare la versione di `gvmd`

```
gvm-cli --gmp-username admin socket --pretty --xml "<get_version/>"
```

e, alla richiesta, inserire la password dell'utente `admin`. Si ottiene come risposta

```
<get_version_response status="200" status_text="OK">
  <version>22.4</version>
</get_version_response>
```

Installazione dei `gvm-tools`

Per accedere alla CLI tramite socket (sezione successiva) è necessario installare i `gvm-tools`

```
python3 -m pip install --user gvm-tools
```

Accesso alla CLI tramite socket

Accedere alla CLI di GCE tramite il container è una procedura non molto comoda da utilizzare solo in parrticolari situazioni tipo troubleshooting: risulta complicata da usare con frequenza o integrarla in appositi script.
Si preferisce accedere tramite socket a gvmd ed utilizzare il protocollo GMP (Greenbone Management Protocol) basato su XML col quale si può effettuare da riga di comando tutte le operazioni previste dall'interfaccia web.

Modifica alla configurazione per rendere disponibile il socket di `gvmd`

1. Creazione della directory opportuna

```
sudo mkdir -p /GVM/GVMD
sudo chmod -R 777 /GVM
```

2. Modifica del file del *docker compose file* di GCE

```
gvmd:
    image: greenbone/gvmd:stable
    restart: on-failure
    volumes:
       - gvmd_data_vol:/var/lib/gvm
       - vt_data_vol:/var/lib/openvas
       - psql_data_vol:/var/lib/postgresql
-      - gvmd_socket_vol:/run/gvmd           <--- sostituire questa riga
con
+      - /GVM/GVMD:/run/gvmd                 <--- questa
       - ospd_openvas_socket_vol:/run/ospd
       - psql_socket_vol:/var/run/postgresql
     depends_on:
      - pg-gvm

[...]

  gsa:
    image: greenbone/gsa:stable
    restart: on-failure
     ports:
       - 9392:80
     volumes:
-      - gvmd_socket_vol:/run/gvmd          <--- sostituire questa riga
con
+      - /GVM/GVMD:/run/gvmd                <--- questa
     depends_on:
       - gvmd
```

3. Riavvio del container
   Comando:

```
docker-compose -f $GCE_CONTAINER_DIR/docker-compose.yml -p greenbone-
community-edition up -d
```

   Output:

```
Starting greenbone-community-edition_cert-bund-data_1      ... done
Starting greenbone-community-edition_vulnerability-tests_1 ... done
Starting greenbone-community-edition_notus-data_1          ... done
Starting greenbone-community-edition_gpg-data_1            ... done
Starting greenbone-community-edition_data-objects_1        ... done
Starting greenbone-community-edition_scap-data_1           ... done
Starting greenbone-community-edition_dfn-cert-data_1       ... done
greenbone-community-edition_notus-scanner_1 is up-to-date
Starting greenbone-community-edition_report-formats_1      ... done
Recreating greenbone-community-edition_gvmd_1              ... done
Recreating greenbone-community-edition_gsa_1               ...
Recreating greenbone-community-edition_gsa_1               ... done
Recreating greenbone-community-edition_gvm-tools_1         ... done
```

4. Verifica del funzionamento
   Richiedere la versione di `gvmd` con il seguente comando ed inserire username e password qando richiesti
   Comando:

```
gvm-cli socket --socketpath /GVM/GVMD/gvmd.sock --pretty --xml "
<get_version/>"
```

   Output:

```
Enter username: admin
Enter password for admin:
<get_version_response status="200" status_text="OK">
  <version>22.4</version>
</get_version_response>
```

Uso della CLI

Qaundo si creano *Port List*, *Scan Configs*, *Credentials*, *Targets*, *Tasks*, ... si consiglia di **non** utilizzare spazi o caratteri particolari nel definire i nomi (*Name*) per evitare di dover usare le virgolette e gli escape in tutti i comandi da shell. L'`undersore` invece può essere utilizzato.

Salvataggio delle credenziali

Creare il file `~/.config/gvm-tools.conf` con il seguente contenuto

```
[unixsocket]
socketpath=/GVM/GVMD/gvmd.sock
```

```
[gmp]
username=admin
password=1-love-CCR-tooo-much!
```

Eseguendo il seguente comando di dovrebbe ottenere subito l'output senza dover inserire le credenziali
Comando:

```
gvm-cli socket --socketpath /GVM/GVMD/gvmd.sock --pretty --xml "
<get_version/>"
```

Output:

```
<get_version_response status="200" status_text="OK">
  <version>22.4</version>
</get_version_response>
```

Alias per semplificare i comandi

Per semplificare l'invio di comandi tramite CLI può essere utile impostare il seguente alias in `~/.bashrc`

```
alias mygvm-cli="gvm-cli socket --socketpath /GVM/GVMD/gvmd.sock --pretty --
xml"
```

In questo modo, una volta ricaricate le variabili d'ambiente con la modica appena fatta

```
source ~/.bashrc
```

si possono dare i comandi direttamente nel seguente modo
Comando:

```
mygvm-cli "<get_version/>"
```

Output:

```
<get_version_response status="200" status_text="OK">
  <version>22.4</version>
</get_version_response>
```

Uso di `xml2`

Per elaborare in modo semplice l'output dei comandi dati tramite la CLI può essere utile utlizzare `xml2` che converte un `xml` in un formato "piatto".

L'output di `xml2` risulta, per esempio, più semplice da elaborare tramite `grep` e `sed`.
Per installare il pacchetto eseguire il seguente comando

```
sudo apt install -y xml2
```

Per vedere cosa fa `xml2`, per esempio, eseguire i seguenti comandi e confrontare l'output.

Comando:

```
mygvm-cli "<get_version/>"
```

Output:

```
<get_version_response status="200" status_text="OK">
  <version>22.4</version>
</get_version_response>
```

Comando:

```
mygvm-cli "<get_version/>" | xml2
```

Output:

```
/get_version_response/@status=200
/get_version_response/@status_text=OK
/get_version_response/version=22.4
```

In questo formato ogni riga riporta sia ciascun **attributo** (preceduto dal segno `@`) che ciascun **elemento** entrambi con il loro percorso completo.

Comandi dalla CLI

Si suppone di aver creato tramite l'interfaccia web:

- una *Port Lists* con nome `INFN_221008`
- una *Scan Config* con nome `INFN_221008`
  e di aver definito il seguente alias

```
alias mygvm-cli="gvm-cli socket --socketpath /GVM/GVMD/gvmd.sock --pretty --
xml"
```

Comandi per visualizzare le *Port Lists*

- Elenco di tutte le *Port Lists*.

```
mygvm-cli "<get_port_lists/>"

### oppure ###

mygvm-cli "<get_port_lists/>" | xml2

### oppure ###

mygvm-cli "<get_port_lists/>" | xml2 | \
grep
'^/get_port_lists_response/port_list/name=\|^/get_port_lists_response/por
t_list/@id='
```

- Visualizzazione di una particolare *Port List*.
  - Ricerca per **nome**

```
mygvm-cli "<get_port_lists filter='name=INFN_221008'/>"

### oppure ###

mygvm-cli "<get_port_lists filter='name=INFN_221008'/>" | xml2

### oppure ###

mygvm-cli "<get_port_lists filter='name=INFN_221008'/>" | xml2 | \
grep
'^/get_port_lists_response/port_list/name=\|^/get_port_lists_respons
e/port_list/@id='

### oppure (se ci sono degli spazi nel nome, escape!) ###

mygvm-cli "<get_port_lists filter='name=\"All IANA assigned
TCP\"'/>" | xml2 | \
grep
'^/get_port_lists_response/port_list/name=\|^/get_port_lists_respons
e/port_list/@id='
```

  - Ricerca per **ID**

```
mygvm-cli "<get_port_lists port_list_id='33d0cd82-57c6-11e1-8ed1-
406186ea4fc5'/>"

### oppure ###
```

```
mygvm-cli "<get_port_lists port_list_id='33d0cd82-57c6-11e1-8ed1-406186ea4fc5'/>" | xml2

### oppure ###

mygvm-cli "<get_port_lists port_list_id='33d0cd82-57c6-11e1-8ed1-406186ea4fc5'/>" | xml2 | grep '^/get_port_lists_response/port_list/name=\|^/get_port_lists_response/port_list/@id='
```

Comandi per visualizzare le *Scan Configs*

- Elenco di tutte le *Scan Config*.

```
mygvm-cli "<get_configs/>"

### oppure ###

mygvm-cli "<get_configs/>" | xml2

### oppure ###

mygvm-cli "<get_configs/>" | xml2 | \
grep '^/get_configs_response/config/@id=\|^/get_configs_response/config/name='
```

- Visualizzazione di una particolare *Scan Config*.
  - Ricerca per **nome**

```
mygvm-cli "<get_configs filter='name=INFN_221008'/>"

### oppure ###

mygvm-cli "<get_configs filter='name=INFN_221008'/>" | xml2

### oppure ###

mygvm-cli "<get_configs filter='name=INFN_221008'/>" | xml2 | \
grep '^/get_configs_response/config/@id=\|^/get_configs_response/config/name='

### oppure (se ci sono degli spazi nel nome, escape!) ###
```

```
mygvm-cli "<get_configs filter='name=\"Full and fast\"'/>" | xml2 | \
grep
'^/get_configs_response/config/@id=\|^/get_configs_response/config/name='
```

- Ricerca per **ID**

```
mygvm-cli "<get_configs config_id='daba56c8-73ec-11df-a475-
002264764cea'/>"

### oppure ###

mygvm-cli "<get_configs config_id='daba56c8-73ec-11df-a475-
002264764cea'/>" | xml2

### oppure ###

mygvm-cli "<get_configs config_id='daba56c8-73ec-11df-a475-
002264764cea'/>" | xml2 | grep
'^/get_configs_response/config/name\|^//get_configs_response/config/
@id='
```

Comandi per impostare un *Target*

Si considera di voler creare il seguente *Target* con

- Nome: `Bersaglio`
- IP(s):
  - `192.168.0.0/24`
  - `172.16.0.1`
  - `172.16.0.2`
  - `172.16.0.3`
- IP(s) da escludere:
  - `192.168.0.10`
  - `192.168.0.20`
  - `192.168.0.30`
- Port List: `INFN_221008`

Come prima cosa si deve trovare l'ID della *Port List*, che si vuole usare, con il seguente comando

```
PORT_LIST_NAME="INFN_221008"
PORT_LIST_ID=$( mygvm-cli "<get_port_lists
filter='name=\"$PORT_LIST_NAME\"'/>" | xml2 | \
```

```
                grep '^/get_port_lists_response/port_list/@id=' | \
                sed 's/^\/get_port_lists_response\/port_list\/@id=//')
```

Per semplificare la scrittura del comando si definiscono delle variabili

```
NAME="Bersaglio"
HOSTS="192.168.0.0/24,172.16.0.1,172.16.0.2,172.16.0.3"
EXCLUDE_HOSTS="192.168.0.10, 192.168.0.20, 192.168.0.30"
```

Con le variabili così definite e con l'ID della *Port List* ricavato in precedenza, il comando per impostare il *Target* è il seguente

```
mygvm-cli "<create_target><name>$NAME</name><hosts>$HOSTS</hosts>
<exclude_hosts>$EXCLUDE_HOSTS</exclude_hosts><port_list id='$PORT_LIST_ID'/>
</create_target>"
```

L'output del comando precedente restitutisce l'ID del *Target* appena definito.
Se si vuol salvare l'ID del *Target* in una variabile (`TARGET_ID`) si può utilizzare in sostituzione al precedente comando, il seguente comando

```
TARGET_ID=$(mygvm-cli "<create_target><name>$NAME</name><hosts>$HOSTS</hosts>
<exclude_hosts>$EXCLUDE_HOSTS</exclude_hosts><port_list id='$PORT_LIST_ID'/>
</create_target>" | sed 's/^.*id=//' | cut -d "/" -f 1 | sed 's/\"//g')
```

Comandi per visualizzare un *Target*

- Elenco di tutti i *Target*.

  ```
  mygvm-cli "<get_targets/>"

  ### oppure ###

  mygvm-cli "<get_targets/>" | xml2

  ### oppure ###

  mygvm-cli "<get_targets/>" | xml2 | \
  grep
  '^/get_targets_response/target/@id=\|^/get_targets_response/target/name='
  ```

- Visualizzazione di un particolare *Target*.

  - Ricerca per **nome**

```
mygvm-cli "<get_targets filter='name=Bersaglio'/>"

### oppure ###

mygvm-cli "<get_targets filter='name=Bersaglio'/>" | xml2

### oppure ###

mygvm-cli "<get_targets filter='name=Bersaglio'/>" | xml2 | \
grep
'^/get_targets_response/target/@id=\|^/get_targets_response/target/name='

### oppure (se ci sono degli spazi nel nome, escape!) ###

mygvm-cli "<get_targets filter='name=\"Prova di bersaglio\"'/>" | xml2 |
\
grep
'^/get_targets_response/target/@id=\|^/get_targets_response/target/name='
```

- Ricerca per **ID**

```
mygvm-cli "<get_targets target_id='551f5200-b8b5-49c1-b643-
3cb02a52fb6f'/>"

### oppure ###

mygvm-cli "<get_targets target_id='551f5200-b8b5-49c1-b643-
3cb02a52fb6f'/>" | xml2

### oppure ###

mygvm-cli "<get_targets target_id='551f5200-b8b5-49c1-b643-
3cb02a52fb6f'/>" | xml2 | grep
'^/get_targets_response/target/@id=\|^/get_targets_response/target/n
ame='
```

Comandi per impostare un *Task*

Si considera di voler creare il seguente *Task* (alterabile) con

- *Target*: `Bersaglio`
- *Scan Config*: `INFN_221008`

Come prima cosa si deve trovare l'ID del *Target*

```
TARGET_NAME="Bersaglio"
TARGET_ID=$(mygvm-cli "<get_targets filter='name=$TARGET_NAME'/>" | xml2 | \
        grep '^/get_targets_response/target/@id=' | sed
's/^\/get_targets_response\/target\/@id=//')
```

e l'ID della *Scan Config*

```
SCAN_CONFIG_NAME="INFN_221008"
SCAN_CONFIG_ID=$(mygvm-cli "<get_configs filter='name=$SCAN_CONFIG_NAME'/>" |
xml2 |    grep '^/get_configs_response/config/@id=' | sed
's/^\/get_configs_response\/config\/@id=//')
```

Per semplificare la scrittura del comando si definiscono delle variabili

```
NAME="Compito"
```

Con la variabile `NAME` così definita e con gli ID del *Target* e della *Scan Config* ricavate in precedenza, il comando per impostare il *Task* è il seguente

```
mygvm-cli "<create_task><name>$NAME</name><alterable>1</alterable><config
id='$SCAN_CONFIG_ID'></config><target id='$TARGET_ID'></target></create_task>"
```

L'output del comando precedente restitutisce l'ID del *Target* appena definito.
Se si vuol salvare l'ID del Target in una variabile (`TARGET_ID`) si può utilizzare in sostituzione al precedente comando, il seguente comando

```
TASK_ID=$(mygvm-cli "<create_task><name>$NAME</name><alterable>1</alterable>
<config id='$SCAN_CONFIG_ID'></config><target id='$TARGET_ID'></target>
</create_task>" | sed 's/^.*id=//' | cut -d "/" -f 1 | sed 's/\"//g')
```

Comandi per visualizzare un *Task* ed il suo stato di esecuzione

- Elenco di tutti i *Task*.

```
mygvm-cli "<get_tasks/>"

### oppure ###

mygvm-cli "<get_tasks/>" | xml2

### oppure ###

mygvm-cli "<get_tasks/>" | xml2 | \
grep
```

```
'^/get_tasks_response/task/@id=\|^/get_tasks_response/task/name=\|^/get_tasks_response/task/status=\|/get_tasks_response/task/progress='
```

- Visualizzazione di un particolare *Task*.
  - Ricerca per **nome**

```
mygvm-cli "<get_tasks filter='name=Compito'/>"

### oppure ###

mygvm-cli "<get_tasks filter='name=Compito'/>" | xml2

### oppure ###

mygvm-cli "<get_tasks filter='name=Compito'/>" | xml2 | \
grep
'^/get_tasks_response/task/@id=\|^/get_tasks_response/task/name=\|^/get_tasks_response/task/status=\|/get_tasks_response/task/progress='

### oppure (se ci sono degli spazi nel nome, escape!) ###

mygvm-cli "<get_tasks filter='name=\"Compito di prova\"'/>" | xml2 | \
grep
'^/get_tasks_response/task/@id=\|^/get_tasks_response/task/name=\|^/get_tasks_response/task/status=\|/get_tasks_response/task/progress='
```

  - Ricerca per **ID**

```
mygvm-cli "<get_tasks task_id='25ad41ba-50fd-431e-99d6-556ec6d02054'/>"

### oppure ###

mygvm-cli "<get_tasks task_id='25ad41ba-50fd-431e-99d6-556ec6d02054'/>" | xml2

### oppure ###

mygvm-cli "<get_tasks task_id='25ad41ba-50fd-431e-99d6-556ec6d02054'/>" | xml2 | grep
'^/get_tasks_response/task/@id=\|^/get_tasks_response/task/name=\|^/get_tasks_response/task/status=\|/get_tasks_response/task/progress='
```

Comandi per avviare le scansioni (eseguire i *Task*)

Si considera di aver già creato un *Task* con nome: `Compito`.
Come prima cosa si deve trovare l'ID del *Task*

```
TASK_NAME="Compito"
TASK_ID=$(mygvm-cli "<get_tasks filter='name=$TASK_NAME'/>" | xml2 | \
        grep '^/get_tasks_response/task/@id=' | sed
's/^\/get_tasks_response\/task\/@id=//')
```

Avvio del *Task* di scansione

```
mygvm-cli "<start_task task_id='$TASK_ID'/>"
```

Comandi per esportare un *Report* di un *Task*

Si considera di voler esportare l'ultimo report del *Task* con nome: `Compito`.
Come prima cosa si deve trovare l'ID dell'ultimo *Report*

```
TASK_NAME="Compito"
REPORT_ID=$(mygvm-cli "<get_tasks filter='name=$TASK_NAME'/>" | xml2 | \
        grep '^/get_tasks_response/task/last_report/report/@id=' | sed
's/^\/get_tasks_response\/task\/last_report\/report\/@id=//')
```

Devono essere inoltre ricavati gli ID relativi ai vari formati di esportazione dei *Report* con la seguente procedura.

```
REPORT_FORMATS=$(mygvm-cli "<get_report_formats/>" | xml2 | grep -E
'^/get_report_formats_response/report_format/@id=|/get_report_formats_response
/report_format/name=' | cut -d "=" -f 2 | paste -sd '|\n')
REPORT_FORMAT_TXT_ID=$(echo "$REPORT_FORMATS" | grep '|TXT$' | cut -d "|" -f
1)
REPORT_FORMAT_PDF_ID=$(echo "$REPORT_FORMATS" | grep '|PDF$' | cut -d "|" -f
1)
REPORT_FORMAT_CSV_ID=$(echo "$REPORT_FORMATS" | grep '|CSV Results$' | cut -d
"|" -f 1)
REPORT_FORMAT_XML_ID=$(echo "$REPORT_FORMATS" | grep '|XML$' | cut -d "|" -f
1)
```

- Esportazione dei *Report* in formato TXT nel file `$TASK_NAME-report.txt`

```
mygvm-cli "<get_reports report_id='$REPORT_ID' filter='rows=-1 levels=hmlg'
format_id='$REPORT_FORMAT_TXT_ID' details='1'/>" | sed 1,1d | sed s'/^.*
```

```
<\/report_format>//' | head -n 1 | sed 's/<\/report>//' | base64 -d >
$TASK_NAME-report.txt
```

- Esportazione dei *Report* in formato PDF nel file `$TASK_NAME-report.pdf`

```
mygvm-cli "<get_reports report_id='$REPORT_ID' filter='rows=-1 levels=hmlg'
format_id='$REPORT_FORMAT_PDF_ID' details='1'/>" | sed 1,1d | sed s'/^.*
<\/report_format>//' | head -n 1 | sed 's/<\/report>//' | base64 -d >
$TASK_NAME-report.pdf
```

- Esportazione dei *Report* in formato CSV nel file `$TASK_NAME-report.csv`

```
mygvm-cli "<get_reports report_id='$REPORT_ID' filter='rows=-1 levels=hmlg'
format_id='$REPORT_FORMAT_CSV_ID' details='1'/>" | sed 1,1d | sed s'/^.*
<\/report_format>//' | head -n 1 | sed 's/<\/report>//' | base64 -d >
$TASK_NAME-report.csv
```

- Esportazione dei *Report* in formato XML nel file `$TASK_NAME-report.xml`

```
mygvm-cli "<get_reports report_id='$REPORT_ID' filter='rows=-1 levels=hmlg'
format_id='$REPORT_FORMAT_XML_ID' details='1'/>" > $TASK_NAME-report.xml
```

Cancellazione di un *Report*

Si considera di voler cancellare l'ultimo report del *Task* con nome: `Compito`.
Come prima cosa si deve trovare l'ID dell'ultimo *Report*

```
TASK_NAME="Compito"
REPORT_ID=$(mygvm-cli "<get_tasks filter='name=$TASK_NAME'/>" | xml2 | \
        grep '^/get_tasks_response/task/last_report/report/@id=' | sed
's/^\/get_tasks_response\/task\/last_report\/report\/@id=//')
```

Eliminazione del *Report*

```
mygvm-cli "<delete_report report_id='$REPORT_ID'/>"
```

Cancellazione di un *Task*

Si considera di voler cancellare il *Task* con nome: `Compito`.
Come prima cosa si deve trovare l'ID del *Task*

```
TASK_NAME="Compito"
TASK_ID=$(mygvm-cli "<get_tasks filter='name=$TASK_NAME'/>" | xml2 | \
```

```
        grep '^/get_tasks_response/task/@id=' | sed
's/^\/get_tasks_response\/task\/@id=//')
```

Eliminazione del *Task*

```
mygvm-cli "<delete_task task_id='$TASK_ID'/>"
```

Cancellazione di un *Target*

Si considera di voler cancellare il *Target* con nome: `Bersaglio`.
Come prima cosa si deve trovare l'ID del *Target*

```
TARGET_NAME="Bersaglio"
TARGET_ID=$(mygvm-cli "<get_targets filter='name=$TARGET_NAME'/>" | xml2 | \
        grep '^/get_targets_response/target/@id=' | sed
's/^\/get_targets_response\/target\/@id=//')
```

Eliminazione del *Target*

```
mygvm-cli "<delete_target target_id='$TARGET_ID'/>"
```

Una procedura tipica

L'interfaccia web permette di definire in modo molto semplice le *Port Lists* e le *Scan Configs*, oppure possono essere importate da shell tramite file XML precedentemente esportati.
In processi automatizzati o ripetuti o su un gran numero di IP magari raggruppati in modi particolari può essere molto utile gestire le scansioni tramite script di shell.

Si studia in dettaglio una procedura tipica in cui si esegue una scansione su certi IP e si esportano i risultati per poi cancellare tutto utilizzando i seguenti parametri

- Nome della scansione: `Scansione_di_prova`
- IP(s):
  - `192.168.0.0/24`
  - `172.16.0.1`
  - `172.16.0.2`
  - `172.16.0.3`
- IP(s) da escludere:
  - `192.168.0.10`
  - `192.168.0.20`
  - `192.168.0.30`
- Port List: `INFN_221008`
- Scan Config: `INFN_221008`

La procedura prevede di

- definizione di un Target;
- definizione di un Task
- avvio della scansione;
- verifica dello stato di avanzamento della scansione;
- export del Report;
- eliminazione dei Report, Task e Target.

Esecuzione della procedura

1. Definizione di tutte le variabili

```
# nome della scansione
SCAN_NAME="Scansione_di_prova"

# ip da passare a scansione e da escludere
HOSTS="192.168.0.0/24,172.16.0.1,172.16.0.2,172.16.0.3"
EXCLUDE_HOSTS="192.168.0.10, 192.168.0.20, 192.168.0.30"

# scelta della port list
PORT_LIST_NAME="INFN_221008"

# scelta della scan config
SCAN_CONFIG_NAME="INFN_221008"
```

2. Definizione di un Target

```
TARGET_NAME="Target_$SCAN_NAME"

PORT_LIST_ID=$( mygvm-cli "<get_port_lists
filter='name=\"$PORT_LIST_NAME\"'/>" | xml2 | \
grep '^/get_port_lists_response/port_list/@id=' | \
sed 's/^\/get_port_lists_response\/port_list\/@id=//')

TARGET_ID=$(mygvm-cli "<create_target><name>$TARGET_NAME</name>
<hosts>$HOSTS</hosts><exclude_hosts>$EXCLUDE_HOSTS</exclude_hosts>
<port_list id='$PORT_LIST_ID'/></create_target>" | sed 's/^.*id=//' | cut
-d "/" -f 1 | sed 's/\"//g')
```

3. Definizione di un Task

```
TASK_NAME="Task_$SCAN_NAME"

SCAN_CONFIG_ID=$(mygvm-cli "<get_configs
filter='name=$SCAN_CONFIG_NAME'/>" | xml2 |    grep
```

```
'^/get_configs_response/config/@id=' | sed
's/^\/get_configs_response\/config\/@id=//')

TASK_ID=$(mygvm-cli "<create_task><name>$TASK_NAME</name>
<alterable>1</alterable><config id='$SCAN_CONFIG_ID'></config><target
id='$TARGET_ID'></target></create_task>" | sed 's/^.*id=//' | cut -d "/"
-f 1 | sed 's/\"//g')
```

4. Avvio della scansione

```
mygvm-cli "<start_task task_id='$TASK_ID'/>"
```

5. Verifica dello stato di avanzamento della scansione

```
mygvm-cli "<get_tasks task_id='$TASK_ID'/>" | xml2 | grep
'^/get_tasks_response/task/@id=\|^/get_tasks_response/task/name=\|^/get_t
asks_response/task/status=\|/get_tasks_response/task/progress='
```

6. Export del Report (solo a compimento del task di scansione)

```
REPORT_ID=$(mygvm-cli "<get_tasks filter='name=$TASK_NAME'/>" | xml2 | \
grep '^/get_tasks_response/task/last_report/report/@id=' | sed
's/^\/get_tasks_response\/task\/last_report\/report\/@id=//')

REPORT_FORMATS=$(mygvm-cli "<get_report_formats/>" | xml2 | grep -E
'^/get_report_formats_response/report_format/@id=|/get_report_formats_res
ponse/report_format/name=' | cut -d "=" -f 2 | paste -sd '|\n')
REPORT_FORMAT_TXT_ID=$(echo "$REPORT_FORMATS" | grep '|TXT$' | cut -d "|"
-f 1)
REPORT_FORMAT_PDF_ID=$(echo "$REPORT_FORMATS" | grep '|PDF$' | cut -d "|"
-f 1)
REPORT_FORMAT_CSV_ID=$(echo "$REPORT_FORMATS" | grep '|CSV Results$' |
cut -d "|" -f 1)
REPORT_FORMAT_XML_ID=$(echo "$REPORT_FORMATS" | grep '|XML$' | cut -d "|"
-f 1)

mygvm-cli "<get_reports report_id='$REPORT_ID' filter='rows=-1
levels=hmlg' format_id='$REPORT_FORMAT_TXT_ID' details='1'/>" | sed 1,1d
| sed s'/^.*<\/report_format>//' | head -n 1 | sed 's/<\/report>//' |
base64 -d > $TASK_NAME-report.txt

mygvm-cli "<get_reports report_id='$REPORT_ID' filter='rows=-1
levels=hmlg' format_id='$REPORT_FORMAT_PDF_ID' details='1'/>" | sed 1,1d
| sed s'/^.*<\/report_format>//' | head -n 1 | sed 's/<\/report>//' |
```

```
base64 -d > $TASK_NAME-report.pdf

mygvm-cli "<get_reports report_id='$REPORT_ID' filter='rows=-1
levels=hmlg' format_id='$REPORT_FORMAT_CSV_ID' details='1'/>" | sed 1,1d
| sed s'/^.*<\/report_format>//' | head -n 1 | sed 's/<\/report>//' |
base64 -d > $TASK_NAME-report.csv

mygvm-cli "<get_reports report_id='$REPORT_ID' filter='rows=-1
levels=hmlg' format_id='$REPORT_FORMAT_XML_ID' details='1'/>" >
$TASK_NAME-report.xml
```

7. Eliminazione dei Report, Task e Target

```
mygvm-cli "<delete_report report_id='$REPORT_ID'/>"
mygvm-cli "<delete_task task_id='$TASK_ID'/>"
mygvm-cli "<delete_target target_id='$TARGET_ID'/>"
```