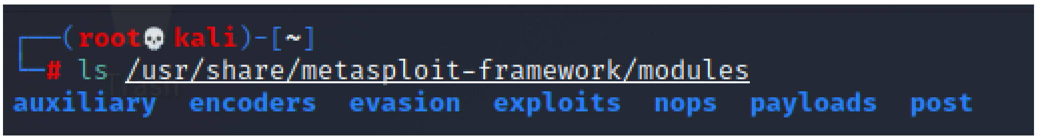
Aryaman Mishra

19BCE1027

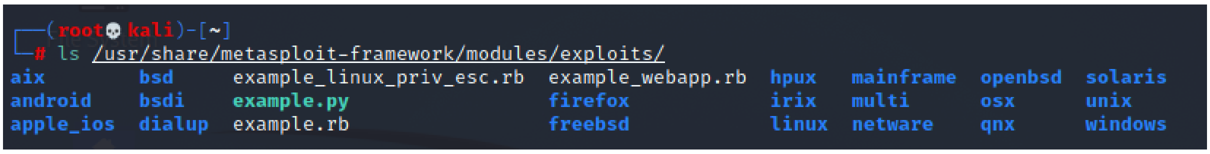
i) Browse through various folders of Metasploit and explore the folders like payload, exploits and write a paragraph about every folder and one script in every folder

Almost all of your interaction with Metasploit will be through its many *modules*, which it looks for in two locations. The first is the primary module store under **/usr/share/metasploit-framework/modules/** and the second, which is where you will store custom modules, is under your home directory at **~/.msf4/modules/**.



```
┌──(root💀kali)-[~]
└─# ls /usr/share/metasploit-framework/modules
auxiliary  encoders  evasion  exploits  nops  payloads  post
```

All Metasploit modules are organized into separate directories, according to their purpose. A basic overview of the various types of Metasploit modules is shown below.

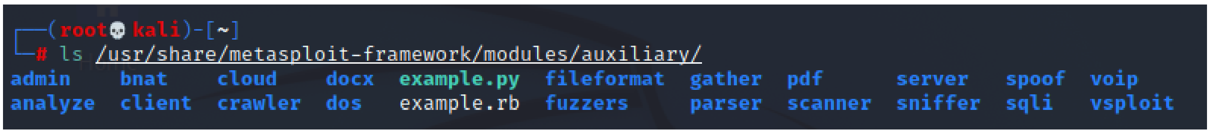


```
┌──(root💀kali)-[~]
└─# ls /usr/share/metasploit-framework/modules/exploits/
aix        bsd     example_linux_priv_esc.rb  example_webapp.rb  hpux   mainframe  openbsd  solaris
android    bsdi    example.py                 firefox            irix   multi      osx      unix
apple_ios  dialup  example.rb                 freebsd            linux  netware    qnx      windows
```

Example.py

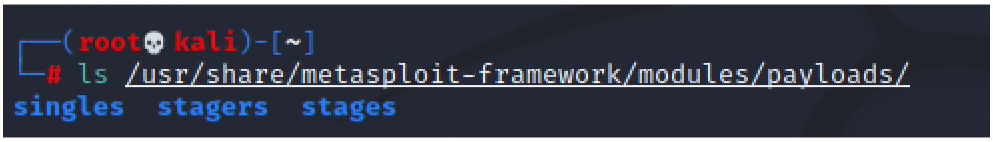
Resource scripts provide an easy way for you to automate repetitive tasks in Metasploit. Conceptually, they're just like batch scripts. They contain a set of commands that are automatically and sequentially executed when you load the script in Metasploit. You can create a resource script by chaining together a series of Metasploit console commands and by directly embedding Ruby to do things like call APIs, interact with objects in the database, and iterate actions.

In the Metasploit Framework, *exploit* modules are defined as modules that use payloads.



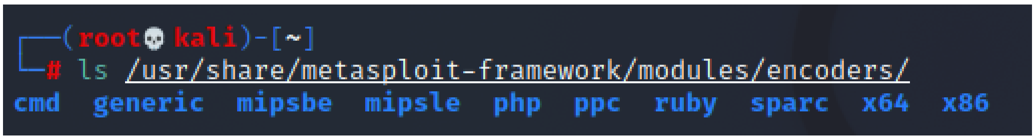
```
┌──(root💀kali)-[~]
└─# ls /usr/share/metasploit-framework/modules/auxiliary/
admin    bnat    cloud    docx  example.py  fileformat  gather  pdf      server   spoof  voip
analyze  client  crawler  dos   example.rb  fuzzers     parser  scanner  sniffer  sqli   vsploit
```

Auxiliary modules include port scanners, fuzzers, sniffers, and more.



```
┌──(root💀kali)-[~]
└─# ls /usr/share/metasploit-framework/modules/payloads/
singles  stagers  stages
```

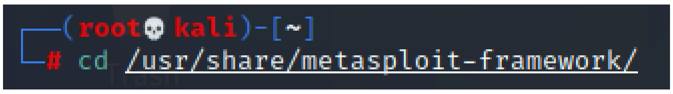
Payloads consist of code that runs remotely, while *encoders* ensure that payloads make it to their destination intact. *Nops* keep the payload sizes consistent across exploit attempts.



```
┌──(root💀kali)-[~]
└─# ls /usr/share/metasploit-framework/modules/encoders/
cmd  generic  mipsbe  mipsle  php  ppc  ruby  sparc  x64  x86
```

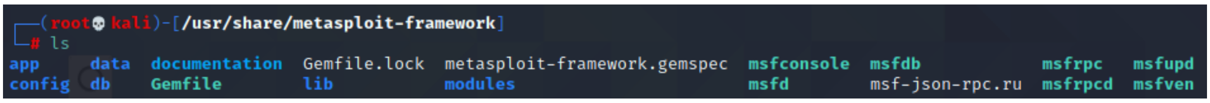
(ii)Run Information Gathering for the protocols like SMTP, secure shell, HTTP. For every protocol minimum of three scanner commands should be run.

Access Framwork folder:



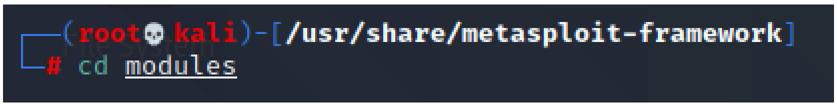
```
┌──(root💀kali)-[~]
└─# cd /usr/share/metasploit-framework/
```

View Contents of Folder:



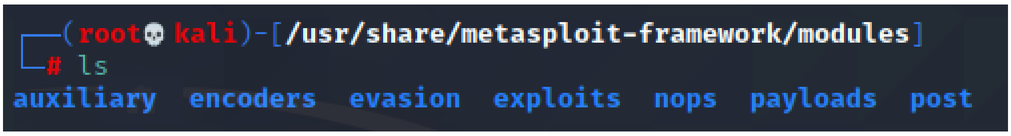
```
┌──(root💀kali)-[/usr/share/metasploit-framework]
└─# ls
app     data  documentation  Gemfile.lock  metasploit-framework.gemspec  msfconsole  msfdb            msfrpc   msfupd
config  db    Gemfile        lib           modules                       msfd        msf-json-rpc.ru  msfrpcd  msfven
```

Access Modules folder:



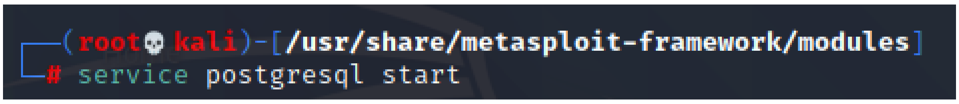
```
┌──(root💀kali)-[/usr/share/metasploit-framework]
└─# cd modules
```

View Contents of Folder:



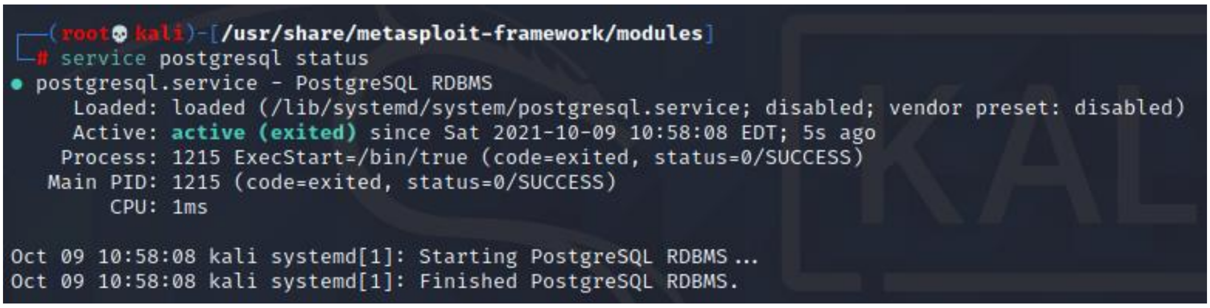
```
┌──(root💀kali)-[/usr/share/metasploit-framework/modules]
└─# ls
auxiliary  encoders  evasion  exploits  nops  payloads  post
```

Connect to Database:



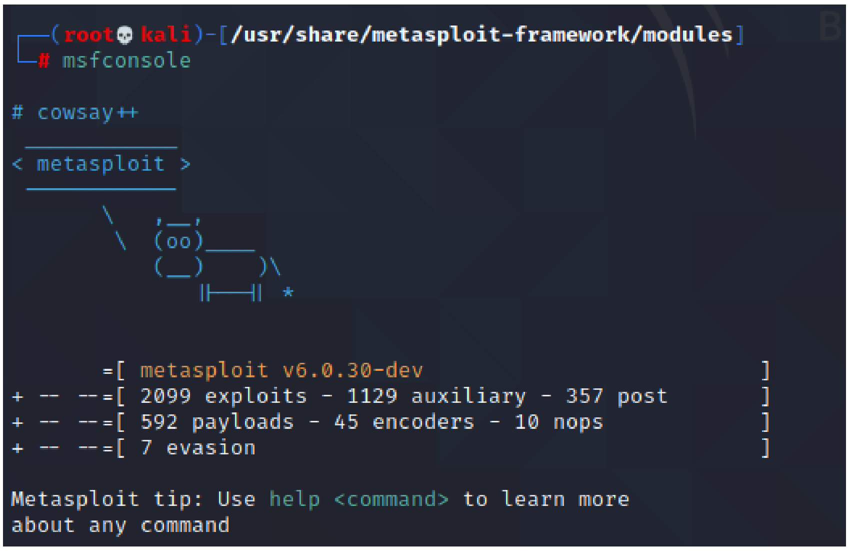
```
┌──(root💀kali)-[/usr/share/metasploit-framework/modules]
└─# service postgresql start
```

Check database status:



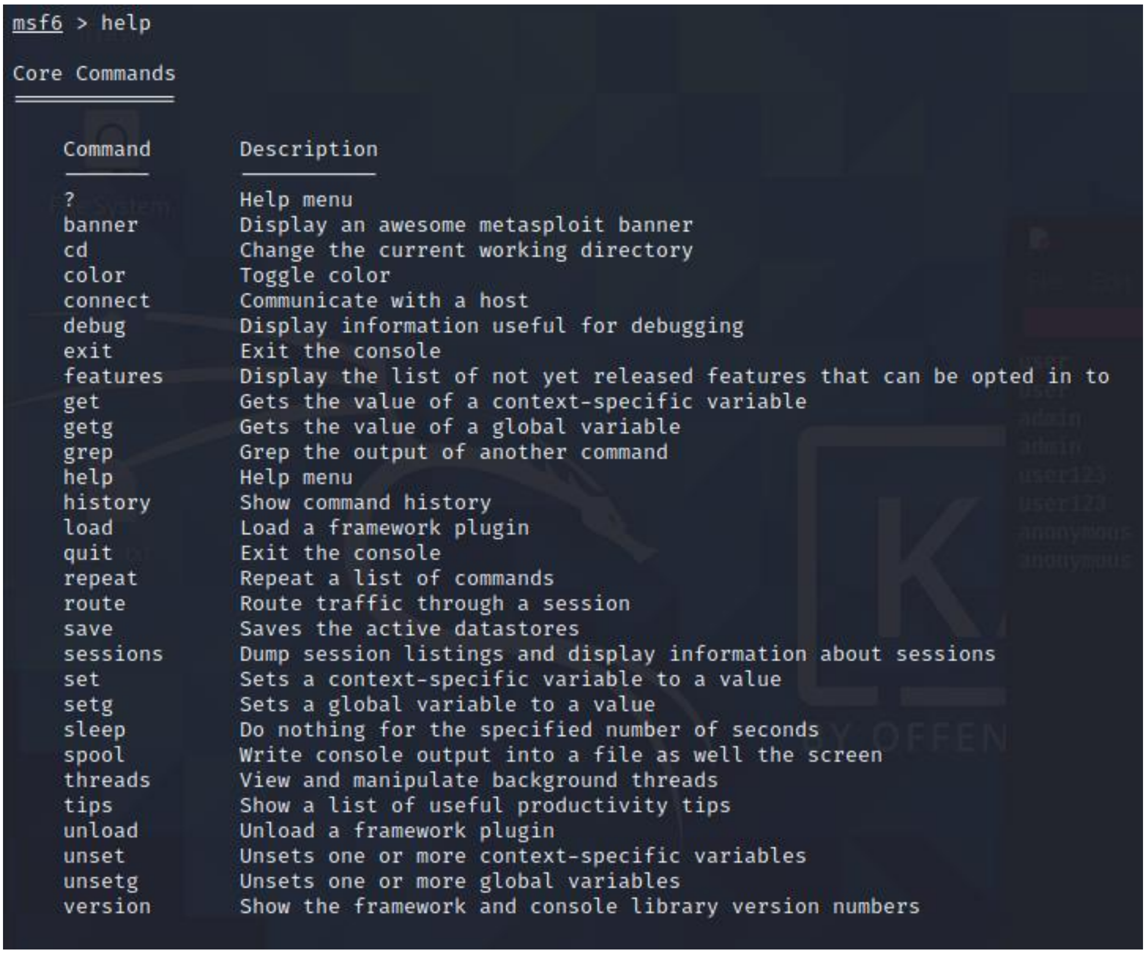
```
┌──(root💀kali)-[/usr/share/metasploit-framework/modules]
└─# service postgresql status
● postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/lib/systemd/system/postgresql.service; disabled; vendor preset: disabled)
     Active: active (exited) since Sat 2021-10-09 10:58:08 EDT; 5s ago
    Process: 1215 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 1215 (code=exited, status=0/SUCCESS)
        CPU: 1ms

Oct 09 10:58:08 kali systemd[1]: Starting PostgreSQL RDBMS...
Oct 09 10:58:08 kali systemd[1]: Finished PostgreSQL RDBMS.
```

Launch Metasploit:

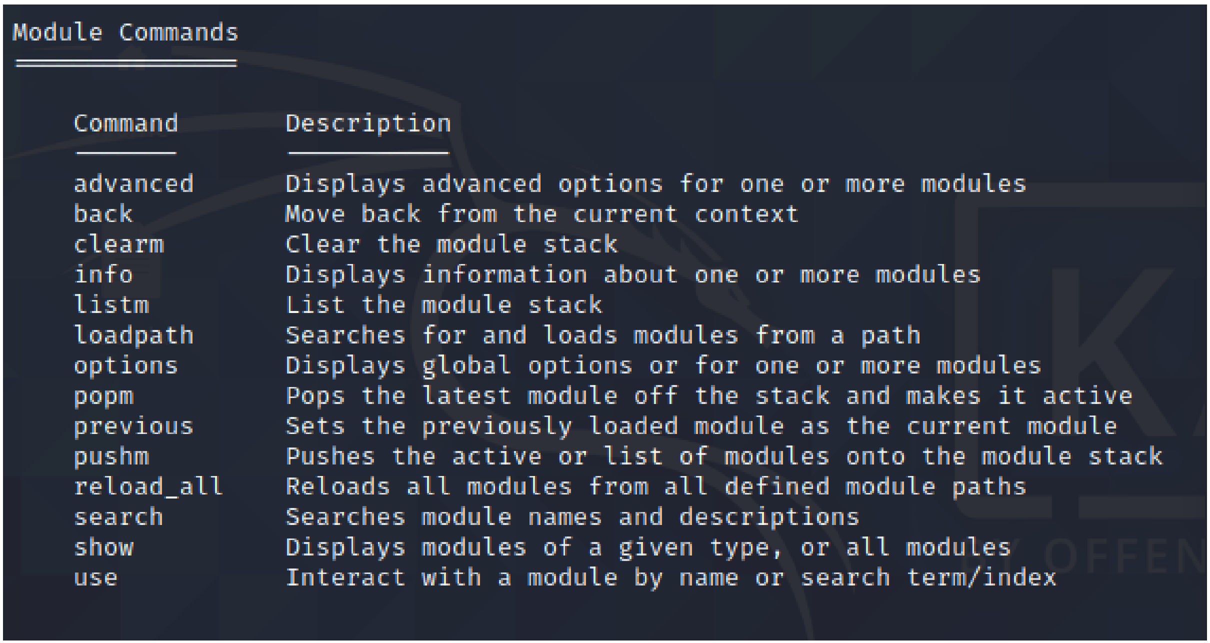


```
┌──(root💀kali)-[/usr/share/metasploit-framework/modules]
└─# msfconsole

# cowsay++
 ____________
< metasploit >
 ------------
       \   ,__,
        \  (oo)____
           (__)    )\
              ||--|| *


       =[ metasploit v6.0.30-dev                          ]
+ -- --=[ 2099 exploits - 1129 auxiliary - 357 post       ]
+ -- --=[ 592 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 7 evasion                                       ]

Metasploit tip: Use help <command> to learn more
about any command
```

View commands:

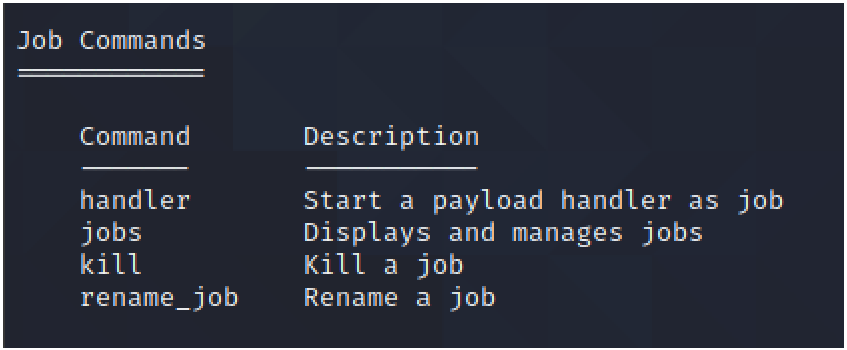


```
msf6 > help

Core Commands
=============

    Command       Description
    -------       -----------
    ?             Help menu
    banner        Display an awesome metasploit banner
    cd            Change the current working directory
    color         Toggle color
    connect       Communicate with a host
    debug         Display information useful for debugging
    exit          Exit the console
    features      Display the list of not yet released features that can be opted in to
    get           Gets the value of a context-specific variable
    getg          Gets the value of a global variable
    grep          Grep the output of another command
    help          Help menu
    history       Show command history
    load          Load a framework plugin
    quit          Exit the console
    repeat        Repeat a list of commands
    route         Route traffic through a session
    save          Saves the active datastores
    sessions      Dump session listings and display information about sessions
    set           Sets a context-specific variable to a value
    setg          Sets a global variable to a value
    sleep         Do nothing for the specified number of seconds
    spool         Write console output into a file as well the screen
    threads       View and manipulate background threads
    tips          Show a list of useful productivity tips
    unload        Unload a framework plugin
    unset         Unsets one or more context-specific variables
    unsetg        Unsets one or more global variables
    version       Show the framework and console library version numbers
```

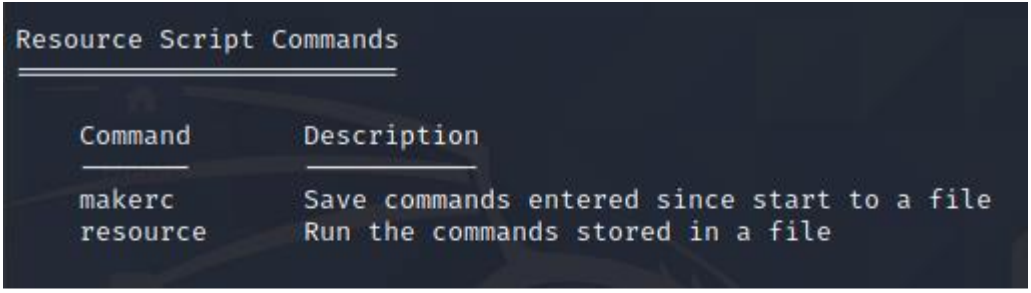


```
Module Commands
===============

    Command       Description
    -------       -----------
    advanced      Displays advanced options for one or more modules
    back          Move back from the current context
    clearm        Clear the module stack
    info          Displays information about one or more modules
    listm         List the module stack
    loadpath      Searches for and loads modules from a path
    options       Displays global options or for one or more modules
    popm          Pops the latest module off the stack and makes it active
    previous      Sets the previously loaded module as the current module
    pushm         Pushes the active or list of modules onto the module stack
    reload_all    Reloads all modules from all defined module paths
    search        Searches module names and descriptions
    show          Displays modules of a given type, or all modules
    use           Interact with a module by name or search term/index
```

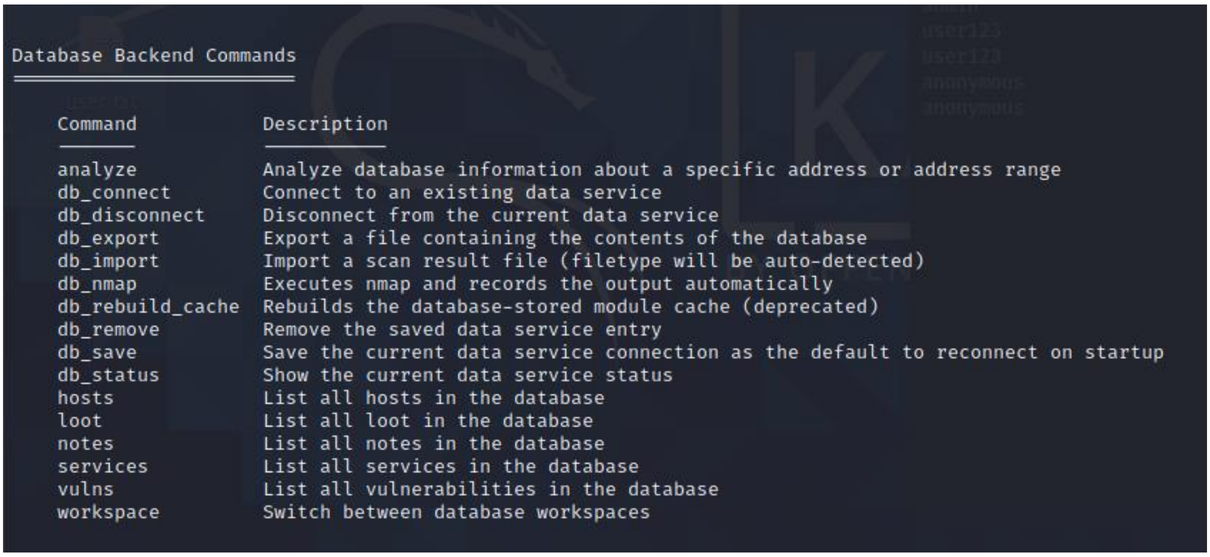
ii)



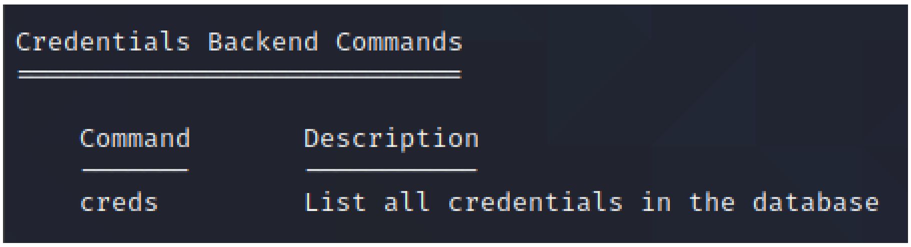
```
Job Commands
============

    Command       Description
    -------       -----------
    handler       Start a payload handler as job
    jobs          Displays and manages jobs
    kill          Kill a job
    rename_job    Rename a job
```



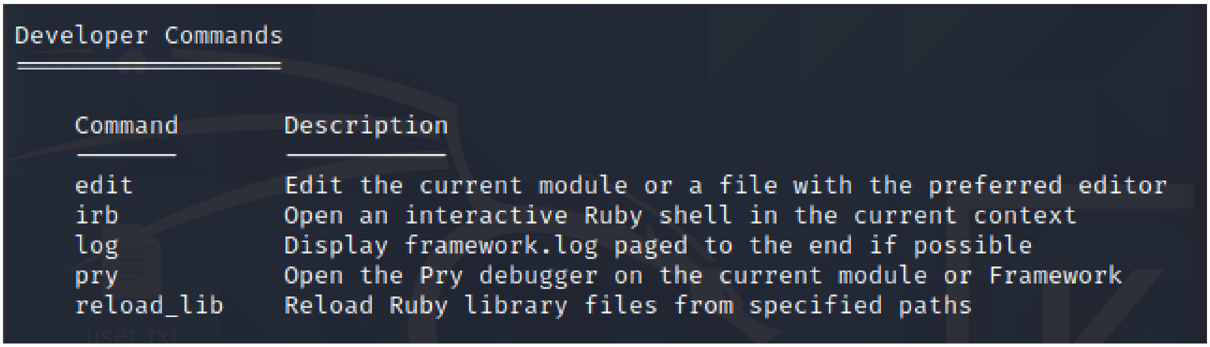
```
Resource Script Commands
========================

    Command       Description
    -------       -----------
    makerc        Save commands entered since start to a file
    resource      Run the commands stored in a file
```



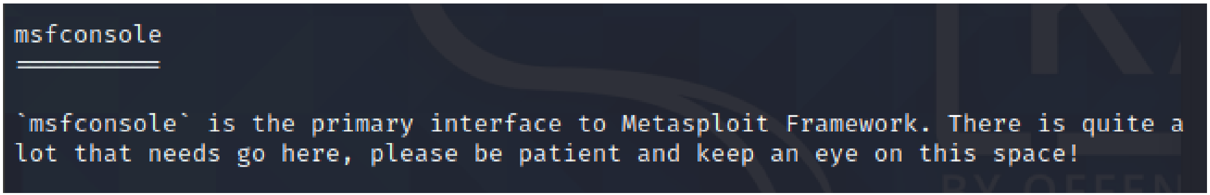
```
Database Backend Commands
=========================

    Command           Description
    -------           -----------
    analyze           Analyze database information about a specific address or address range
    db_connect        Connect to an existing data service
    db_disconnect     Disconnect from the current data service
    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 (deprecated)
    db_remove         Remove the saved data service entry
    db_save           Save the current data service connection as the default to reconnect on startup
    db_status         Show the current data service 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
```



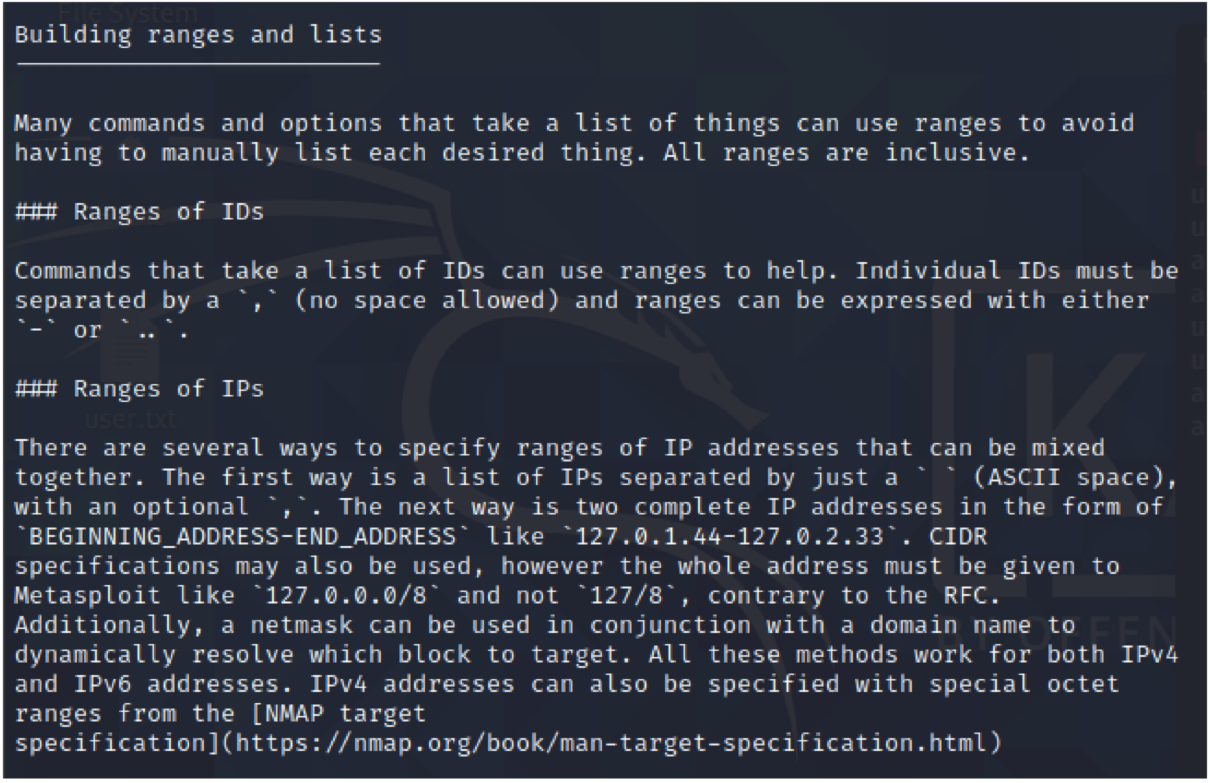
```
Credentials Backend Commands
============================

    Command       Description
    -------       -----------
    creds         List all credentials in the database
```



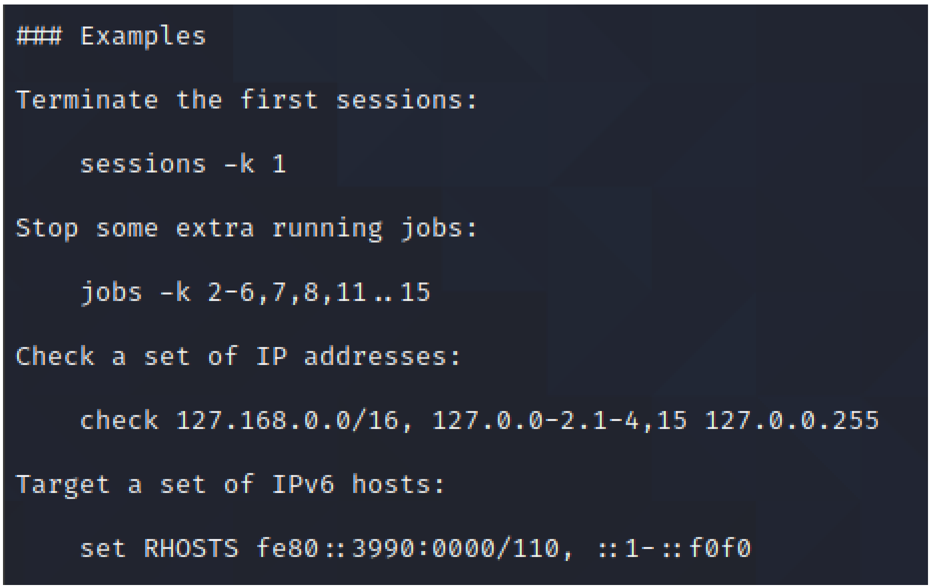
```
Developer Commands
==================

    Command       Description
    -------       -----------
    edit          Edit the current module or a file with the preferred editor
    irb           Open an interactive Ruby shell in the current context
    log           Display framework.log paged to the end if possible
    pry           Open the Pry debugger on the current module or Framework
    reload_lib    Reload Ruby library files from specified paths
```



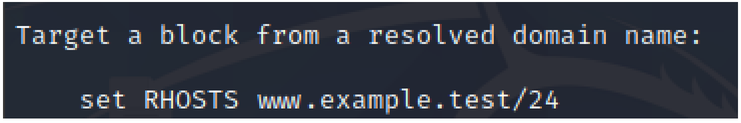
```
msfconsole
==========

`msfconsole` is the primary interface to Metasploit Framework. There is quite a
lot that needs go here, please be patient and keep an eye on this space!
```



```
Building ranges and lists
-------------------------

Many commands and options that take a list of things can use ranges to avoid
having to manually list each desired thing. All ranges are inclusive.

### Ranges of IDs

Commands that take a list of IDs can use ranges to help. Individual IDs must be
separated by a `,` (no space allowed) and ranges can be expressed with either
`-` or `..`.

### Ranges of IPs

There are several ways to specify ranges of IP addresses that can be mixed
together. The first way is a list of IPs separated by just a ` ` (ASCII space),
with an optional `,`. The next way is two complete IP addresses in the form of
`BEGINNING_ADDRESS-END_ADDRESS` like `127.0.1.44-127.0.2.33`. CIDR
specifications may also be used, however the whole address must be given to
Metasploit like `127.0.0.0/8` and not `127/8`, contrary to the RFC.
Additionally, a netmask can be used in conjunction with a domain name to
dynamically resolve which block to target. All these methods work for both IPv4
and IPv6 addresses. IPv4 addresses can also be specified with special octet
ranges from the [NMAP target
specification](https://nmap.org/book/man-target-specification.html)
```



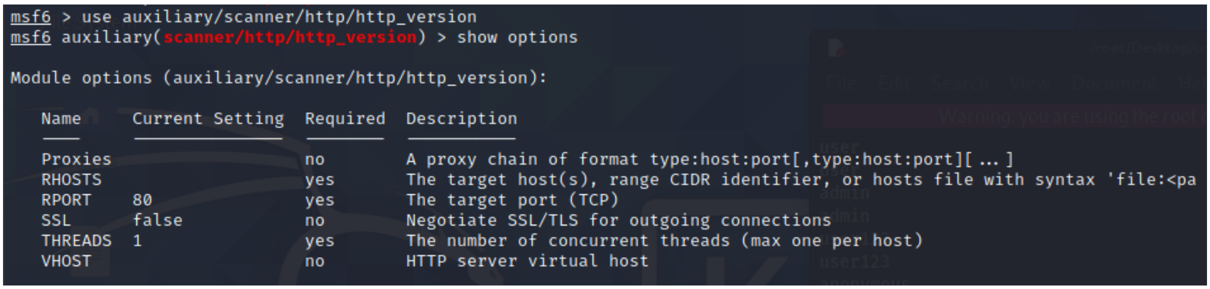
```
### Examples

Terminate the first sessions:

    sessions -k 1

Stop some extra running jobs:

    jobs -k 2-6,7,8,11..15

Check a set of IP addresses:

    check 127.168.0.0/16, 127.0.0-2.1-4,15 127.0.0.255

Target a set of IPv6 hosts:

    set RHOSTS fe80::3990:0000/110, ::1-::f0f0
```



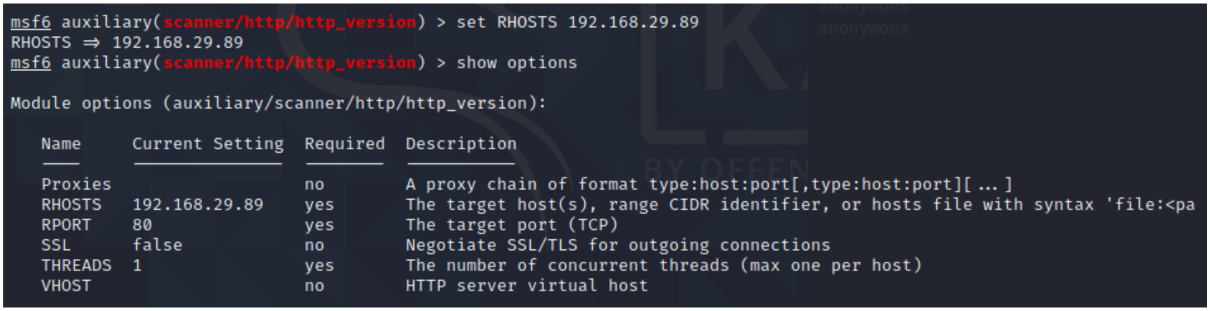
```
Target a block from a resolved domain name:

    set RHOSTS www.example.test/24
```

HTTP:

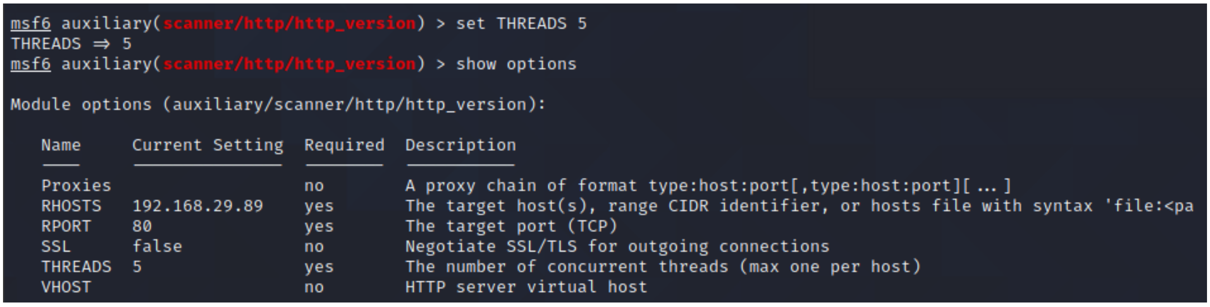
The **http_version** scanner will scan a range of hosts and determine the web server version that is running on them.



```
msf6 > use auxiliary/scanner/http/http_version
msf6 auxiliary(scanner/http/http_version) > show options

Module options (auxiliary/scanner/http/http_version):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                    yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<pa
   RPORT    80               yes       The target port (TCP)
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   THREADS  1                yes       The number of concurrent threads (max one per host)
   VHOST                     no        HTTP server virtual host
```

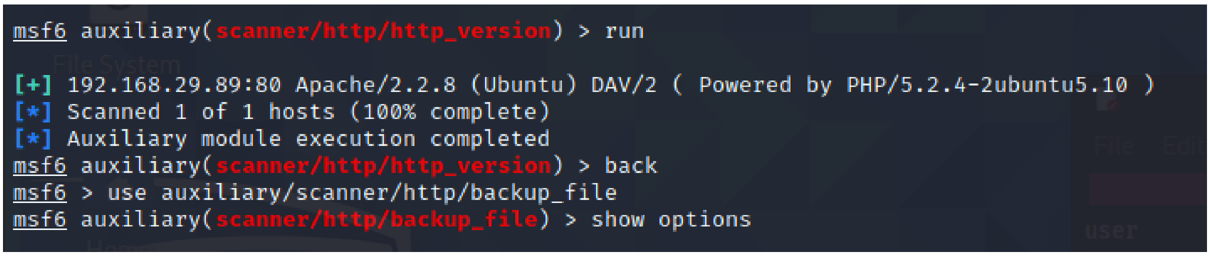


```
msf6 auxiliary(scanner/http/http_version) > set RHOSTS 192.168.29.89
RHOSTS => 192.168.29.89
msf6 auxiliary(scanner/http/http_version) > show options

Module options (auxiliary/scanner/http/http_version):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS   192.168.29.89    yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<pa
   RPORT    80               yes       The target port (TCP)
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   THREADS  1                yes       The number of concurrent threads (max one per host)
   VHOST                     no        HTTP server virtual host
```

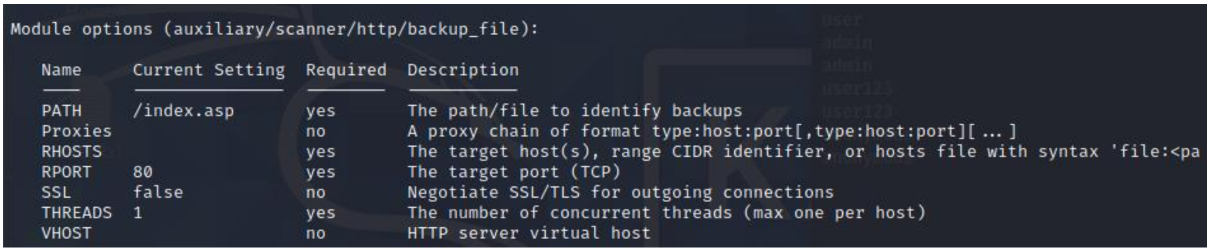


```
msf6 auxiliary(scanner/http/http_version) > set THREADS 5
THREADS => 5
msf6 auxiliary(scanner/http/http_version) > show options

Module options (auxiliary/scanner/http/http_version):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS   192.168.29.89    yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<pa
   RPORT    80               yes       The target port (TCP)
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   THREADS  5                yes       The number of concurrent threads (max one per host)
   VHOST                     no        HTTP server virtual host
```

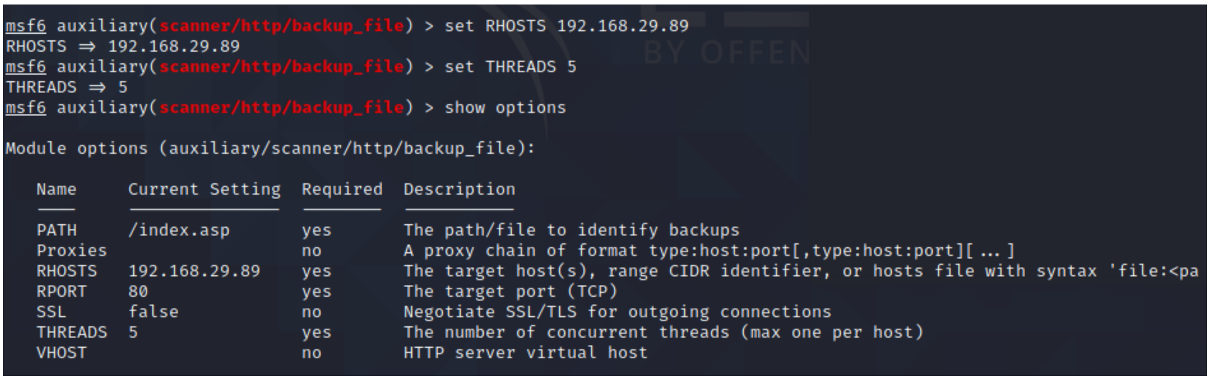
To run the scan, we set the RHOSTS and THREADS values and let it run.



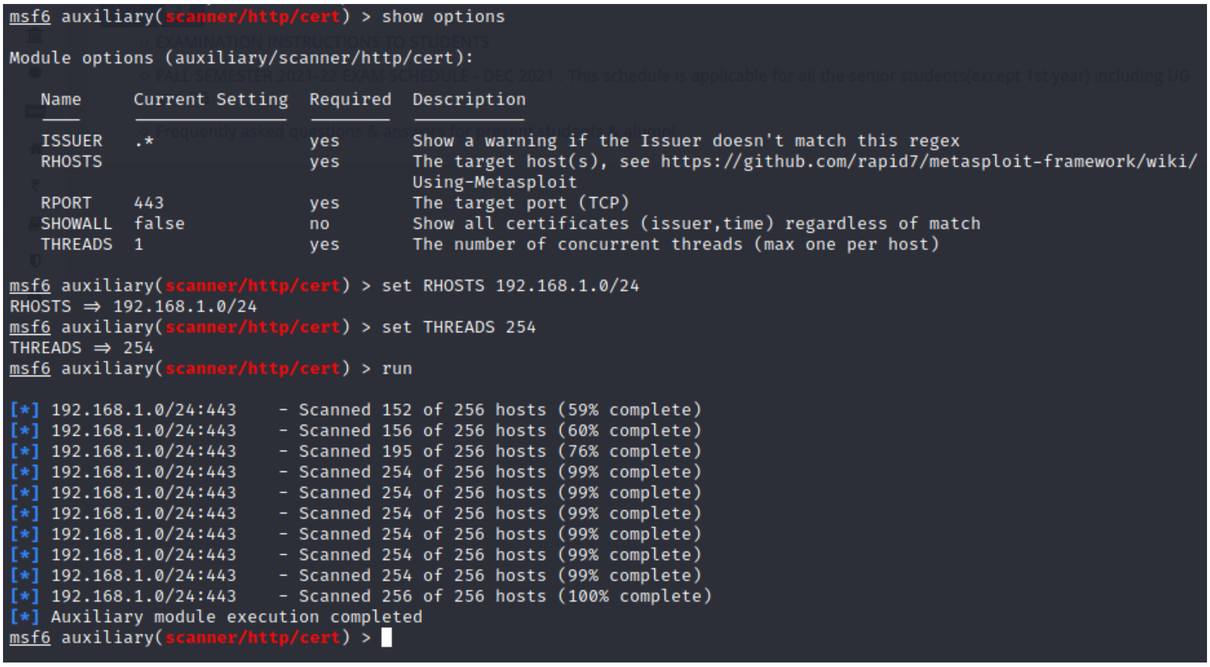
```
msf6 auxiliary(scanner/http/http_version) > run

[+] 192.168.29.89:80 Apache/2.2.8 (Ubuntu) DAV/2 ( Powered by PHP/5.2.4-2ubuntu5.10 )
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/http/http_version) > back
msf6 > use auxiliary/scanner/http/backup_file
msf6 auxiliary(scanner/http/backup_file) > show options
```



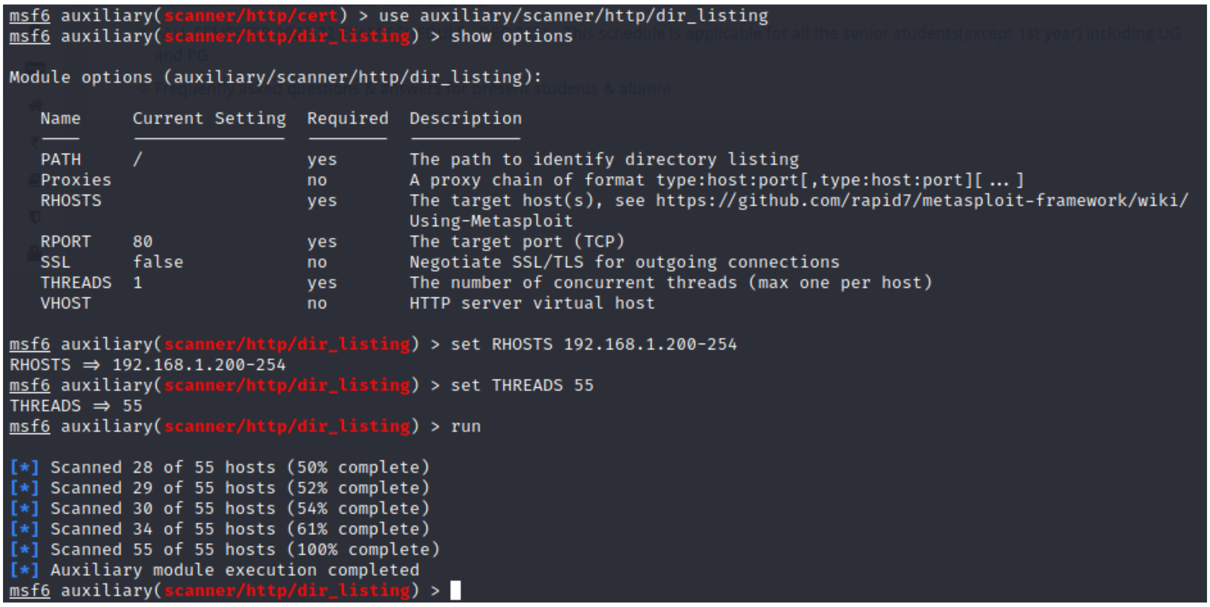
```
Module options (auxiliary/scanner/http/backup_file):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   PATH     /index.asp       yes       The path/file to identify backups
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS                    yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<pa
   RPORT    80               yes       The target port (TCP)
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   THREADS  1                yes       The number of concurrent threads (max one per host)
   VHOST                     no        HTTP server virtual host
```



```
msf6 auxiliary(scanner/http/backup_file) > set RHOSTS 192.168.29.89
RHOSTS ⇒ 192.168.29.89
msf6 auxiliary(scanner/http/backup_file) > set THREADS 5
THREADS ⇒ 5
msf6 auxiliary(scanner/http/backup_file) > show options

Module options (auxiliary/scanner/http/backup_file):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   PATH     /index.asp       yes       The path/file to identify backups
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS   192.168.29.89    yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<pa
   RPORT    80               yes       The target port (TCP)
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   THREADS  5                yes       The number of concurrent threads (max one per host)
   VHOST                     no        HTTP server virtual host
```



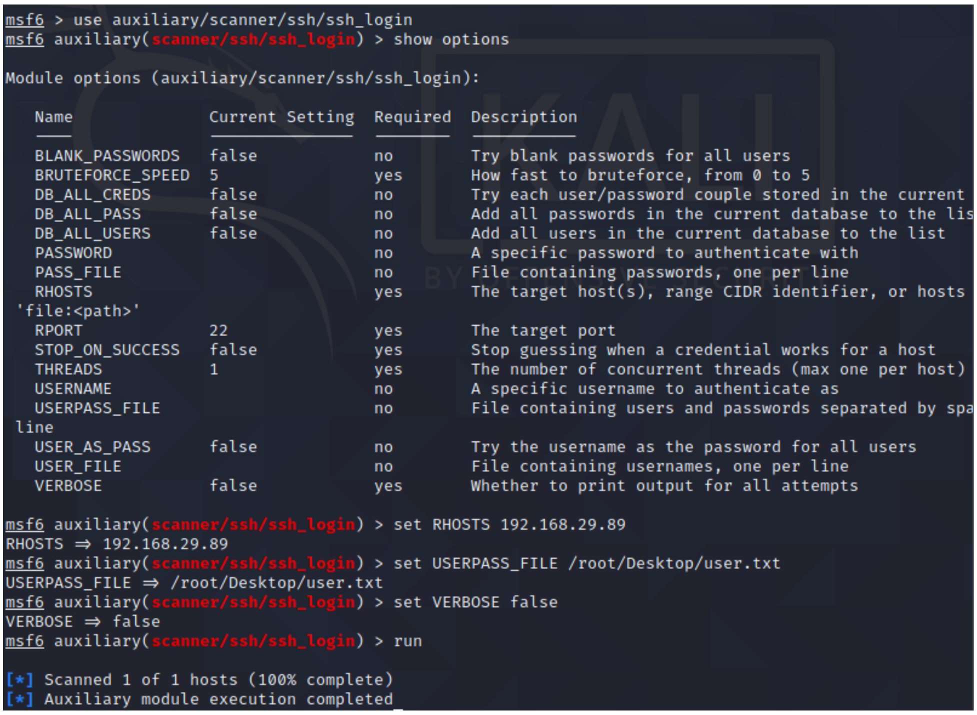
```
msf6 auxiliary(scanner/http/cert) > show options

Module options (auxiliary/scanner/http/cert):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   ISSUER   .*               yes       Show a warning if the Issuer doesn't match this regex
   RHOSTS                    yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/
                                       Using-Metasploit
   RPORT    443              yes       The target port (TCP)
   SHOWALL  false            no        Show all certificates (issuer,time) regardless of match
   THREADS  1                yes       The number of concurrent threads (max one per host)

msf6 auxiliary(scanner/http/cert) > set RHOSTS 192.168.1.0/24
RHOSTS ⇒ 192.168.1.0/24
msf6 auxiliary(scanner/http/cert) > set THREADS 254
THREADS ⇒ 254
msf6 auxiliary(scanner/http/cert) > run

[*] 192.168.1.0/24:443    - Scanned 152 of 256 hosts (59% complete)
[*] 192.168.1.0/24:443    - Scanned 156 of 256 hosts (60% complete)
[*] 192.168.1.0/24:443    - Scanned 195 of 256 hosts (76% complete)
[*] 192.168.1.0/24:443    - Scanned 254 of 256 hosts (99% complete)
[*] 192.168.1.0/24:443    - Scanned 254 of 256 hosts (99% complete)
[*] 192.168.1.0/24:443    - Scanned 254 of 256 hosts (99% complete)
[*] 192.168.1.0/24:443    - Scanned 254 of 256 hosts (99% complete)
[*] 192.168.1.0/24:443    - Scanned 254 of 256 hosts (99% complete)
[*] 192.168.1.0/24:443    - Scanned 254 of 256 hosts (99% complete)
[*] 192.168.1.0/24:443    - Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/http/cert) > █
```



```
msf6 auxiliary(scanner/http/cert) > use auxiliary/scanner/http/dir_listing
msf6 auxiliary(scanner/http/dir_listing) > show options

Module options (auxiliary/scanner/http/dir_listing):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   PATH     /                yes       The path to identify directory listing
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                    yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/
                                       Using-Metasploit
   RPORT    80               yes       The target port (TCP)
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   THREADS  1                yes       The number of concurrent threads (max one per host)
   VHOST                     no        HTTP server virtual host

msf6 auxiliary(scanner/http/dir_listing) > set RHOSTS 192.168.1.200-254
RHOSTS => 192.168.1.200-254
msf6 auxiliary(scanner/http/dir_listing) > set THREADS 55
THREADS => 55
msf6 auxiliary(scanner/http/dir_listing) > run

[*] Scanned 28 of 55 hosts (50% complete)
[*] Scanned 29 of 55 hosts (52% complete)
[*] Scanned 30 of 55 hosts (54% complete)
[*] Scanned 34 of 55 hosts (61% complete)
[*] Scanned 55 of 55 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/http/dir_listing) >
```

SECURE SHELL

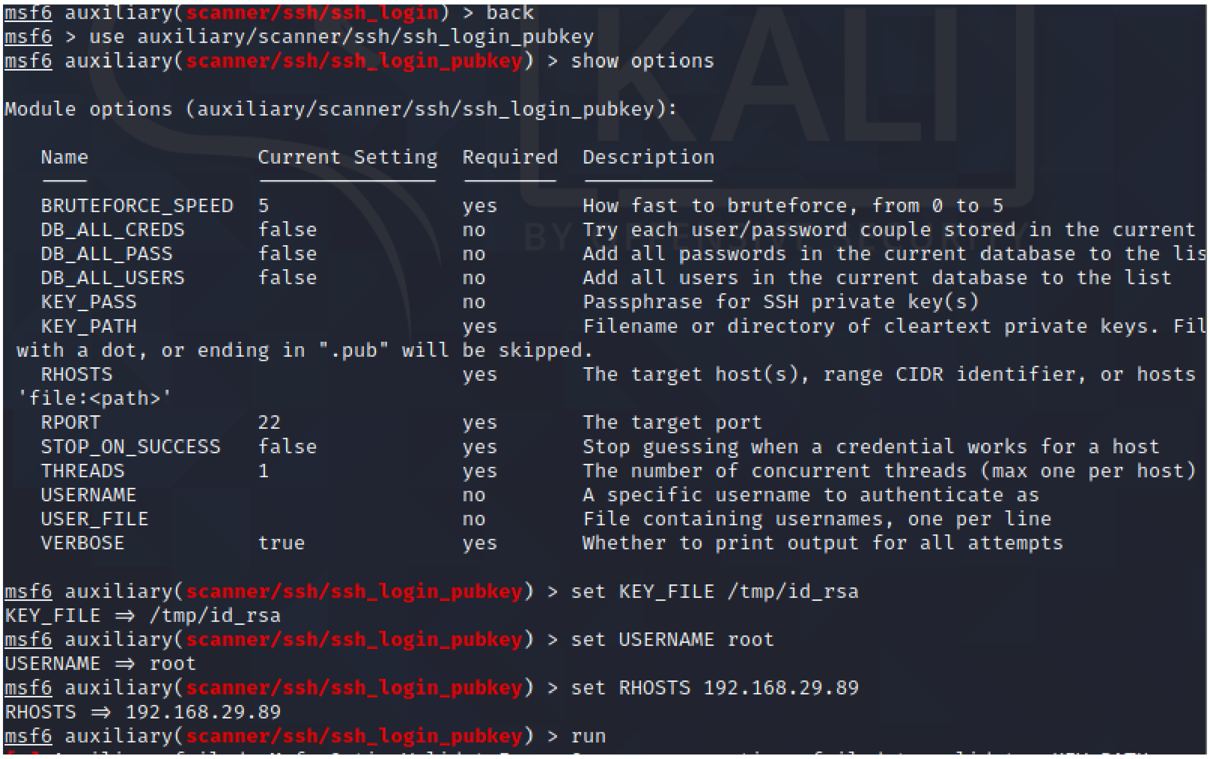
The **ssh_login** module is quite versatile in that it can not only test a set of credentials across a range of IP addresses, but it can also perform brute force login attempts. We will pass a file to the module containing usernames and passwords separated by a space as shown below. Next, we load up the scanner module in Metasploit and set USERPASS_FILE to point to our list of credentials to attempt.



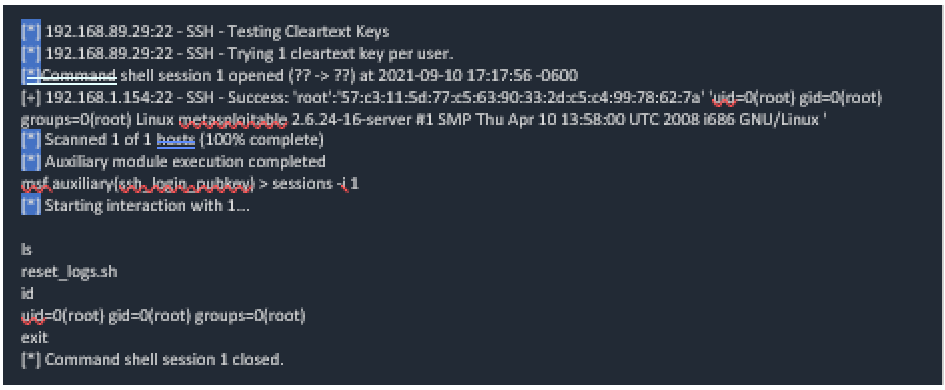
```
msf6 > use auxiliary/scanner/ssh/ssh_login
msf6 auxiliary(scanner/ssh/ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_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
   DB_ALL_CREDS      false            no        Try each user/password couple stored in the current
   DB_ALL_PASS       false            no        Add all passwords in the current database to the lis
   DB_ALL_USERS      false            no        Add all users in the current database to the list
   PASSWORD                           no        A specific password to authenticate with
   PASS_FILE                          no        File containing passwords, one per line
   RHOSTS                             yes       The target host(s), range CIDR identifier, or hosts
 'file:<path>'
   RPORT             22               yes       The target port
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads (max one per host)
   USERNAME                           no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords separated by spa
 line
   USER_AS_PASS      false            no        Try the username as the password for all users
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE           false            yes       Whether to print output for all attempts

msf6 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 192.168.29.89
RHOSTS => 192.168.29.89
msf6 auxiliary(scanner/ssh/ssh_login) > set USERPASS_FILE /root/Desktop/user.txt
USERPASS_FILE => /root/Desktop/user.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set VERBOSE false
VERBOSE => false
msf6 auxiliary(scanner/ssh/ssh_login) > run

[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

With everything ready to go, we run the module.

Using public key authentication for SSH is highly regarded as being far more secure than using usernames and passwords to authenticate. The caveat to this is that if the private key portion of the key pair is not kept secure, the security of the configuration is thrown right out the window. If, during an engagement, you get access to a private SSH key, you can use the **ssh_login_pubkey** module to attempt to login across a range of devices.

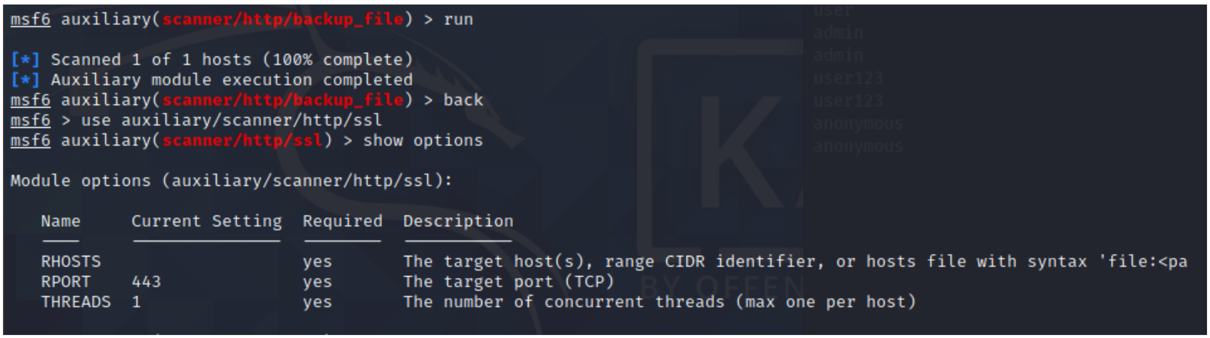


```
msf6 auxiliary(scanner/ssh/ssh_login) > back
msf6 > use auxiliary/scanner/ssh/ssh_login_pubkey
msf6 auxiliary(scanner/ssh/ssh_login_pubkey) > show options

Module options (auxiliary/scanner/ssh/ssh_login_pubkey):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false            no        Try each user/password couple stored in the current
   DB_ALL_PASS       false            no        Add all passwords in the current database to the lis
   DB_ALL_USERS      false            no        Add all users in the current database to the list
   KEY_PASS                           no        Passphrase for SSH private key(s)
   KEY_PATH                           yes       Filename or directory of cleartext private keys. Fil
 with a dot, or ending in ".pub" will be skipped.
   RHOSTS                             yes       The target host(s), range CIDR identifier, or hosts
 'file:<path>'
   RPORT             22               yes       The target port
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads (max one per host)
   USERNAME                           no        A specific username to authenticate as
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE           true             yes       Whether to print output for all attempts

msf6 auxiliary(scanner/ssh/ssh_login_pubkey) > set KEY_FILE /tmp/id_rsa
KEY_FILE ⇒ /tmp/id_rsa
msf6 auxiliary(scanner/ssh/ssh_login_pubkey) > set USERNAME root
USERNAME ⇒ root
msf6 auxiliary(scanner/ssh/ssh_login_pubkey) > set RHOSTS 192.168.29.89
RHOSTS ⇒ 192.168.29.89
msf6 auxiliary(scanner/ssh/ssh_login_pubkey) > run
```

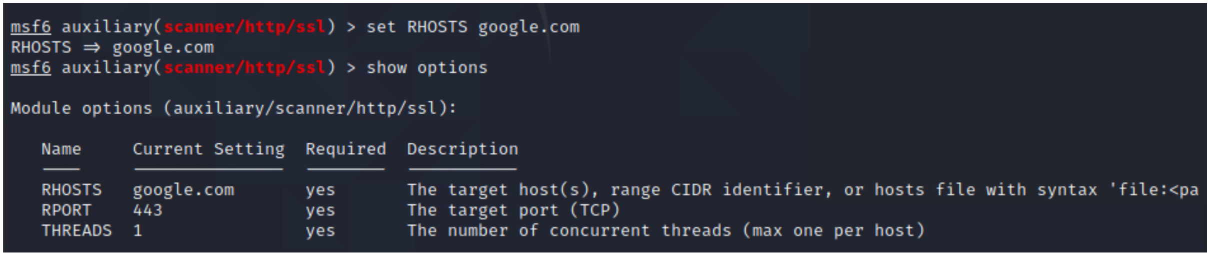


```
[*] 192.168.89.29:22 - SSH - Testing Cleartext Keys
[*] 192.168.89.29:22 - SSH - Trying 1 cleartext key per user.
[*] Command shell session 1 opened (?? -> ??) at 2021-09-10 17:17:56 -0600
[+] 192.168.1.154:22 - SSH - Success: 'root':'57:c3:11:5d:77:c5:63:90:33:2d:c5:c4:99:78:62:7a' 'uid=0(root) gid=0(root)
groups=0(root) Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux '
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(ssh_login_pubkey) > sessions -i 1
[*] Starting interaction with 1...

ls
reset_logs.sh
id
uid=0(root) gid=0(root) groups=0(root)
exit
[*] Command shell session 1 closed.
```

The **ssl** module queries a host or range of hosts and pull the SSL certificate information if present.

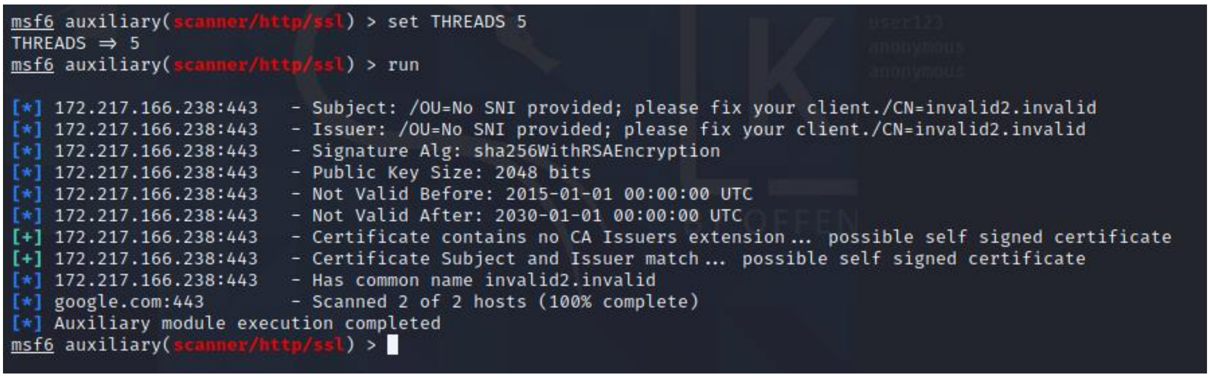


```
msf6 auxiliary(scanner/http/backup_file) > run

[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/http/backup_file) > back
msf6 > use auxiliary/scanner/http/ssl
msf6 auxiliary(scanner/http/ssl) > show options

Module options (auxiliary/scanner/http/ssl):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS                    yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<pa
   RPORT    443              yes       The target port (TCP)
   THREADS  1                yes       The number of concurrent threads (max one per host)
```



```
msf6 auxiliary(scanner/http/ssl) > set RHOSTS google.com
RHOSTS ⇒ google.com
msf6 auxiliary(scanner/http/ssl) > show options

Module options (auxiliary/scanner/http/ssl):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS   google.com       yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<pa
   RPORT    443              yes       The target port (TCP)
   THREADS  1                yes       The number of concurrent threads (max one per host)
```

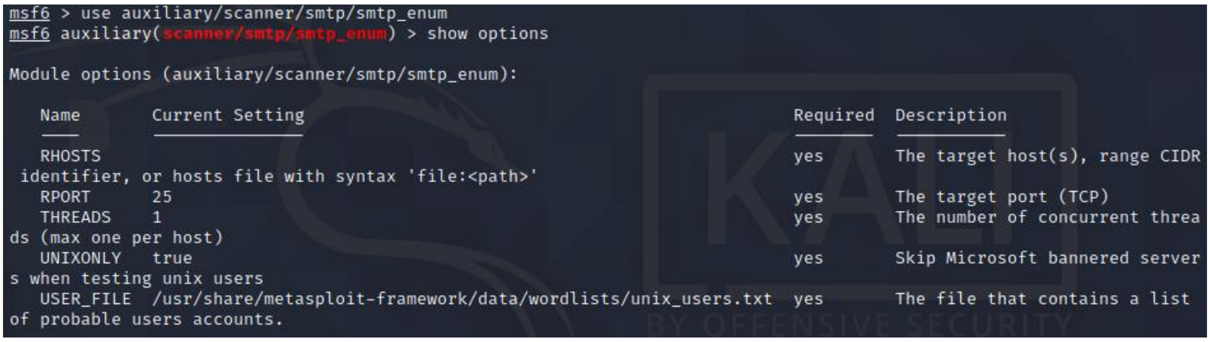
To configure the module, we set our RHOSTS and THREADS values and let it run.



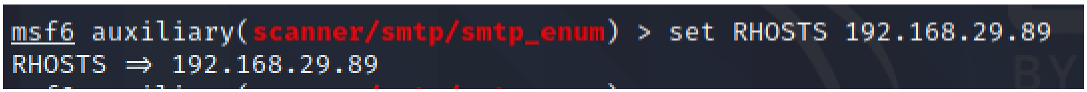
```
msf6 auxiliary(scanner/http/ssl) > set THREADS 5
THREADS ⇒ 5
msf6 auxiliary(scanner/http/ssl) > run

[*] 172.217.166.238:443   - Subject: /OU=No SNI provided; please fix your client./CN=invalid2.invalid
[*] 172.217.166.238:443   - Issuer: /OU=No SNI provided; please fix your client./CN=invalid2.invalid
[*] 172.217.166.238:443   - Signature Alg: sha256WithRSAEncryption
[*] 172.217.166.238:443   - Public Key Size: 2048 bits
[*] 172.217.166.238:443   - Not Valid Before: 2015-01-01 00:00:00 UTC
[*] 172.217.166.238:443   - Not Valid After: 2030-01-01 00:00:00 UTC
[+] 172.217.166.238:443   - Certificate contains no CA Issuers extension ... possible self signed certificate
[+] 172.217.166.238:443   - Certificate Subject and Issuer match ... possible self signed certificate
[*] 172.217.166.238:443   - Has common name invalid2.invalid
[*] google.com:443        - Scanned 2 of 2 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/http/ssl) > █
```

SMTP:

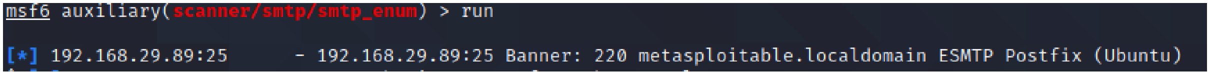
The SMTP Enumeration module will connect to a given mail server and use a wordlist to enumerate users that are present on the remote system.



```
msf6 > use auxiliary/scanner/smtp/smtp_enum
msf6 auxiliary(scanner/smtp/smtp_enum) > show options

Module options (auxiliary/scanner/smtp/smtp_enum):

   Name       Current Setting                                             Required  Description
   ----       ---------------                                             --------  -----------
   RHOSTS                                                                 yes       The target host(s), range CIDR
 identifier, or hosts file with syntax 'file:<path>'
   RPORT      25                                                          yes       The target port (TCP)
   THREADS    1                                                           yes       The number of concurrent threa
ds (max one per host)
   UNIXONLY   true                                                        yes       Skip Microsoft bannered server
s when testing unix users
   USER_FILE  /usr/share/metasploit-framework/data/wordlists/unix_users.txt  yes       The file that contains a list
of probable users accounts.
```



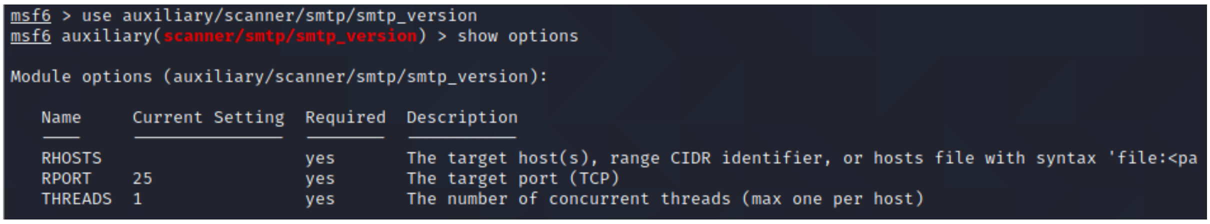
```
msf6 auxiliary(scanner/smtp/smtp_enum) > set RHOSTS 192.168.29.89
RHOSTS ⇒ 192.168.29.89
```



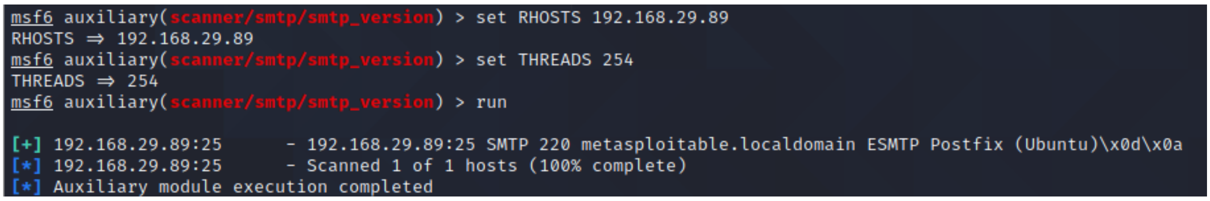
```
msf6 auxiliary(scanner/smtp/smtp_enum) > run

[*] 192.168.29.89:25      - 192.168.29.89:25 Banner: 220 metasploitable.localdomain ESMTP Postfix (Ubuntu)
```

Since the email username and system username are frequently the same, you can now use any enumerated users for further logon attempts against other network services.

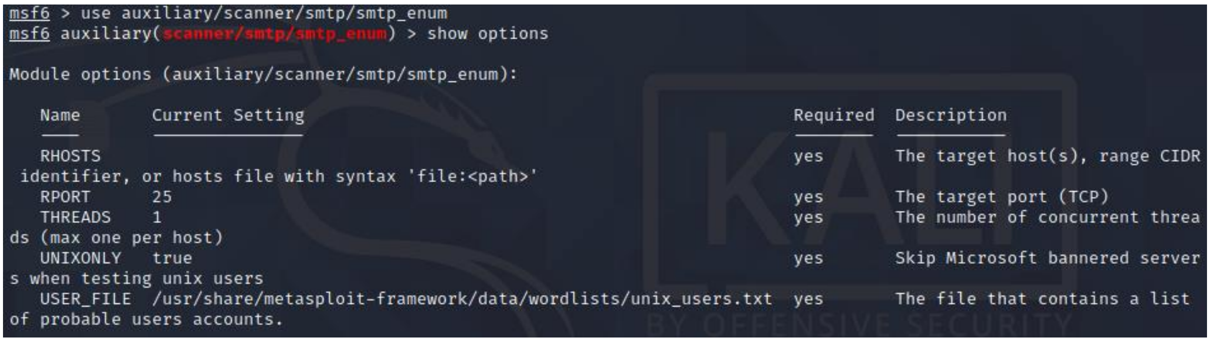
Poorly configured or vulnerable mail servers can often provide an initial foothold into a network but prior to launching an attack, we want to fingerprint the server to make our targeting as precise as possible. The **smtp_version** module, as its name implies, will scan a range of IP addresses and determine the version of any mail servers it encounters.



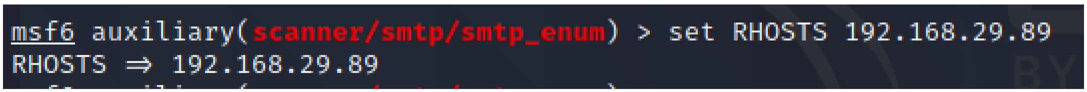
```
msf6 > use auxiliary/scanner/smtp/smtp_version
msf6 auxiliary(scanner/smtp/smtp_version) > show options

Module options (auxiliary/scanner/smtp/smtp_version):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS                    yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<pa
   RPORT    25               yes       The target port (TCP)
   THREADS  1                yes       The number of concurrent threads (max one per host)
```



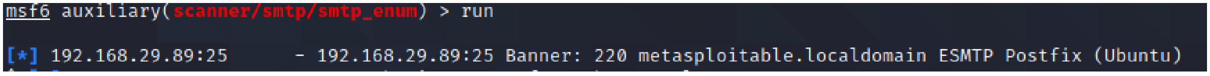
```
msf6 auxiliary(scanner/smtp/smtp_version) > set RHOSTS 192.168.29.89
RHOSTS ⇒ 192.168.29.89
msf6 auxiliary(scanner/smtp/smtp_version) > set THREADS 254
THREADS ⇒ 254
msf6 auxiliary(scanner/smtp/smtp_version) > run

[+] 192.168.29.89:25      - 192.168.29.89:25 SMTP 220 metasploitable.localdomain ESMTP Postfix (Ubuntu)\x0d\x0a
[*] 192.168.29.89:25      - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```



```
msf6 > use auxiliary/scanner/smtp/smtp_enum
msf6 auxiliary(scanner/smtp/smtp_enum) > show options

Module options (auxiliary/scanner/smtp/smtp_enum):

   Name       Current Setting                                          Required  Description
   ----       ---------------                                          --------  -----------
   RHOSTS                                                              yes       The target host(s), range CIDR
 identifier, or hosts file with syntax 'file:<path>'
   RPORT      25                                                       yes       The target port (TCP)
   THREADS    1                                                        yes       The number of concurrent threa
ds (max one per host)
   UNIXONLY   true                                                     yes       Skip Microsoft bannered server
s when testing unix users
   USER_FILE  /usr/share/metasploit-framework/data/wordlists/unix_users.txt  yes       The file that contains a list
of probable users accounts.
```



```
msf6 auxiliary(scanner/smtp/smtp_enum) > set RHOSTS 192.168.29.89
RHOSTS ⇒ 192.168.29.89
```



```
msf6 auxiliary(scanner/smtp/smtp_enum) > run

[*] 192.168.29.89:25      - 192.168.29.89:25 Banner: 220 metasploitable.localdomain ESMTP Postfix (Ubuntu)
```