PCAP generator

Generation of pcap files using python and docker.

Installation

Docker, docker compose and docker swarm is needed to run scenarios.

1. Install Docker on Linux: CentOS, Debian, Fedora, or Ubuntu.
2. Post installation steps do run Docker as non sudo user: https://docs.docker.com/install/linux/linux-postinstall/
3. Install Docker Compose.
4. *Optional* for Docker Swarm on multiple computers - init cluster.

Project structure

There are several folder in the project, below is a description of each.

`output` directory

Here are stored outputs from container like tcpdump, http crawler logs, selenium tests ouptput, etc. Use the directory for any files needed to be generated from containers. Include .gitignore file to write directory to git repository (but not content).

`runs` directory

Contains settings for sambla clients and server. This directory will be deleted in next release, do not use this directory.

`scenarios` directory

Here is the complete list of all implemented scenarios. In each subdirectory is usually included documentation (README.md) file, bash script for running scenario and docker-compose.yml file.

`sides` directory

This directory contains test files for Selenium IDE. Also this directory will be moved to src directory in the future.

`src` directory

To this directory belongs all source codes for containers. All containers have its own subdirectory.

`support` directory

Directory for another files which are not relevant for running containers. Put here web pages, images, or any other files.

Other files in the root directory

`generate_pdf.sh` is a bash script which generates `manual.pdf` file from this file and README.md in `scenarios` subdirectories. It uses Pandoc which is a universal document converter. `pcap_template.latex` is latex template for the output pdf.

Usage

All scenarios are in the `scenarios` directory. Scenarios can be run by `cd` to the directory and run bash script `bash run.sh`. The script runs the docker-compose file, waits 10 minutes and then ps containers and kills them.

Manual way to run a scenario:

```
docker stack deploy --compose-file docker-compose.yml webservice
```

Monitor running containers:

```
docker stack ps  --no-trunc webservice
```

Show logs for given service:

```
docker service logs webservice_[container_name]
```

Terminate running scenario:

```
docker stack rm webservice
```

Note: in all scenarios is `webservice` name of service which containers run under.

Another docker useful commands

Create new tag (default is latest):

```
docker tag mylocalimage:latest username/reponame:tag
```

Example:

```
docker tag udfb:nginx-tcpdump udfb/nginx-tcpdump
```

Add new version to the Docker Hub:

```
docker push username/reponame:tag
```

Example:

```
docker push udfb/nginx-tcpdump
```

Shows log for given service:

```
docker service logs webservice_http_fuzz
```

Generate pdf version of this file:

https://www.markdowntopdf.com/

or directly from github,

or just install Pandoc and run generate_pdf.sh to generate whole documentation:

```
bash generate_pdf.sh
```

DAVTest

Information obtained from: Kali Tool page

DAVTest tests WebDAV enabled servers by uploading test executable files, and then (optionally) uploading files which allow for command execution or other actions directly on the target. It is meant for penetration testers to quickly and easily determine if enabled DAV services are exploitable.

DAVTest supports: - Automatically send exploit files - Automatic randomization of directory to help hide files - Send text files and try MOVE to executable name - Basic and Digest authorization - Automatic clean-up of uploaded files - Send an arbitrary file

Source: https://code.google.com/p/davtest/

- Author: Sunera, LLC.
- License: GPLv3

```
root@kali:~# davtest

ERROR: Missing -url

/usr/bin/davtest -url <url> [options]

 -auth+     Authorization (user:password)
 -cleanup   delete everything uploaded when done
 -directory+    postfix portion of directory to create
 -debug+    DAV debug level 1-3 (2 & 3 log req/resp to
            /tmp/perldav_debug.txt)
 -move      PUT text files then MOVE to executable
 -nocreate  don't create a directory
 -quiet     only print out summary
 -rand+     use this instead of a random string for filenames
 -sendbd+   send backdoors:
            auto - for any succeeded test
            ext - extension matching file name(s) in
                  backdoors/ dir
 -uploadfile+   upload this file (requires -uploadloc)
 -uploadloc+    upload file to this location/name
                (requires -uploadfile)
 -url+      url of DAV location

Example: /usr/bin/davtest -url http://localhost/davdir
```

Scan the given WebDAV server (-url http://192.168.1.209):

```
root@kali:~# davtest -url http://192.168.1.209
********************************************************
 Testing DAV connection
OPEN        SUCCEED:        http://192.168.1.209
********************************************************
NOTE    Random string for this session: B0yG9nhdFS8gox
********************************************************
 Creating directory
MKCOL SUCCEED: Created http://192.168.1.209/DavTestDir_B0yG9nhdFS8gox
********************************************************
 Sending test files
PUT asp FAIL
PUT cgi FAIL
PUT txt SUCCEED:    http://192.168.1.209/DavTestDir_B0yG9nhdFS8gox/
                    davtest_B0yG9nhdFS8gox.txt
PUT pl  SUCCEED:    http://192.168.1.209/DavTestDir_B0yG9nhdFS8gox/
                    davtest_B0yG9nhdFS8gox.pl
PUT jsp SUCCEED:    http://192.168.1.209/DavTestDir_B0yG9nhdFS8gox/
                    davtest_B0yG9nhdFS8gox.jsp
PUT cfm SUCCEED:    http://192.168.1.209/DavTestDir_B0yG9nhdFS8gox/
                    davtest_B0yG9nhdFS8gox.cfm
PUT aspx    FAIL
PUT jhtml   SUCCEED:    http://192.168.1.209/DavTestDir_B0yG9nhdFS8gox/
                        davtest_B0yG9nhdFS8gox.jhtml
PUT php SUCCEED:    http://192.168.1.209/DavTestDir_B0yG9nhdFS8gox/
                    davtest_B0yG9nhdFS8gox.php
PUT html    SUCCEED:    http://192.168.1.209/DavTestDir_B0yG9nhdFS8gox/
                        davtest_B0yG9nhdFS8gox.html
PUT shtml   FAIL
********************************************************
```

```
 Checking for test file execution
EXEC    txt SUCCEED:    http://192.168.1.209/DavTestDir_B0yG9nhdFS8gox/
                        davtest_B0yG9nhdFS8gox.txt
EXEC    pl  FAIL
EXEC    jsp FAIL
EXEC    cfm FAIL
EXEC    jhtml   FAIL
EXEC    php FAIL
EXEC    html    SUCCEED:    http://192.168.1.209/DavTestDir_
                            B0yG9nhdFS8gox/davtest_B0yG9nhdFS8gox.html

*****************************************************
/usr/bin/davtest Summary:
Created: http://192.168.1.209/DavTestDir_B0yG9nhdFS8gox
PUT File: http://192.168.1.209/DavTestDir_B0yG9nhdFS8gox/
          davtest_B0yG9nhdFS8gox.txt
PUT File: http://192.168.1.209/DavTestDir_B0yG9nhdFS8gox/
          davtest_B0yG9nhdFS8gox.pl
PUT File: http://192.168.1.209/DavTestDir_B0yG9nhdFS8gox/
          davtest_B0yG9nhdFS8gox.jsp
PUT File: http://192.168.1.209/DavTestDir_B0yG9nhdFS8gox/
          davtest_B0yG9nhdFS8gox.cfm
PUT File: http://192.168.1.209/DavTestDir_B0yG9nhdFS8gox/
          davtest_B0yG9nhdFS8gox.jhtml
PUT File: http://192.168.1.209/DavTestDir_B0yG9nhdFS8gox/
          davtest_B0yG9nhdFS8gox.php
PUT File: http://192.168.1.209/DavTestDir_B0yG9nhdFS8gox/
          davtest_B0yG9nhdFS8gox.html
Executes: http://192.168.1.209/DavTestDir_B0yG9nhdFS8gox/
          davtest_B0yG9nhdFS8gox.txt
Executes: http://192.168.1.209/DavTestDir_B0yG9nhdFS8gox/
          davtest_B0yG9nhdFS8gox.html
```

fimap

Information obtained from: Kali Tool page

fimap is a little python tool which can find, prepare, audit, exploit and even google automatically for local and remote file inclusion bugs in webapps. fimap should be something like sqlmap just for LFI/RFI bugs instead of SQL injection. It's currently under heavy development but it's usable.

Source: https://tha-imax.de/git/root/fimap

- Author: Iman Karim
- License: GPLv2

```
root@kali:~# fimap -h
fimap v.1.00_svn (My life for Aiur)
:: Automatic LFI/RFI scanner and exploiter
:: by Iman Karim (fimap.dev@gmail.com)

Usage: fimap [options]
## Operating Modes:
   -s , --single      Mode to scan a single URL for FI errors.
                      Needs URL (-u). This mode is the default.
   -m , --mass        Mode for mass scanning. Will check every URL
                      from a given list (-l) for FI errors.
   -g , --google      Mode to use Google to aquire URLs.
                      Needs a query (-q) as google search query.
   -B , --bing        Use bing to get URLs.
                      Needs a query (-q) as bing search query.
                      Also needs a Bing APIKey (--bingkey)
   -H , --harvest     Mode to harvest a URL recursivly for new URLs.
                      Needs a root url (-u) to start crawling there.
                      Also needs (-w) to write a URL list for mass mode.
   -4 , --autoawesome With the AutoAwesome mode fimap will fetch all
                      forms and headers found on the site you defined
                      and tries to find file inclusion bugs thru them. Needs an
                      URL (-u).
## Techniques:
   -b , --enable-blind  Enables blind FI-Bug testing when no error messages are printed.
                        Note that this mode will cause lots of requests compared to the
                        default method. Can be used with -s, -m or -g.
   -D , --dot-truncation   Enables dot truncation technique to get rid of the suffix
                           if the default mode (nullbyte poison) failed. This mode
                           can cause tons of requests depending how you configure it.
                           By default this mode only tests windows servers.
                           Can be used with -s, -m or -g. Experimental.
   -M , --multiply-term=X  Multiply terminal symbols like '.' and '/' in the path by X.
## Variables:
  -u , --url=URL               The URL you want to test.
                               Needed in single mode (-s).
  -l , --list=LIST             The URL-LIST you want to test.
                               Needed in mass mode (-m).
  -q , --query=QUERY           The Google Search QUERY.
                               Example: 'inurl:include.php'
                               Needed in Google Mode (-g)
       --bingkey=APIKEY        This is your the Bing APIKey. You have to set this when
                               you want to use the BingScanner (-B).
       --skip-pages=X          Skip the first X pages from the Googlescanner.
  -p , --pages=COUNT           Define the COUNT of pages to search (-g).
                               Default is 10.
       --results=COUNT         The count of results the Googlescanner should get per
                               page. Possible values: 10, 25, 50 or 100(default).
       --googlesleep=TIME      The time in seconds the Googlescanner should wait befor
                               each request to google. fimap will count the time between
                               two requests and will sleep if it's needed to reach your
                               cooldown. Default is 5.
  -w , --write=LIST            The LIST which will be written if you have choosen
                               harvest mode (-H). This file will be opened in
                               APPEND mode.
  -d , --depth=CRAWLDEPTH      The CRAWLDEPTH (recurse level) you want to crawl your
                               target site in harvest mode (-H). Default is 1.
  -P , --post=POSTDATA         The POSTDATA you want to send. All variables inside
                               will also be scanned for file inclusion bugs.
       --cookie=COOKIES        Define the cookie which should be send with each request.
                               Also the cookies will be scanned for file inclusion bugs.
```

```
                                Concatenate multiple cookies with the ';' character.
      --ttl=SECONDS             Define the TTL (in seconds) for requests. Default is
                                30 seconds.
      --no-auto-detect          Use this switch if you don't want to let fimap
                                automaticly detect the target language in blind-mode.
                                In that case you will get some options you can choose
                                if fimap isn't sure which lang it is.
      --bmin=BLIND_MIN          Define here the minimum count of directories fimap should
                                walk thru in blind mode. The default number is defined in
                                the generic.xml
      --bmax=BLIND_MAX          Define here the maximum count of directories fimap should
                                walk thru.
      --dot-trunc-min=700       The count of dots to begin with in dot-truncation mode.
      --dot-trunc-max=2000      The count of dots to end with in dot-truncation mode.
      --dot-trunc-step=50       The step size for each round in dot-truncation mode.
      --dot-trunc-ratio=0.095   The maximum ratio to detect if dot truncation was
                                successfull.
      --dot-trunc-also-unix     Use this if dot-truncation should also be tested on unix
                                servers.
      --force-os=OS             Forces fimap to test only files for the OS.
                                OS can be 'linux' or 'windows'
# Attack Kit:
  -x, --exploit             Starts an interactive session where you can
                            select a target and do some action.
  -X                        Same as -x but also shows not exploitable which might can be
                            hax0red with plugins.
  -T, --tab-complete        Enables TAB-Completation in exploit mode. Needs readline module.
                            Use this if you want to be able to tab-complete thru remote
                            files\dirs. Eats an extra request for every 'cd' command.
      --x-host=HOSTNAME     The host to use exploits on. fimap won't prompt you for the
                            domain in exploit mode if you set this value.
      --x-vuln=VULNNUMBER   The vulnerability ID you want to use. It's the same number you
                            type into the exploit mode where you choose the vulnerable
                            script.
      --x-cmd=CMD           The CMD you want to execute on the vulnerable system. Use
                            this parameter more than once to execute commands one after
                            another. Remember that each command opens a new shell and
                            closes it after execution.
# Disguise Kit:
  -A , --user-agent=UA       The User-Agent which should be sent.
       --http-proxy=PROXY    Setup your proxy with this option. But read this facts:
                               * The googlescanner will ignore the proxy to get the URLs,
                                 but the pentest\attack itself will go thru proxy.
                               * PROXY should be in format like this: 127.0.0.1:8080
                               * It's experimental
       --show-my-ip          Shows your internet IP, current country and user-agent.
                             Useful if you want to test your vpn\proxy config.
# Plugins:
       --plugins                 List all loaded plugins and quit after that.
  -I , --install-plugins         Shows some official exploit-mode plugins you can install
                                 and\or upgrade.
# Other:
       --update-def          Checks and updates your definition files found in the
                             config directory.
       --test-rfi            A quick test to see if you have configured RFI nicely.
       --merge-xml=XMLFILE   Use this if you have another fimap XMLFILE you want to
                             include to your own fimap_result.xml.
  -C , --enable-color        Enables a colorful output. Works only in linux!
       --force-run           Ignore the instance check and just run fimap even if lockfile
                             exists. WARNING: This may erase your fimap_results.xml file!
  -v , --verbose=LEVEL       Verbose level you want to receive.
                             LEVEL=3 -> Debug
                             LEVEL=2 -> Info(Default)
                             LEVEL=1 -> Messages
                             LEVEL=0 -> High-Level
       --credits             Shows some credits.
       --greetings           Some greetings ;)
  -h , --help                Shows this cruft.
# Examples:
 1. Scan a single URL for FI errors:
       fimap -u 'http://localhost/test.php?file=bang&id=23'
 2. Scan a list of URLS for FI errors:
```

```
      fimap -m -l '/tmp/urllist.txt'
 3. Scan Google search results for FI errors:
      fimap -g -q 'inurl:include.php'
 4. Harvest all links of a webpage with recurse level of 3 and
    write the URLs to /tmp/urllist
      fimap -H -u 'http://localhost' -d 3 -w /tmp/urllist
```

Scan the web application (-u “http://192.168.1.202/index.php”) for file inclusion issues:

```
root@kali:~# fimap -u "http://192.168.1.202/index.php"
fimap v.09 (For the Swarm)
:: Automatic LFI/RFI scanner and exploiter
:: by Iman Karim (fimap.dev@gmail.com)

SingleScan is testing URL: 'http://192.168.1.202/index.php'
```

Flood

This scenario is based on Wreckuests python script. Here is official documentation from https://github.com/JamesJGoodwin/wre

What is this?

Wreckuests is a script, which allows you to run DDoS attacks with HTTP-flood(GET/POST). It's written in pure Python and uses proxy-servers as "bots". OF COURSE, this script is not universal and you can't just drop Pentagon/NSA/whatever website with just a single mouse click. Each attack is unique, and for each website you'd need to search for vulnerabilities and exult them.

Warning: This script is published for educational purposes only! Author will accept no responsibility for any consequences, damage or loss which might result from use. ## Features * Cache bypass with URL parameters randomization * CloudFlare detection and notification of * Automatic gzip/deflate toggling * HTTP Authentication bypass * UserAgent substitution * Referers randomizer * HTTP proxy support

. . . and everything else that kennethreitz/requests can do

Dependencies

- Python 3.5+
- Requests 2.10.0 or higher
- netaddr tested with 0.7.19

Usage

Type under *sudo* mode:

```
python3 wreckuests.py -v <target url> -a <login:pass> -t <timeout>
```

Possible parameters:

`-h` or `--help`:

Prints a message with possible parameters.

`-v` or `--victim`:

Specifies a link to the victim's site page. It could be the website's main page, someone's profile, `.php`-file or even image. Everything that has a lot of weight or is hard for server to give. The choice is yours.

`-a` or `--auth`:

Parameter for bypassing authentication. You'r victim could enable basic HTTP authentication and his website will ask you to enter login and password in popup window. Victim may previously publish login and password data for his users in VK/FB/Twitter and whatever social network.

`-t` or `--timeout`(default: 10):

Parameter to control connection'n'read timeout. This option also controls terminating time. **Note:** if you set `timeout=1` or somewhere about 2-3 seconds, the slow(but still working) proxies will not have any time to even connect to your victim's website and will not even hit it. If you still do not understand how it works - do not change this option. Also, this parameter regulates the intensiveness of requests you sending. So, if you sure your proxies are fast enough - you can reduce this value. Use this accordingly.

Important

A separate thread is created for each proxy address. The more proxies you use - the more threads you create. So, please, do not use way too much proxies. Otherwise, the script may exit abnormaly by meeting segmentation fault.

Grabber

Information obtained from: Kali Tool page

Grabber is a web application scanner. Basically it detects some kind of vulnerabilities in your website. Grabber is simple, not fast but portable and really adaptable. This software is designed to scan small websites such as personals, forums etc. absolutely not big application: it would take too long time and flood your network.

Features:

- Cross-Site Scripting
- SQL Injection (there is also a special Blind SQL Injection module)
- File Inclusion
- Backup files check
- Simple AJAX check (parse every JavaScript and get the URL and try to get the parameters)
- Hybrid analysis/Crystal ball testing for PHP application using PHP-SAT
- JavaScript source code analyzer: Evaluation of the quality/correctness of the JavaScript with JavaScript Lint
- Generation of a file [session_id, time(t)] for next stats analysis.

Source: http://rgaucher.info/beta/grabber/

- Author: Romain Gaucher
- License: BSD

```
root@kali:~# grabber -h
Usage: grabber [options]

Options:
  -h, --help            show this help message and exit
  -u ARCHIVES_URL, --url=ARCHIVES_URL
                        Adress to investigate
  -s, --sql             Look for the SQL Injection
  -x, --xss             Perform XSS attacks
  -b, --bsql            Look for blind SQL Injection
  -z, --backup          Look for backup files
  -d SPIDER, --spider=SPIDER
                        Look for every files
  -i, --include         Perform File Insertion attacks
  -j, --javascript      Test the javascript code ?
  -c, --crystal         Simple crystal ball test.
  -e, --session         Session evaluations
```

Spider the web application to a depth of 1 (–spider 1) and attempt SQL (–sql) and XSS (–xss) attacks at the given URL (–url http://192.168.1.224):

```
root@kali:~# grabber --spider 1 --sql --xss --url http://192.168.1.224
Start scanning... http://192.168.1.224
runSpiderScan @  http://192.168.1.224  |   # 1
Start investigation...
Method = GET  http://192.168.1.224
[Cookie]    0   :   <Cookie PHPSESSID=2742cljd8u6aclfktf1sh284u7 for 192.168.1.224/>
[Cookie]    1   :   <Cookie security=high for 192.168.1.224/>
Method = GET  http://192.168.1.224
[Cookie]    0   :   <Cookie PHPSESSID=2742cljd8u6aclfktf1sh284u7 for 192.168.1.224/>
[Cookie]    1   :   <Cookie security=high for 192.168.1.224/>
```

IHULK

This scenario is based on IHULK.py python script. Here is official documentation from https://github.com/iamaamir/ihulk.py:

IHULK (Improved Http Unbearable Load King) DoS Tool Ported to Python3

This is a python 3 version of HULK by Barry Shteiman with some improvements

Usage:

```
python3 ihulk.py <url>
```

you can add “safe” after url, to autoshut after dos

```
python3 ihulk.py <url> safe
```

Note:

The tool is meant for educational purposes only, and should not be used for malicious activity of any kind.

Nikto

Information obtained from: Kali Tool page

Nikto is an Open Source (GPL) web server scanner which performs comprehensive tests against web servers for multiple items, including over 6700 potentially dangerous files/programs, checks for outdated versions of over 1250 servers, and version specific problems on over 270 servers. It also checks for server configuration items such as the presence of multiple index files, HTTP server options, and will attempt to identify installed web servers and software. Scan items and plugins are frequently updated and can be automatically updated.

Nikto is not designed as a stealthy tool. It will test a web server in the quickest time possible, and is obvious in log files or to an IPS/IDS. However, there is support for LibWhisker's anti-IDS methods in case you want to give it a try (or test your IDS system).

Not every check is a security problem, though most are. There are some items that are "info only" type checks that look for things that may not have a security flaw, but the webmaster or security engineer may not know are present on the server. These items are usually marked appropriately in the information printed. There are also some checks for unknown items which have been seen scanned for in log files.

Features: - Here are some of the major features of Nikto. See the documentation for a full list of features and how to use them. - SSL Support (Unix with OpenSSL or maybe Windows with ActiveState's Perl/NetSSL) - Full HTTP proxy support - Checks for outdated server components - Save reports in plain text, XML, HTML, NBE or CSV - Template engine to easily customize reports - Scan multiple ports on a server, or multiple servers via input file (including nmap output) - LibWhisker's IDS encoding techniques - Easily updated via command line - Identifies installed software via headers, favicons and files - Host authentication with Basic and NTLM - Subdomain guessing - Apache and cgiwrap username enumeration - Mutation techniques to "fish" for content on web servers - Scan tuning to include or exclude entire classes of vulnerability checks - Guess credentials for authorization realms (including many default id/pw combos) - Authorization guessing handles any directory, not just the root directory - Enhanced false positive reduction via multiple methods: headers, - page content, and content hashing - Reports "unusual" headers seen - Interactive status, pause and changes to verbosity settings - Save full request/response for positive tests - Replay saved positive requests - Maximum execution time per target - Auto-pause at a specified time - Checks for common "parking" sites - Logging to Metasploit - Thorough documentation

Source: https://www.cirt.net/Nikto2

Author: Chris Sullo & David Lodge License: GNU General Public License (GPL)

```
root@kali:~# nikto -Help

   Options:
       -ask+               Whether to ask about submitting updates
                               yes   Ask about each (default)
                               no    Don't ask, don't send
                               auto  Don't ask, just send
       -Cgidirs+           Scan these CGI dirs: "none", "all",
                           or values like "/cgi/ /cgi-a/"
       -config+            Use this config file
       -Display+           Turn on/off display outputs:
                               1     Show redirects
                               2     Show cookies received
                               3     Show all 200/OK responses
                               4     Show URLs which require
                                     authentication
                               D     Debug output
                               E     Display all HTTP errors
                               P     Print progress to STDOUT
                               S     Scrub output of IPs and
                                     hostnames
                               V     Verbose output
       -dbcheck            Check database and other key files for
                           syntax errors
       -evasion+           Encoding technique:
                               1     Random URI encoding (non-UTF8)
                               2     Directory self-reference (/./)
                               3     Premature URL ending
                               4     Prepend long random string
                               5     Fake parameter
                               6     TAB as request spacer
                               7     Change the case of the URL
                               8     Use Windows directory
                                     separator (\)
                               A     Use a carriage return (0x0d)
                                     as a request spacer
                               B     Use binary value 0x0b as
```

```
                          a request spacer
 -Format+           Save file (-o) format:
                        csv   Comma-separated-value
                        htm   HTML Format
                        nbe   Nessus NBE format
                        sql   Generic SQL (see docs
                              for schema)
                        txt   Plain text
                        xml   XML Format
                        (if not specified the format will be
                        taken from the file extension passed
                        to -output)
-Help               Extended help information
-host+              Target host
-404code            Ignore these HTTP codes as negative
                    responses (always). Format is "302,301".
-404string          Ignore this string in response body
                    content as negative response (always).
                    Can be a regular expression.
-id+                Host authentication to use, format is
                    id:pass or id:pass:realm
-key+               Client certificate key file
-list-plugins       List all available plugins, perform no
                    testing
-maxtime+           Maximum testing time per host
                    (e.g., 1h, 60m, 3600s)
-mutate+            Guess additional file names:
                         1   Test all files with all root directories
                         2   Guess for password file names
                         3   Enumerate user names via Apache (/~user type requests)
                         4   Enumerate user names via cgiwrap
                             (/cgi-bin/cgiwrap/~user type requests)
                         5   Attempt to brute force sub-domain names,
                             assume that the host name is the parent domain
                         6   Attempt to guess directory names from the supplied
                             dictionary file
-mutate-options     Provide information for mutates
-nointeractive      Disables interactive features
-nolookup           Disables DNS lookups
-nossl              Disables the use of SSL
-no404              Disables nikto attempting to guess a 404 page
-Option             Over-ride an option in nikto.conf, can be issued multiple times
-output+            Write output to this file ('.' for auto-name)
-Pause+             Pause between tests (seconds, integer or float)
-Plugins+           List of plugins to run (default: ALL)
-port+              Port to use (default 80)
-RSAcert+           Client certificate file
-root+              Prepend root value to all requests, format is /directory
-Save               Save positive responses to this directory ('.' for auto-name)
-ssl                Force ssl mode on port
-Tuning+            Scan tuning:
                         1     Interesting File / Seen in logs
                         2     Misconfiguration / Default File
                         3     Information Disclosure
                         4     Injection (XSS/Script/HTML)
                         5     Remote File Retrieval -
                               Inside Web Root
                         6     Denial of Service
                         7     Remote File Retrieval - Server Wide
                         8     Command Execution / Remote Shell
                         9     SQL Injection
                         0     File Upload
                         a     Authentication Bypass
                         b     Software Identification
                         c     Remote Source Inclusion
                         d     WebService
                         e     Administrative Console
                         x     Reverse Tuning Options
                         (i.e., include all except specified)
-timeout+           Timeout for requests (default 10 seconds)
-Userdbs            Load only user databases, not the
                     standard databases
```

```
                          all   Disable standard dbs and load
                               only user dbs
                          tests Disable only db_tests and
                               load udb_tests
       -useragent         Over-rides the default useragent
       -until             Run until the specified time or duration
       -update            Update databases and plugins
                          from CIRT.net
       -useproxy          Use the proxy defined in nikto.conf,
                          or argument http://server:port
       -Version           Print plugin and database versions
       -vhost+            Virtual host (for Host header)
      + requires a value

root@kali:~#
root@kali:~# nikto -Display 1234EP -o report.html -Format htm \
                   -Tuning 123bde -host 192.168.0.102
- Nikto v2.1.6
---------------------------------------------------------------------
+ Target IP:          192.168.0.102
+ Target Hostname:    192.168.0.102
+ Target Port:        80
+ Start Time:         2018-03-23 10:49:04 (GMT0)
---------------------------------------------------------------------
+ Server: Apache/2.2.22 (Ubuntu)
+ Server leaks inodes via ETags, header found with file /,
  inode: 287, size: 11832, mtime: Fri Feb  2 15:27:56 2018
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can
  hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow
  the user agent to render the content of the site in a different
  fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all
  possible dirs)
+ "robots.txt" contains 1 entry which should be manually viewed.
+ Uncommon header 'tcn' found, with contents: list
+ Apache mod_negotiation is enabled with MultiViews, which allows
  attackers to easily brute force file names. See
  http://www.wisec.it/sectou.php?id=4698ebdc59d15. The following
  alternatives for 'index' were found: index.html
+ Apache/2.2.22 appears to be outdated (current is at least
  Apache/2.4.12). Apache 2.0.65 (final release) and 2.2.29 are
  also current.
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS
+ 371 requests: 0 error(s) and 9 item(s) reported on remote host
+ End Time:           2018-03-23 10:50:44 (GMT0) (100 seconds)
---------------------------------------------------------------------
+ 1 host(s) tested
root@kali:~#
root@kali:~# firefox report.html
```

Skipfish

Information obtained from: Kali Tool page

Skipfish is an active web application security reconnaissance tool. It prepares an interactive sitemap for the targeted site by carrying out a recursive crawl and dictionary-based probes. The resulting map is then annotated with the output from a number of active (but hopefully non-disruptive) security checks. The final report generated by the tool is meant to serve as a foundation for professional web application security assessments.

Key features:

- High speed: pure C code, highly optimized HTTP handling, minimal CPU footprint – easily achieving 2000 requests per second with responsive targets.
- Ease of use: heuristics to support a variety of quirky web frameworks and mixed-technology sites, with automatic learning capabilities, on-the-fly wordlist creation, and form autocompletion.
- Cutting-edge security logic: high quality, low false positive, differential security checks, capable of spotting a range of subtle flaws, including blind injection vectors.

Source: https://code.google.com/p/skipfish/

- Author: Google Inc, Michal Zalewski, Niels Heinen, Sebastian Roschke
- License: Apache-2.0

```
root@kali:~# skipfish -h
skipfish web application scanner - version 2.10b
Usage: skipfish [ options ... ] -W wordlist -o output_dir \
                start_url [ start_url2 ... ]

Authentication and access options:

  -A user:pass      - use specified HTTP authentication credentials
  -F host=IP        - pretend that 'host' resolves to 'IP'
  -C name=val       - append a custom cookie to all requests
  -H name=val       - append a custom HTTP header to all requests
  -b (i|f|p)        - use headers consistent with MSIE / Firefox / iPhone
  -N                - do not accept any new cookies
  --auth-form url   - form authentication URL
  --auth-user user  - form authentication user
  --auth-pass pass  - form authentication password
  --auth-verify-url -  URL for in-session detection

Crawl scope options:

  -d max_depth     - maximum crawl tree depth (16)
  -c max_child     - maximum children to index per node (512)
  -x max_desc      - maximum descendants to index per branch (8192)
  -r r_limit       - max total number of requests to send (100000000)
  -p crawl%        - node and link crawl probability (100%)
  -q hex           - repeat probabilistic scan with given seed
  -I string        - only follow URLs matching 'string'
  -X string        - exclude URLs matching 'string'
  -K string        - do not fuzz parameters named 'string'
  -D domain        - crawl cross-site links to another domain
  -B domain        - trust, but do not crawl, another domain
  -Z               - do not descend into 5xx locations
  -O               - do not submit any forms
  -P               - do not parse HTML, etc, to find new links

Reporting options:

  -o dir          - write output to specified directory (required)
  -M              - log warnings about mixed content / non-SSL passwords
  -E              - log all HTTP/1.0 / HTTP/1.1 caching intent mismatches
  -U              - log all external URLs and e-mails seen
  -Q              - completely suppress duplicate nodes in reports
  -u              - be quiet, disable realtime progress stats
  -v              - enable runtime logging (to stderr)

Dictionary management options:

  -W wordlist     - use a specified read-write wordlist (required)
  -S wordlist     - load a supplemental read-only wordlist
  -L              - do not auto-learn new keywords for the site
  -Y              - do not fuzz extensions in directory brute-force
```

```
  -R age          - purge words hit more than 'age' scans ago
  -T name=val     - add new form auto-fill rule
  -G max_guess    - maximum number of keyword guesses to keep (256)

  -z sigfile      - load signatures from this file

Performance settings:

  -g max_conn     - max simultaneous TCP connections, global (40)
  -m host_conn    - max simultaneous connections, per target IP (10)
  -f max_fail     - max number of consecutive HTTP errors (100)
  -t req_tmout    - total request response timeout (20 s)
  -w rw_tmout     - individual network I/O timeout (10 s)
  -i idle_tmout   - timeout on idle HTTP connections (10 s)
  -s s_limit      - response size limit (400000 B)
  -e              - do not keep binary responses for reporting

Other settings:

  -l max_req      - max requests per second (0.000000)
  -k duration     - stop scanning after the given duration h:m:s
  --config file   - load the specified configuration file

Send comments and complaints to <heinenn@google.com>.
```

Using the given directory for output (-o 202) , scan the web application URL (http://192.168.1.202/wordpress):

```
root@kali:~# skipfish -o 202 http://192.168.1.202/wordpress

skipfish version 2.10b by lcamtuf@google.com

  - 192.168.1.202 -

Scan statistics:

      Scan time : 0:00:05.849
  HTTP requests : 2841 (485.6/s), 1601 kB in, 563 kB out (370.2 kB/s)
    Compression : 802 kB in, 1255 kB out (22.0% gain)
    HTTP faults : 0 net errors, 0 proto errors, 0 retried, 0 drops
 TCP handshakes : 46 total (61.8 req/conn)
     TCP faults : 0 failures, 0 timeouts, 16 purged
 External links : 512 skipped
   Reqs pending : 0

Database statistics:

         Pivots : 13 total, 12 done (92.31%)
    In progress : 0 pending, 0 init, 0 attacks, 1 dict
  Missing nodes : 0 spotted
     Node types : 1 serv, 4 dir, 6 file, 0 pinfo, 0 unkn, 2 par, 0 val
   Issues found : 10 info, 0 warn, 0 low, 8 medium, 0 high impact
      Dict size : 20 words (20 new), 1 extensions, 202 candidates
     Signatures : 77 total

[+] Copying static resources...
[+] Sorting and annotating crawl nodes: 13
[+] Looking for duplicate entries: 13
[+] Counting unique nodes: 11
[+] Saving pivot data for third-party tools...
[+] Writing scan description...
[+] Writing crawl tree: 13
[+] Generating summary views...
[+] Report saved to '202/index.html' [0x7054c49d].
[+] This was a great day for science!
```

Slowloris

This scenario is based on slowloris.py python script. Here is official documentation from https://github.com/gkbrk/slowloris:

What is Slowloris?

Slowloris is basically an HTTP Denial of Service attack that affects threaded servers. It works like this:

1. We start making lots of HTTP requests.
2. We send headers periodically (every ~15 seconds) to keep the connections open.
3. We never close the connection unless the server does so. If the server closes a connection, we create a new one keep doing the same thing.

This exhausts the servers thread pool and the server can't reply to other people.

How to install and run?

You can clone the git repo or install using **pip**. Here's how you run it.

- `sudo pip3 install slowloris`
- `slowloris example.com`

That's all it takes to install and run slowloris.py.

If you want to clone using git instead of pip, here's how you do it.

- `git clone https://github.com/gkbrk/slowloris.git`
- `cd slowloris`
- `python3 slowloris.py example.com`

SOCKS5 proxy support

However, if you plan on using the `-x` option in order to use a SOCKS5 proxy for connecting instead of a direct connection over your IP address, you will need to install the `PySocks` library (or any other implementation of the `socks` library) as well. `PySocks` is a fork from `SocksiPy` by GitHub user @Anorov and can easily be installed by adding `PySocks` to the `pip` command above or running it again like so:

- `sudo pip3 install PySocks`

You can then use the `-x` option to activate SOCKS5 support and the `--proxy-host` and `--proxy-port` option to specify the SOCKS5 proxy host and its port, if they are different from the standard `127.0.0.1:8080`.

Configuration options

It is possible to modify the behaviour of slowloris with command-line arguments.

License

sqlmap

Information obtained from: project homepage

Introduction

sqlmap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections.

Features

- Full support for MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase, SAP MaxDB, Informix, HSQLDB and H2 database management systems.
- Full support for six SQL injection techniques: boolean-based blind, time-based blind, error-based, UNION query-based, stacked queries and out-of-band.
- Support to directly connect to the database without passing via a SQL injection, by providing DBMS credentials, IP address, port and database name.
- Support to enumerate users, password hashes, privileges, roles, databases, tables and columns.
- Automatic recognition of password hash formats and support for cracking them using a dictionary-based attack.
- Support to dump database tables entirely, a range of entries or specific columns as per user's choice. The user can also choose to dump only a range of characters from each column's entry.
- Support to search for specific database names, specific tables across all databases or specific columns across all databases' tables. This is useful, for instance, to identify tables containing custom application credentials where relevant columns' names contain string like name and pass.
- Support to download and upload any file from the database server underlying file system when the database software is MySQL, PostgreSQL or Microsoft SQL Server.
- Support to execute arbitrary commands and retrieve their standard output on the database server underlying operating system when the database software is MySQL, PostgreSQL or Microsoft SQL Server.
- Support to establish an out-of-band stateful TCP connection between the attacker machine and the database server underlying operating system. This channel can be an interactive command prompt, a Meterpreter session or a graphical user interface (VNC) session as per user's choice.
- Support for database process' user privilege escalation via Metasploit's Meterpreter getsystem command.
- Refer to the wiki for an exhaustive breakdown of the features.

Documentation

- sqlmap User's manual.
- sqlmap History.
- sqlmap Frequently Asked Questions (FAQ).
- Material around sqlmap presented at conferences.

License

This program is free software; you may redistribute and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; Version 2 (or later) with the clarifications and exceptions described in the license file. This guarantees your right to use, modify, and redistribute this software under certain conditions. If you wish to embed sqlmap technology into proprietary software, we sell alternative licenses (contact sales@sqlmap.org).

Disclaimer

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License v2.0 for more details at http://www.gnu.org/licenses/gpl-2.0.html.

Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program.

Uniscan

Information obtained from: Kali Tool page

Uniscan is a simple Remote File Include, Local File Include and Remote Command Execution vulnerability scanner.

Source: http://sourceforge.net/projects/uniscan/

- Author: Douglas Poerschke Rocha
- License: GPLv2

```
root@kali:~# uniscan -h
####################################
# Uniscan project                  #
# http://uniscan.sourceforge.net/  #
####################################
V. 6.3


OPTIONS:
    -h  help
    -u  <url> example: https://www.example.com/
    -f  <file> list of url's
    -b  Uniscan go to background
    -q  Enable Directory checks
    -w  Enable File checks
    -e  Enable robots.txt and sitemap.xml check
    -d  Enable Dynamic checks
    -s  Enable Static checks
    -r  Enable Stress checks
    -i  <dork> Bing search
    -o  <dork> Google search
    -g  Web fingerprint
    -j  Server fingerprint

usage:
[1] perl ./uniscan.pl -u http://www.example.com/ -qweds
[2] perl ./uniscan.pl -f sites.txt -bqweds
[3] perl ./uniscan.pl -i uniscan
[4] perl ./uniscan.pl -i "ip:xxx.xxx.xxx.xxx"
[5] perl ./uniscan.pl -o "inurl:test"
[6] perl ./uniscan.pl -u https://www.example.com/ -r
```

Scan the given URL (-u http://192.168.1.202/) for vulnerabilities, enabling directory and dynamic checks (-qd):

```
root@kali:~# uniscan -u http://192.168.1.202/ -qd
####################################
# Uniscan project                  #
# http://uniscan.sourceforge.net/  #
####################################
V. 6.2


Scan date: 16-5-2014 16:29:48
============================================================
| Domain: http://192.168.1.202/
| Server: Apache/2.2.22 (Debian)
| IP: 192.168.1.202
============================================================
|
| Directory check:
| [+] CODE: 200 URL: http://192.168.1.202/joomla/
| [+] CODE: 200 URL: http://192.168.1.202/wordpress/
============================================================
|
| Crawler Started:
| Plugin name: FCKeditor upload test v.1 Loaded.
| Plugin name: Web Backdoor Disclosure v.1.1 Loaded.
| Plugin name: phpinfo() Disclosure v.1 Loaded.
| Plugin name: E-mail Detection v.1.1 Loaded.
| Plugin name: Timthumb <= 1.32 vulnerability v.1 Loaded.
| Plugin name: Code Disclosure v.1.1 Loaded.
| Plugin name: Upload Form Detect v.1.1 Loaded.
```

```
| Plugin name: External Host Detect v.1.2 Loaded.
| [+] Crawling finished, 27 URL's found!
```

Wfuzz

Information obtained from: Kali Tool page

Wfuzz is a tool designed for bruteforcing Web Applications, it can be used for finding resources not linked (directories, servlets, scripts, etc), bruteforce GET and POST parameters for checking different kind of injections (SQL, XSS, LDAP,etc), bruteforce Forms parameters (User/Password), Fuzzing,etc.

Some features: - Multiple Injection points capability with multiple dictionaries - Recursion (When doing directory bruteforce) - Post, headers and authentication data brute forcing - Output to HTML - Colored output - Hide results by return code, word numbers, line numbers, regex - Cookies fuzzing - Multi threading - Proxy support - SOCK support - Time delays between requests - Authentication support (NTLM, Basic) - All parameters bruteforcing (POST and GET) - Multiple encoders per payload - Payload combinations with iterators - Baseline request (to filter results against) - Brute force HTTP methods - Multiple proxy support (each request through a different proxy) - HEAD scan (faster for resource discovery) - Dictionaries tailored for known applications (Weblogic, Iplanet, Tomcat, Domino, Oracle 9i, Vignette, Coldfusion and many more

Source: https://github.com/xmendez/wfuzz/

- Author: Christian Martorella, Carlos del ojo, Xavier Mendez aka Javi
- License: GPLv2

```
root@kali:~# wfuzz --help

********************************************************
* Wfuzz 2.2.11 - The Web Fuzzer                        *
*                                                      *
* Version up to 1.4c coded by:                         *
* Christian Martorella (cmartorella@edge-security.com) *
* Carlos del ojo (deepbit@gmail.com)                   *
*                                                      *
* Version 1.4d to 2.2.11 coded by:                     *
* Xavier Mendez (xmendez@edge-security.com)            *
********************************************************

Usage:  wfuzz [options] -z payload,params <url>

    FUZZ, ..., FUZnZ  wherever you put these keywords wfuzz will replace them with the
                values of the specified payload.
    FUZZ{baseline_value} FUZZ will be replaced by baseline_value.
        It will be the first request performed and could be used as a base for filtering.


Options:
  -h/--help           : This help
  --help              : Advanced help
  --version           : Wfuzz version details
  -e <type>           : List of available encoders/payloads/iterators/printers/scripts

  --recipe <filename>       : Reads options from a recipe
  --dump-recipe <filename>  : Prints current options as a recipe
  --oF <filename>           : Saves fuzz results to a file. These can be consumed
                              later using the wfuzz payload.

  -c                   : Output with colors
  -v                   : Verbose information.
  -f filename,printer  : Store results in the output file using the specified printer
                         (raw printer if omitted).
  -o printer           : Show results using the specified printer.
  --interact           : (beta) If selected,all key presses are captured. This allows
                         you to interact with the program.
  --dry-run            : Print the results of applying the requests without actually
                         making any HTTP request.
  --prev               : Print the previous HTTP requests (only when using payloads
                         generating fuzzresults)

  -p addr             : Use Proxy in format ip:port:type. Repeat option for using various
                        proxies. Where type could be SOCKS4,SOCKS5 or HTTP if omitted.

  -t N                : Specify the number of concurrent connections (10 default)
  -s N                : Specify time delay between requests (0 default)
  -R depth            : Recursive path discovery being depth the maximum recursion level.
  -L,--follow         : Follow HTTP redirections
  -Z                  : Scan mode (Connection errors will be ignored).
```

```
  --req-delay N       : Sets the maximum time in seconds the request is allowed to take
                        (CURLOPT_TIMEOUT). Default 90.
  --conn-delay N      : Sets the maximum time in seconds the connection phase to the server
                        to take (CURLOPT_CONNECTTIMEOUT). Default 90.

  -A              : Alias for --script=default -v -c
  --script=           : Equivalent to --script=default
  --script=<plugins>      : Runs script's scan. <plugins> is a comma separated list of
                            plugin-files or plugin-categories
  --script-help=<plugins>     : Show help about scripts.
  --script-args n1=v1,...     : Provide arguments to scripts. ie. --script-args
                                grep.regex="<A href="(.*?)">"

  -u url                      : Specify a URL for the request.
  -m iterator         : Specify an iterator for combining payloads (product by default)
  -z payload          : Specify a payload for each FUZZ keyword used in the form
                        of name[,parameter][,encoder].
                      A list of encoders can be used, ie. md5-sha1. Encoders can be chained,
                      ie. md5@sha1.
                      Encoders category can be used. ie. url
                      Use help as a payload to show payload plugin's details
                      (you can filter using --slice)
  --zP <params>           : Arguments for the specified payload
                            (it must be preceded by -z or -w).
  --slice <filter>        : Filter payload's elements using the specified expression.
                            It must be preceded by -z.
  -w wordlist         : Specify a wordlist file (alias for -z file,wordlist).
  -V alltype          : All parameters bruteforcing (allvars and allpost). No need for
                        FUZZ keyword.
  -X method           : Specify an HTTP method for the request, ie. HEAD or FUZZ

  -b cookie           : Specify a cookie for the requests. Repeat option for various
                        cookies.
  -d postdata         : Use post data (ex: "id=FUZZ&catalogue=1")
  -H header           : Use header (ex:"Cookie:id=1312321&user=FUZZ").
                        Repeat option for various headers.
  --basic/ntlm/digest auth  : in format "user:pass" or "FUZZ:FUZZ" or "domain\FUZ2Z:FUZZ"

  --hc/hl/hw/hh N[,N]+  : Hide responses with the specified code/lines/words/chars
                          (Use BBB for taking values from baseline)
  --sc/sl/sw/sh N[,N]+  : Show responses with the specified code/lines/words/chars
                          (Use BBB for taking values from baseline)
  --ss/hs regex         : Show/hide responses with the specified regex within the content
  --filter <filter>     : Show/hide responses using the specified filter expression
                          (Use BBB for taking values from baseline)
  --prefilter <filter>  : Filter items before fuzzing using the specified expression.
```

Use colour output (-c), a wordlist as a payload (-z file,/usr/share/wfuzz/wordlist/general/common.txt), and hide 404 messages (–hc 404) to fuzz the given URL (http://192.168.1.202/FUZZ):

```
root@kali:~# wfuzz -c -z file,/usr/share/wfuzz/wordlist/general/common.txt \
                  --hc 404 http://192.168.1.202/FUZZ

********************************************************
* Wfuzz 2.2.11 - The Web Fuzzer                        *
********************************************************

Target: http://192.168.1.202/FUZZ
Payload type: file,/usr/share/wfuzz/wordlist/general/common.txt

Total requests: 950
=====================================================================
ID  Response   Lines      Word         Chars          Request
=====================================================================

00429:  C=200      4 L        25 W      177 Ch    " - index"
00466:  C=301      9 L        28 W      319 Ch    " - javascript"
```

Crawler

TODO

SMB Attack

TODO

SMB Normal

TODO