

[TOOL] WeBrute – Directory Brute Forcer

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2004-11/0107.html>

From: SecuriTeam (support_at_securiteam.com)

Date: 11/30/04

To: list@securiteam.com

Date: 30 Nov 2004 18:54:18 +0200

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

The SecuriTeam alerts list – Free, Accurate, Independent.

Get your security news from a reliable source.

<http://www.securiteam.com/maillinglist.html>

WeBrute – Directory Brute Forcer

SUMMARY

DETAILS

```
#!/usr/bin/perl
#ooOOoo
#
# ** !!! WARNING !!! *
# * DO NOT DISTRIBUTE **
# * FOR SECURITY TESTING ONLY! *
# **
# * By using this code you agree that I makes no warranties or
representations, express or implied, about the *
# * accuracy, timeliness or completeness of this, including without
limitations the implied warranties of *
# * merchantability and fitness for a particular purpose. *
# * I makes NO Warranty of non-infringement. This code may contain
technical inaccuracies or typographical errors. *
# * This code can never be copyrighted or owned by any commercial company,
under no circumstances what so ever. *
# * but can be use for as long the developer, are giving explicit approval
of the usage, and the user understand *
# * and approve of all the parts written in this notice. *
```

Securiteam: [TOOL] WeBrute – Directory Brute Forcer

```
# * Neither myself nor any of my Affiliates shall be liable for any
direct, incidental, consequential, indirect *
# * or punitive damages arising out of access to, inability to access, or
any use of the content of this code, *
# * including without limitation any PC, other equipment or other
property, even if I am Expressly advised of *
# * the possibility of such damages. I DO NOT encourage criminal
activities. If you use this code or commit *
# * criminal acts with it, then you are solely responsible for your own
actions and by use, downloading,transferring, *
# * and/or reading anything from this code you are considered to have
accepted the terms and conditions and have read *
# * this disclaimer. Once again this code is for penetration testing
purposes only. And once again, DO NOT DISTRIBUTE! *
# **
#
#ooOOoo
#
# (C)opyright 2004 by Dennis Rand
# This tool is developed, maintained and owned by Dennis Rand.
# Ideas for new features or changes within the code or possible bugs can
be included by informing me.
# email: tools@cirt.dk – http://www.cirt.dk/Tools/webrute.txt –
http://www.cirt.dk/Tools/common.txt
#
#ooOOoo
#
# What this tool does:
# WeBrute is a Brute Forcing tool to discover hidden directories, files or
parameters in the URL
# of a webserver.
# Normally looking for anything that does NOT respond "404"., this however
can be modified, reversed
# to only look and log if specific error messages, or other information
within the responce from the
# webserver.
#
# Advanced options tries to check for Buffer Overflows in the URL
parameters
#
#ooOOoo
#
# Examples:
# Scan 127.0.0.1 port 80, incremental all chars from A–Z, 0–9 and some
special chars but max 3 combinations long
# WeBrute.pl –t 127.0.0.1 –p 80 –i all –min 1 –max 3
#
# Scan 127.0.0.1 port 80, incremental all chars from A–Z and 0–9 but max 3
combinations long
# WeBrute.pl –t 127.0.0.1 –p 80 –i str,int –min 1 –max 3
#
```

Securiteam: [TOOL] WeBrute – Directory Brute Forcer

```
# Scan 127.0.0.1 port 80, Use wordlist
# WeBrute.pl -t 127.0.0.1 -p 80 -w common.txt
#
# Scan 127.0.0.1 port 80, Use wordlist and admin as start path
# WeBrute.pl -t 127.0.0.1 -p 80 -w common.txt -pre admin/<bruteforce>
#
# Scan 127.0.0.1 port 80, incremental from 1-3 combinations and secret/ as
start path
# WeBrute.pl -t 127.0.0.1 -p 80 -i all -min 1 -max 3 -pre
secret/<bruteforce>
#
# Scan 127.0.0.1 port 80, Incremental and bruteforce for xxxxxx.asp
# WeBrute.pl -t 127.0.0.1 -p 80 -i all -min 1 -max 3 -append
<bruteforce>.asp
#
# Scan 127.0.0.1 port 80, Incremental and bruteforce path secret for
xxxxx.asp
# WeBrute.pl -t 127.0.0.1 -p 80 -i all -min 1 -max 3 -prepend secret/
-append <bruteforce>.asp
#
# Scan 127.0.0.1 port 80, incremental and bruteforce for articleID
# WeBrute.pl -t 127.0.0.1 -p 80 -i int -min 1 -max 3 -prepend
showarticle.asp?articleid=<bruteforce> -append .art
#
# Scan 127.0.0.1 port 80, Incremental, bruteforce ID and look for modified
error message "Denied"
# WeBrute.pl -t 127.0.0.1 -p 80 -i int -min 1 -max 3 -prepend
showproduct.asp?id=<bruteforce> -error Denied
#
# Scan 127.0.0.1 port 80, Use wordlist, and HTTP/1.1
# WeBrute.pl -t 127.0.0.1 -p 80 -w common.txt -httpver HTTP/1.1
#
# Scan 127.0.0.1 port, use Wordlist and set your own referer
# WeBrute.pl -t 127.0.0.1 -p 80 -w common.txt -referer
"http://127.0.0.1/refererpoint"
#
# Scan 127.0.0.1 port 80, Use wordlist, and set your Cookie of choice
# WeBrute.pl -t 127.0.0.1 -p 80 -w common.txt -setcookie
"ASPCOOKIE_AND_WHATEVER Set_Path=/"
#
# Scan 127.0.0.1 port 80, Use wordlist, and HTTP/1.1, Just keeps scanning
even if the FalsePositive are true
# WeBrute.pl -t 127.0.0.1 -p 80 -w common.txt -httpver HTTP/1.1 -force
#
# Scan 127.0.0.1 port 443, Use wordlist, and SSL
# WeBrute.pl -t 127.0.0.1 -p 443 -w common.txt -ssl
#
# Scan 127.0.0.1 port 80, Use wordlist, and Positive scanning and Error
message to look for specific response
# WeBrute.pl -t 127.0.0.1 -p 80 -w common.txt -Positive -error "200 OK"
#
```

Securiteam: [TOOL] WeBrute – Directory Brute Forcer

```
# Scan 127.0.0.1 port 80, Use wordlist, and traverse scanning and verbose
# WeBrute.pl -t 127.0.0.1 -p 80 -w common.txt -trav -verbose
#
# Scan 127.0.0.1 port 80, Use wordlist, and Positive scanning and Error
message to look for specific response
# and look for the PUT when scanning with OPTIONS, -command is Default:
GET
# WeBrute.pl -t 127.0.0.1 -p 80 -w common.txt -Positive -error "PUT"
-command OPTIONS
#
# Use a file containing many ip/hostnames to discover what the banner is
of current WebServer
# WeBrute.pl -word IP-range.txt -banner
#
# Scan 127.0.0.1 port 80, Use wordlist, and in Debug mode that writes all
output to screen
# WeBrute.pl -t 127.0.0.1 -p 80 -w common.txt -debug
#
# Scan 127.0.0.1 port 80, Use wordlist, and in Debug mode and delays 10
seconds between requests
# WeBrute.pl -t 127.0.0.1 -p 80 -w common.txt -debug -delay 10
#
# Scan 127.0.0.1 port 80, Use wordlist, and in Debug mode and writes 20
lines of output from server response to screen
# WeBrute.pl -t 127.0.0.1 -p 80 -w common.txt -debug -showli 20
#
# Scan 127.0.0.1 port 80, Use incremental, and POST numbers to a form, use
-prepend and -append to set data before and after
# WeBrute.pl -t 127.0.0.1 -p 80 -inc int -mi 1 -ma 2 -postreq
"scripts/reqdll.dll"
#
# Resumes, the file WeBrute.Resume.File has to be present
# WeBrute.pl -resume
#
# Advanced options for incremental usage (POSSIBLE DoS, iF VULNERABLE):
#
# Scans 127.0.0.1 port 80, minimum 1 and maximum 4096, tries to make a
BufferOverflow with A's
# A good idea if possible is to have a Debugger running in the server
being scanned
# WeBrute.pl -t 127.0.0.1 -p 80 -i bof -mi 1 -max 4096
#
# Scans 127.0.0.1 port 80, minimum 1 and maximum 512, tries to make a
Format String Overflow with %s combinations
# WeBrute.pl -t 127.0.0.1 -p 80 -i bos -mi 1 -max 512
#
# Scans 127.0.0.1 port 80, minimum 1 and maximum 512, tries to make a
Format String Overflow with %r combinations
# WeBrute.pl -t 127.0.0.1 -p 80 -i bor -mi 1 -max 512
#
# Scans 127.0.0.1 port 80, minimum 1 and maximum 512, tries to make a
```

Securiteam: [TOOL] WeBrute – Directory Brute Forcer

Format String Overflow with %n combinations

```
# WeBrute.pl -t 127.0.0.1 -p 80 -i bon -mi 1 -max 512
#
#
#ooOOoo
#
# Options:
# -t, -target=HOST Hostname of Webserver (default at localhost)
# -p, -port=PORT Port Webserver is running at (default at 80)
# -w, -wordlist=WORD,FILE Word or filename(s) of word lists
# -i, -incremental=MODE Incremental mode = str, int, spec, high or all
they can also be combined
# -i -incremental=MODE Incremental mode, advanced = bof, bos, bor, bon
(Dangerous)
# -ba -banner Banner Discovery – Scan an file with IP-ranges for banner
info
# -pr -prepend=STR Eg.: Tries to bruteforce a file name with a defined
extension
# -ap -append=STR Eg.: Define further down an url to scan
# -e -error=ERROR Define the error message that server gives (Default:
404)
# -l, -log=FILE Filename to write result to
# -mi, -min-chars=NUMBER Min. chars in incremental mode (defaults at 1)
# -ma -max-chars=NUMBER Max. chars in incremental mode (defaults at 4)
# -ht -httpver=STR Choose what HTTP version you want to use eg.: HTTP/1.1
# -set -setcookie=STR Makes it possible to insert a Cookie of own choice
# -ss -ssl Uses SSL in the scan
# -com -command Set the request option Default: GET
# -fo -force Force the scan even if False_Positive alert are true
# -should only be used with bof,bos,bor or bon
# -pos -positive Insted of looking for anything else then the error
message,
# only look for GET's = Error message
# -us -useragent=STR Setup the scanning with other UserAgent then default
(MS IE 6.0)
# -ref -referer=STR Setup tour own referer in the HTTP header.
# -te -text Saves logfile in clear text not any html crap
# -ht -html Saves logfile in HTML, this is also standard
# -ra -raw Saves logfile in RAW, meaning only what triggered an responce
from the server
# -nol -nolog Does not make any logfile
# -re -resume Resumes from previous scan
# -de -debug Debug writes the output from an request to the screen.
# -sho -showlines Amount of lines to output in Debug mode (Default: 15
lines)
# -del -delay Seconds between request in Debug mode (Default: 5 seconds)
# -po -postreq When wanting to POST to an URL
# -?, -help Print this help, then exit
# -he, -help Print this help, then exit
#
#ooOOoo
```

Securiteam: [TOOL] WeBrute – Directory Brute Forcer

```
#
# TODO:
# Make error message work and give more information, when parameters are
wrong
# Make incremental scanning do real incremental
(1,2,3,4,5,6,7,8,9,10,11,12....23,24....)
#
#ooOOoooOO
#
# Remember:
# It is only possible to have on resume file at a time in the scanning
location so
# run the program from the location where you want the scannings results.
# use the parameter "--resume" and you should be one it again.
#
# Dependencies:
# perl -MCPAN -e shell
# cpan > install Bundle::LWP Used for all WWW related requests (GET -e
xxxxxxxxxxx)
# cpan > install IO::Socket Used for the connection
# cpan > install Getopt::Long Used for the choices, and values
# cpan > install Net::SSLeay::Handle Used for connecting to HTTPS servers
# cpan > install Term::ANSIColor Used for the nice color code
# cpan > quit
#
# Clean a Wordfile before use to avoid doubles:
# cat Common.txt | sort | uniq > Temp.txt
# mv -f Temp.txt Common.txt
#
#ooOOoo
#ooOOoo
#
# Modifications/Version:
# Version 1.0+beta Free version, can be modified an freely used.
# Version 2.8: Made the possibility of making a positive scan instead of
default negative scan
# Version 2.9: Fixed the bug in the scan where it allways says that the
first scan request, were valid
# Fixed the bug in incremental scan, using the $append twice in code....
set sucker = true;
# Fixed minor problem with Return after a scan with -nolog set
# Added "Request number $count" in debug mode, for more information, in
the single requests.
# Added minor modifications when using -debug, label info does not display
now in debug mode.
# Added possibility to use CTRL + C to stop scanning, and then the resume
file is made.
# Fixed Counting bug in -verbose mode, when Successfully attack are made
# Added a delay of 5 seconds for every successfully attempt in -verbose
mode
# Added full request line in -verbose mode.
```

Securiteam: [TOOL] WeBrute – Directory Brute Forcer

```
# Added possibility of setting own Command default is GET
# Fixed error in report of HTML file
# Version 3.0:
# Added Banner grap to include in report Normal txt and HTML
# Fixed a bug in the $success counting succesfully attacks.
# Version 3.1:
# Added save to a local file of all successfully findings, and let html
log point to this.
# Fixed a bug in the savings of the above regarding Foldername.
# Added in the HTML report that it is possible to see both remote and
local version
# Fixed a bug in the HTML report, when scanning an HTTPS server
# Version 3.2:
# Added support for POST request bruteforcing.
# Added support in the HTML report for viewing what HTTP request command
are used.
# Added possibility of setting your own referer in the HTTP header, else
an default will be set.
# Added function to make –verbose mode more nice when it prints out it's
information.
# Optimized the start scan splash screen to get less copy code.
# Version 3.3:
# Added function to put all findings into a file, for furture use.
# Added function to make traverse scanning meaning scanning deeper, into
findings.
# Added Traverse info in the HTML report.
# Fixed a problem regarding –verbose mode when using traversal scan.
# Fixed a error in the HTML report generated.
#
#ooOOoo
#ooOOoo

use IO::Socket;
use Getopt::Long;
use Net::SSLeay::Handle qw/shutdown/;
use Term::ANSIColor;

$name = "Webserver BruteForcer"; # Name of application
$ver = "3.3"; # Version of WeBrute
$copyright = "(c)2004 by Dennis Rand";
@charstr = ('a'..'z'); # Characters a–z used in incremental bruteforcing
@charstr_high = ('A'..'Z'); # Charecters A–Z used in incremental
bruteforcing
@charsint = ('0'..'9'); # Characters 0–9 used in incremental bruteforcing
@charspec = ('=','?','@','$','!','%','-','_','&','!','+'); # Special
Characters used in incremental bruteforcing
@charformats = ('%', 's');
@charformatn = ('%', 'n');
@charformatr = ('%', 'r');
@charbof = ('A');
@charsall = (@charstr, @charsint, @charspec); # Combining Characters used
```

Securiteam: [TOOL] WeBrute – Directory Brute Forcer

```
in incremental bruteforcing
$target = "127.0.0.1"; # Default target
$port = "80"; # Default port
$resume_file = "WeBrute.Resume.File"; # Default resume save file
$user_agent = "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)"; #
Internet Explorer 6.0 on Win2k
$count, $tries, $ssl, $failed = 0; # Counter being reset
$textlog, $verbose, $success = 0; #
$html_log = 1;
$savep_count = 1;
$http_ver = "HTTP/1.1";
$num = 1; # Amount of times to do a False Positive Check
>false = "WEBRUTE_FALSE_POSITIVE_CHECK";
$found = "X";
$notfound = "*";
$delay = 5; # How long time to sleep, in debug mode between requests
$line_numbers = 15; # Amount of lines to show in debug mode
$command = "GET"; # Request command, default GET
$savep_filename = "URL";
$count_max = 80; # Used to make -verbose mode look a bit nicer
$teach_file = "WeBrute.Traverse.File";
```

```
GetOptions(
    "target=s" => \ $target,
    "port=i" => \ $port,
    "min=i" => \ $min_chars,
    "max=i" => \ $max_chars,
    "append=s" => \ $append,
    "prepend=s" => \ $prepend,
    "error=s" => \ $error_name,
    "logname=s" => \ $log,
    "incremental=s" => \ $incremental,
    "wordlist=s" => \ $wordlist,
    "resume" => \ $resume,
    "ssl" => \ $ssl,
    "force" => \ $force,
    "httpver=s" => \ $http_ver,
    "setcookie=s" => \ $setcookie,
    "verbose" => \ $verbose,
    "text|txt" => \ $textlog,
    "html" => \ $html_log,
    "raw" => \ $raw_log,
    "useragent=s" => \ $user_agent,
    "debug" => \ $debug,
    "nolog" => \ $nolog,
    "showlines=i" => \ $line_numbers,
    "delay=i" => \ $delay,
    "positive" => \ $positive,
    "command=s" => \ $command,
    "banner" => \ $bannergrap,
    "alert" => \ $alert,
```

Securiteam: [TOOL] WeBrute – Directory Brute Forcer

```
"postreq=s" => \$post_request,
"referer=s" => \$referer,
"teachmode" => \$teach_mode,
"traverse" => \$traverse,
  "help?" => sub {
    print color("green"), "\n";
    print "\t#####\n";
    print "\t# $name $ver #\n";
    print "\t# * !!! WARNING !!! ** #\n";
    print "\t# ** #\n";
    print "\t# ** #\n";
    print "\t# $copyright #\n";
    print "\t#####\n";
    print "\n\tooOoooo\n";
    print "\t Parameters\n";
    print "\t -target\t\tDefault: $target\n";
    print "\t -port\t\tDefault: $port\n";
    print "\t -prepend\t\tDefault: None – Remember the \"^\" after
directory eg.: admin\n";
    print "\t -append\t\tDefault: None – Remember eg.: \".asp\"\n";
    print "\t Scan Options:\n";
    print "\t -wordlist\t\tDefault: None\n";
    print "\t -incremental\t\tParameters (all, str, int, spec, high)\n";
    print "\t -min\t\tDefault: 1 – (Incremental scanning)\n";
    print "\t -max\t\tDefault: 4 – (Incremental scanning)\n";
    print "\t -resume\t\tResume a stopped scanning\n";
    print "\t -positive\t\tOnly find tests equal to -error\n";
    print "\t -banner\t\tGraps the banner from the Webserver\n";
    print "\t -traverse\t\tGo as deep as possible and traverse into
findings\n";
    print "\t Advanced Options:\n";
    print "\t -error\t\tDefault: 404\n";
    print "\t -httpver\t\tDefault: $http_ver\n";
    print "\t -ssl\t\tTo enable SSL support\n";
    print "\t -setcookie\t\tDefault: None insert the cookie of your
choice\n";
    print "\t -referer\t\tDefault: referer: http://$target:$port\n";
    print "\t -useragent\t\tDefault: $user_agent\n";
    print "\t -command\t\tDefault: $command – Could be used with -positive
scan\n";
    print "\t -force\t\tBypass False Positive Check",color("bold red"),"
(Not a good idea)\n",color("reset");
    print color("green"), "";
    print "\t -postreq\t\tUsed to set URL to POST data to eg.
scripts/ntdll.dll\n";
    print "\t Debugging Options:\n";
    print "\t -debug\t\tWrites response from the server. \n";
    print "\t -showlines\t\tDefault: $line_numbers amount of lines shown
when using the -debug \n";
    print "\t -delay\t\tDefault: $delay seconds, between requests in debug
mode \n\n";
```

Securiteam: [TOOL] WeBrute – Directory Brute Forcer

```
print "\t Log options:\n";
print "\t -logname\t\tDefault: WebBrute-http(s)-$target.html\n";
print "\t -html\t\tDefault: Saves the logfile in HTML,
WeBrute-<targetname>.html\n";
print "\t -text\t\tSaves logfile in clear text,
WeBrute-<targetname>.txt\n";
print "\t -raw\t\tSaves the logfile in raw format,
WeBrute-<targetname>.raw \n";
print "\t -nolog\t\tNo log file is made, used when doing buffer
overflow check \n";
print "\t -verbose\t\tWrite all output to screen\n\n";
print "\t Advanced Options for incremental:",color("bold red")," (Can be
Dangerous, DoS/Buffer Overflow attack)",color("reset")," \n";
print color("green"), "";
print "\t -incremental bof\t\tTries to Overflows with A's\n";
print "\t -incremental bos\t\tTries to Format string overflow, with
%s combinations\n";
print "\t -incremental bor\t\tTries to Format string overflow, with
%r combinations\n";
print "\t -incremental bon\t\tTries to Format string overflow, with
%n combinations\n";
print "\n\tooOOoooo";
print "\n\tooOOoooo\n\n", color("reset");
exit;
}
);
#ooOOoo

$error .= color("bold green"),"WeBrute: You may not specify more than one
of the `~res or ~inc or ~wor or ~ban' option\n",color("reset") if
(($wordlist) && ($incremental) && ($resume) && ($bannergrap));
$error .= color("bold green"),"WeBrute: You must specify one of the `~i or
~w or ~r' options\n",color("reset") if ((!$wordlist) && (!$incremental) &&
(!$resume));
$error .= color("bold green"),"WeBrute: You must only specify
--min-chars/--min-chars with ~i\n",color("reset") if ((!$incremental) &&
(($min_chars) || ($max_chars)));

if ($error) {
print color("bold red")," \n\tnooOOoooo";
print "\n\tTry WeBrute.pl -help or -? for more information.\n";
print "ooOOoooo\n\n",color("reset");
exit;
};

#ooOOoo
#ooOOoo

$min_chars = "1" if (!$min_chars);
$max_chars = "4" if (!$max_chars);
$prepend = "" if (!$prepend);
```

Securiteam: [TOOL] WeBrute – Directory Brute Forcer

```
$append = "" if (!$append);
$error_name = "404" if (!$error_name);
$savep_dirname = "HTML_SAVE-http-".$target."-".$port;
$savep_dirname_ssl = "HTML_SAVE-https-".$target."-".$port;

#ooOOoo
#ooOOoo
# Catch Interupt – CTRL + C

sub catchInterrupt {
    $SIG{INT} = sub {exit;};
    print color("bold red"),"\n\n", "oo00" x 12, "\n[X] Interrupted\t\t\t\t –
DONE\n",color("reset");
    &wresume;
    exit;
};

$SIG{INT} = \&catchInterrupt;

# verify that interrupt handler was installed properly

unless(defined($SIG{INT})){$print "Unable to install signal handler,
contact $copyright";}
unless($SIG{INT} == \&catchInterrupt){print "There was an unexpected error
installing the signal handler, contact $copyright";}

#ooOOoooOO
#ooOOoooOO
# Setting up the connection string for &main

sub Con {
    $| = 1;
    $remote = IO::Socket::INET->new(
        Proto => "tcp",
        PeerAddr => $target,
        PeerPort => $port,
        Reuse => 1,
        # Timeout => 15,
    )
}

#ooOOoo
#ooOOoo
# Setting up the connection string for &banner

sub Connection {
    $| = 1;
    $remoteb = IO::Socket::INET->new(
        Proto => "tcp",
        PeerAddr => $buffer,
        PeerPort => $port,
```

Securiteam: [TOOL] WeBrute – Directory Brute Forcer

```
Reuse => 1,
Timeout => 15,);
if($verbose){
if(!$remoteb){print STDERR "Webserver is: None found\r\n";next;}
} else {
if(!$remoteb){print color("Green"),"$notfound",color("reset");next;}
}
}
#ooOOoo
#ooOOoo
# Saves successfully hits, for offline viewing later

sub savePageContent {
if(!$ssl){
mkdir("$savep_dirname");
open(FH, ">",
$savep_dirname."/".$savep_filename.$savep_count++.".html");
} else {
mkdir("$savep_dirname_ssl");
open(FH, ">",
$savep_dirname_ssl."/".$savep_filename.$savep_count++.".html");
}
print FH "$result\n";
close FH;
}

#ooOOoo
#ooOOoo
# Simple Header information for Cookie request.

$host_header_simple = "TE: deflate,gzip;q=0.3\r\n";
$host_header_simple .= "Connection: Close\r\n";
$host_header_simple .= "Host: $target\r\n";
$host_header_simple .= "User-Agent: $user_agent\r\n";

#ooOOoo
#ooOOoo
# Set the cookie manually or automatic if not set to manual and if
possible
if(!$bannergrap){
if(!$resume){
if ($setcookie) {
$cookie = $setcookie;
if($debug){
print color("bold red"),"\nCookie Manually Set to:
",color("green"),"$cookie\n",color("reset");print "\nDelaying $delay
sec.\n\n";
sleep $delay;
}
}
else {
```

Securiteam: [TOOL] WeBrute – Directory Brute Forcer

```
$response_ccode = undef;
Con();
if($ssl){
    $ssl=1;
eval {
    tie(*SSL, "Net::SSLLeay::Handle", $target,$port);
    };
print SSL "$command / $http_ver\r\n$host_header_simple\r\n\r\n";
shutdown(*SSL, 1);
while(<SSL>){
    if(!defined($response_ccode)){ $response_ccode = $_;}
    $cookie .= $_;
    }
    ($cookie) = $cookie =~ m/Set-Cookie: (.?); path=/;
} else {
print $remote "$command / $http_ver\r\n$host_header_simple\r\n\r\n";
while(<$remote>){
if(!defined($response_ccode)){ $response_ccode = $_;}
    $cookie .= $_;
    }
    ($cookie) = $cookie =~ m/Set-Cookie: (.?); path=/;
if($debug){
    if(!$cookie){
        $cookie = "No Cookie Retrieved";
        print color("bold red"),"\nCookie Set to:
",color("green"),"$cookie\n",color("reset");
        print "\nDelaying $delay sec.\n\n";
        sleep $delay;
        $cookie = "";
    } else {
        print color("bold red"),"\nCookie Set to:
",color("green"),"$cookie\n",color("reset");
        print "\nDelaying $delay sec.\n\n";
        sleep $delay;
    }
}
}
}
}
}

#ooOOoo
#ooOOoo
# Grapping the Banner, for report information

if(!$bannergrap){
    if (!$resume){

        $response_bcode = undef;
        Con();
        if($ssl){
```

Securiteam: [TOOL] WeBrute – Directory Brute Forcer

```
$ssl=1;
eval {
tie(*SSL, "Net::SSLLeay::Handle", $target,$port);
};
print SSL "HEAD / $http_ver\r\n$host_header_simple\r\n\r\n";
shutdown(\*SSL, 1);
while(<SSL>){
    if(!defined($response_bcode)){ $response_bcode = $_;}
    $banner .= $_;
    }
    ($banner) = $banner =~ m/Server:\s+(.*)/;
} else {
print $remote "HEAD / $http_ver\r\n$host_header_simple\r\n\r\n";
while(<$remote>){
if(!defined($response_bcode)){ $response_bcode = $_;}
$banner .= $_;
}
($banner) = $banner =~ m/Server:\s+(.*)/;
if($debug){
print color("bold red"),"Webserver is an:
",color("green"),"$banner\n",color("reset");
print "\nDelaying $delay sec.\n\n";
sleep $delay;
}
}
}

#ooOOoo
#ooOOoo
# Setting up the main header information
if(!$referer){
if($ssl){
$referer = "https://$target:$port/$prepend";
} else {
$referer = "http://$target:$port/$prepend";
}
}
$host_header = "Accept: */*\r\n";
$host_header .= "Accept-Language: en\r\n";
$host_header .= "Accept-Encoding: gzip, deflate\r\n";
$host_header .= "User-Agent: $user_agent\r\n";
$host_header .= "Host: $target\r\n";
$host_header .= "Referer: $referer\r\n";
$host_header .= "Cookie: $cookie\r\n";
$host_header .= "Connection: Close\r\n";

#ooOOoo
#ooOOoo
# Do a false positive test
sub false_positive_scan{
```

Securiteam: [TOOL] WeBrute – Directory Brute Forcer

```

if(!$command eq "POST" ){
if(!$resume){
if(!$bannergrap){
if(!$force){
if(!$positive){
while($tries++ < $num){
    $falsepositive = "";
    Con();
if($ssl){
$ssl=1;
$response_fpcode = undef;
    eval {
        tie(*SSL, "Net::SSLLeay::Handle", $target,$port);
    };
    print SSL "$command /$prepend$false$append
$http_ver\r\n$host_header\r\n\r\n";
shutdown(\*SSL, 1);
while(<SSL>){
    if(!defined($response_fcode)){ $response_fcode = $_;}
    $falsepositive .= $_;
    }
if($debug){
    print color("bold green"),"xxXX" x 10,color("bold red"),"\n\n$command
/$prepend$false$append $http_ver\n",color("reset");
    my @lines = split(/\n/,$falsepositive);
    my $firstlines;
    for(0 .. $line_numbers){
        $firstlines .= $lines[$_];
        print "Line $_ $lines[$_]\n";
    }
    $found = "";
    $notfound = "";
    print "\nDelaying $delay sec.before next request:\n\n\n";
    sleep $delay;
    }
    if($falsepositive !~ m/$error_name/mgs) {print color("bold
red"),"\nFALSE POSITIVE ALERT:\n",color("bold green"),"Seems like there
are something wrong with the error message,\nplease see if the website has
mofified error messages.\n\nIf you want to continue, add the parameter:
\"-force\"
\nREMEMBER FORCE SCANNING GENERATE A LARGE LOGFILE\nIf you don't
want to use log \"-nolog\"
\n\nOr use the parameter -error <error msg>\nnto
specify a error message.\n\n",color("reset"); exit;}
    } else {
$response_fpcode = undef;
print $remote "$command /$prepend$false$append
$http_ver\r\n$host_header\r\n\r\n";
while(<$remote>){
    if(!defined($response_fpcode)){ $response_fpcode = $_;}
    $falsepositive .= $_;
    }
if($debug){

```


Securiteam: [TOOL] WeBrute – Directory Brute Forcer

```
$value = $' if (/^Resume=/);
$http_ver = $' if (/^HTTP_Version=/);
$ssl = $' if (/^SSL=/);
$verbose = $' if (/^Verbose=/);
$textlog = $' if (/^TextLog=/);
$html_log = $' if (/^HTMLLog=/);
$raw_log = $' if (/^RAWLog=/);
$user_agent = $' if (/^UserAgent=/);
$debug = $' if (/^Debug=/);
$cookie = $' if (/^Cookie=/);
$nolog = $' if (/^Nolog=/);
$line_numbers = $' if (/^LineNumbers=/);
$delay = $' if (/^Delay=/);
$positive = $' if (/^Positive=/);
$command = $' if (/^Command=/);
$bannergrap = $' if (/^BannerScan=/);
$post_request = $' if (/^PostRequest=/);
$referer = $' if (/^Referer=/);
$teach_mode = $' if (/^TeachMode=/);
$traverse = $' if (/^Traverse=/);
}
} else {
    print color("bold red"),"WeBrute: The Resume File '$resume'
could not be found\n",color("reset");
    print color("bold red"),"Try WeBrute.pl --help or --? for more
information.\n",color("reset");
    exit;
}
}
```

```
#ooOOoo
#ooOOoo
# Setting the logfile name if not manually set
```

```
if (!$ssl){
    if ($textlog){
        $html_log = 0;
        $raw_log = 0;
        $nolog = 0;
        $textlog = 1;
        $log = "WeBrute-http-$target.txt" if (!$log);
    }
    if ($raw_log){
        $textlog = 0;
        $html_log = 0;
        $nolog = 0;
        $raw_log = 1;
        $log = "WeBrute-http-$target.raw" if (!$log);
    }
    if ($nolog){
        $textlog = 0;
```

Securiteam: [TOOL] WeBrute – Directory Brute Forcer

```
$raw_log = 0;
$html_log = 0;
$nolog = 1;
}
if ($html_log){
    $textlog = 0;
    $raw_log = 0;
    $nolog = 0;
    $html_log = 1;
    $log = "WeBrute-http-$target.html" if (!$log);
}
} else {
if ($textlog){
    $html_log = 0;
    $raw_log = 0;
    $nolog = 0;
    $textlog = 1;
    $log = "WeBrute-https-$target.txt" if (!$log);
}
if ($raw_log){
    $textlog = 0;
    $html_log = 0;
    $nolog = 0;
    $raw_log = 1;
    $log = "WeBrute-https-$target.raw" if (!$log);
}
if ($nolog){
    $textlog = 0;
    $raw_log = 0;
    $html_log = 0;
    $nolog = 1;
    $log = "" if (!$log);
}
if ($html_log){
    $textlog = 0;
    $raw_log = 0;
    $nolog = 0;
    $html_log = 1;
    $log = "WeBrute-https-$target.html" if (!$log);
}
}
#ooOOoo
#ooOOoo
# Start splash screen for scanning

sub start_scan_splash {
    if (!$verbose){
        if (!$debug){
            print color("green"), "\nooOOoo\n",color("reset");
            print color("bold blue")," Description:\n Use \"-verbose\" to get more
information\n",color("green"), "\t$notfound\tNot a positive
```


Setting up incremental procedure

```

if ($incremental) {
&start_scan_splash;
@temp = split(/./, $incremental);
foreach $input (@temp) {
    if ($input eq "str" ) {push @chars, @charstr;}
    elsif ($input eq "int" ) {push @chars, @charsint;}
    elsif ($input eq "spec") {push @chars, @charspec;}
    elsif ($input eq "high") {push @chars, @charstr_high;}
    elsif ($input eq "bos" ) {push @chars, @charformats;}
    elsif ($input eq "bon" ) {push @chars, @charformatn;}
    elsif ($input eq "bor" ) {push @chars, @charformatr;}
    elsif ($input eq "bof" ) {push @chars, @charbof;}
    elsif ($input eq "all" ) { @chars = @charsall; break;}
    else
    {
        print color("bold red"),"Try WeBrute.pl -help or -? for more
information.\nWeBrute: You must specify one or more of the 'all, int, str,
spec' options\n",color("reset");exit;
    }
}
&slog;
if(!$straverse){
&inremen;
} else {
&traverse_scan;
}
sub incremen{
$buffer = $chars[0] x $min_chars;
do {
&main;
$appending = 0;
for ($j = 0; $j < length($buffer); $j++) {
    for ($i = 0; $i < @chars; $i++) {
        if (substr($buffer, $j, 1) eq $chars[$i]) {
            if ($i == $#chars) {
                substr($buffer, $j, 1) = $chars[0];
            } else {
                substr($buffer, $j, 1) = $chars[++$i];
                ++$appending;
            }
        }
    }
}
last if ($appending);
}
$buffer .= $chars[0] if (!$appending);
} until (length($buffer) > $max_chars);
}

```

Securiteam: [TOOL] WeBrute – Directory Brute Forcer

```
&end_log;
}

#ooOOoo
#ooOOoo
# Setting up Wordfile procedure

if ($wordlist) {
&start_scan_splash;
&slog;
if(!$straverse){
&wordlst;
} else {
&traverse_scan;
}

sub wordlst{
foreach (split(/./, $wordlist)) {
if (-f $_) {
open(_FH, $_);

while (<_FH>) {
s/^x//;
chomp;
$buffer = "";
$buffer = $_;
if (!$bannergrap){
&main;
} else {
&banner;
}
}
close(_FH);
} else {
print color("bold red"),"WeBrute: The wordlist file you requested '$_'
could not be located\n";
print "Try WeBrute.pl -help or -? for more information.\n$error\n",
color("reset");
exit;
}
}
&end_log
}

#ooOOoo
#ooOOoo
# Writes the start of the logfile

sub slog {
if(!$nolog){
```

```

$timestamp = localtime;
if ($resume) {
    open(FH, ">>", $log);
    if($textlog){
        print FH "$sendtime\t\=>Scan was Stopped\n";
    }
    print FH "$timestamp\t\=>Scan was Resumed\n";
}
if($raw_log){
}
if($html_log){
print FH " <TR>\r\n";
print FH " <TD>\r\n";
print FH " <B>$sendtime\r\n";
print FH " </TD>\r\n";
print FH " <TD><LEFT>\r\n";
print FH " <B>Scan was Stopped</B>\r\n";
print FH " </LEFT></TD>\r\n";
print FH " </TR>\r\n";
print FH " <TR>\r\n";
print FH " <TD>\r\n";
print FH " <B>$timestamp\r\n";
print FH " </TD>\r\n";
print FH " <TD><LEFT>\r\n";
print FH " <B>Scan was Resumed</B>\r\n";
print FH " </LEFT></TD>\r\n";
print FH " </TR>\r\n";
}
close FH;
} else {
    if($textlog){
        $method .= "Incremental – $incremental – Minimum extension:$min_chars –
Maximum extension:$max_chars\n" if ($incremental);
        $method .= "Wordlist – $wordlist\n" if ($wordlist);
    }
    if($html_log){
        if(!$bannergrap){
            $method .= " <B>Scan type:</B></TD><TD><LEFT><B>Incremental –
$incremental</B></LEFT></TD></TR><TR><TD><B>Minimum
extension:</B></TD><TD><LEFT><B>$min_chars</B></LEFT></TD></TR><TR><TD><B>Maximum
extension:</B></TD><TD><LEFT><B>$max_chars</B></LEFT></TD></TR>\r\n" if ($incremental);
            $method .= " <B>Scan type:</B></TD><TD><LEFT><B>Wordlist
</B></LEFT></TD></TR><TR><TD><B>Wordlist used:
</B></TD><TD><LEFT><B>$wordlist </B></LEFT></TD></TR>\r\n" if ($wordlist);
        } else {
            $method .= " <B>Scan type:</B></TD><TD><LEFT><B>Banner
</B></LEFT></TD></TR><TR><TD><B>Wordlist used:
</B></TD><TD><LEFT><B>$wordlist </B></LEFT></TD></TR>\r\n" if
($bannergrap);
        }
    }
    open(FH, ">", $log);
}

```


Securiteam: [TOOL] WeBrute – Directory Brute Forcer

```
SIZE=1><B>$copyright</B></FONT>\n</LEFT>\n</TD>\n</TABLE>\n<BR>\r\n";
if(!$bannergrap){
  print FH "<TABLE WIDTH=80% BGCOLOR=black CELLSPACING=0 CELLPADDING=2
BORDER=0>\n<TR>\n<TD>\n<CENTER>\n<FONT FACE=Tahoma COLOR=white
SIZE=+1><B>Audit Rapport for
$target</B>\n</FONT>\n</LEFT>\n</TD>\n</TABLE>\n<BR>\r\n";
  print FH "<TABLE WIDTH=80% BGCOLOR=white CELLSPACING=0 CELLPADDING=0
BORDER=0><TR ALIGN=left>\r\n";
} else {
  print FH "<TABLE WIDTH=80% BGCOLOR=black CELLSPACING=0 CELLPADDING=2
BORDER=0>\n<TR>\n<TD>\n<CENTER>\n<FONT FACE=Tahoma COLOR=white
SIZE=+1><B>Banner Audit
Rapport</B>\n</FONT>\n</LEFT>\n</TD>\n</TABLE>\n<BR>\r\n";
  print FH "<TABLE WIDTH=80% BGCOLOR=white CELLSPACING=0 CELLPADDING=0
BORDER=0><TR ALIGN=left>\r\n";
}
if(!$bannergrap){
  print FH "\r\n";
  print FH " <TR>\r\n";
  print FH " <TD>\r\n";
  print FH " <B>Target:</B>\r\n";
  print FH " </TD>\r\n";
  print FH " <TD><LEFT>\r\n";
  if($ssl){
    print FH " <B><A
HREF=\<u>https://$target:$port/$prepend$trav\</u>https://$target:$port/$prepend$trav". "[Bruteforce]". "$append</A></B>
"
  } else {
    print FH " <B><A
HREF=\<u>http://$target:$port/$prepend$trav\</u>http://$target:$port/$prepend$trav". "[Bruteforcing]". "$append</A></B>
"
  }
  print FH " </LEFT></TD>\r\n";
  print FH " </TR>\r\n";
  print FH " <TR>\r\n";
  print FH " <TD>\r\n";
  print FH " <B>Target Banner:</B>\r\n";
  print FH " </TD>\r\n";
  print FH " <TD><LEFT>\r\n";
  print FH " <B>$banner</B>\r\n";
  print FH " </LEFT></TD>\r\n";
  print FH " </TR>\r\n";
  print FH "\r\n";
  print FH "\r\n";
  print FH " <TR>\r\n";
  print FH " <TD>\r\n";
  print FH " <B>HTTP Type:</B>\r\n";
  print FH " </TD>\r\n";
  print FH " <TD><LEFT>\r\n";
  if($ssl){
    print FH " <B>SSL</B>\r\n";
  } else {
    print FH " <B>Normal</B>\r\n";
  }
}
```

```

}
print FH " </LEFT></TD>\r\n";
print FH " </TR>\r\n";
}
print FH "\r\n";
print FH " <TR>\r\n";
print FH " <TD>\r\n";
print FH " <B>Port number:</B>\r\n";
print FH " </TD>\r\n";
print FH " <TD><LEFT>\r\n";
print FH " <B>$port</B>\r\n";
print FH " </LEFT></TD>\r\n";
print FH " </TR>\r\n";
print FH "\r\n";
print FH " <TR>\r\n";
print FH " <TD>\r\n";
print FH " <B>HTTP version:</B>\r\n";
print FH " </TD>\r\n";
print FH " <TD><LEFT>\r\n";
print FH " <B>$http_ver</B>\r\n";
print FH " </LEFT></TD>\r\n";
print FH " </TR>\r\n";
print FH "\r\n";
print FH " <TR>\r\n";
print FH " <TD>\r\n";
print FH " <B>Command used:</B>\r\n";
print FH " </TD>\r\n";
print FH " <TD><LEFT>\r\n";
print FH " <B>$command $post_request</B>\r\n";
print FH " </LEFT></TD>\r\n";
print FH " </TR>\r\n";
print FH "\r\n";
print FH " <TR>\r\n";
print FH " <TD>\r\n";
print FH $method."\n";
print FH "\r\n";
if($cookie){
    print FH " <TR>\r\n";
    print FH " <TD>\r\n";
    print FH " <B>Cookie:</B>\r\n";
    print FH " </TD>\r\n";
    print FH " <TD><LEFT>\r\n";
    print FH " <B>$cookie</B>\r\n";
    print FH " </LEFT></TD>\r\n";
    print FH " </TR>\r\n";
    print FH "\r\n";
}
if($traverse){
    print FH " <TR>\r\n";
    print FH " <TD>\r\n";
    print FH " <B>Traversing:</B>\r\n";
}

```

```

print FH " </TD>\r\n";
print FH " <TD><LEFT>\r\n";
print FH " <B>ENABLED</B>\r\n";
print FH " </LEFT></TD>\r\n";
print FH " </TR>\r\n";
print FH "\r\n";
}
print FH " <TR>\r\n";
print FH " <TD>\r\n";
print FH " <B>Scan Started:</B>\r\n";
print FH " </TD>\r\n";
print FH " <TD><LEFT>\r\n";
print FH " <B>$timestamp</B>\r\n";
print FH " </LEFT></TD>\r\n";
print FH " </TR>\r\n";
print FH " </TABLE>\r\n";
print FH " <BR><TABLE WIDTH=80% BGCOLOR=black CELLSPACING=0 CELLPADDING=2
BORDER=0><TR><TD><CENTER><FONT FACE=Tahoma COLOR=white SIZE=+1><B>Audit
Results</B></FONT></LEFT></TD></TABLE>\r\n";
print FH "\r\n";
print FH " <TABLE WIDTH=80% BGCOLOR=white CELLSPACING=0 CELLPADDING=0
BORDER=0><TR ALIGN=left><BR>\r\n";
}
close FH;
}
}
}
#ooOOoooOO
#ooOOoooOO
# Writes findings to the log

sub wlog {
&teach;
if(!$nolog){
    $timestamp = localtime;
    open(FH, ">>", $log);
    if ($raw_log){
        print FH "$buffer\r\n"
    }
    if ($textlog){
        if(!$bannergrap){
            if($ssl){
                print FH "$timestamp\t\=>
https://$target:$port/$prepend$trav$buffer$append\t\t\t$response_code";
            } else {
                print FH "$timestamp\t\=>
http://$target:$port/$prepend$trav$buffer$append\t\t\t$response_code";
            }
        }
    }
    if($bannergrap){
        if($ssl){

```

Securiteam: [TOOL] WeBrute – Directory Brute Forcer

```

    print FH "https://$buffer:$port \=> $banner";
} else {
print FH "http://$buffer:$port \=> $banner";
}
}
}

if ($html_log){
print FH "\r\n";
print FH " <TR>\r\n";
print FH " <TD>\r\n";
if($ssl){
    if(!$bannergrap){
        print FH " <B><A
HREF=\"\".$savep_dirname_ssl.\"/\".$savep_filename.$savep_count.\".html\">#</A>] – <A
HREF=\"https://$target:$port/$prepend$trav$buffer$append\">https://$target:$port/$prepend$trav$buffer$append>\r\n";
    } else {
    print FH " <B><A
HREF=\"https://$buffer:$port\">https://$buffer:$port>\r\n";
    }
} else {
    if(!$bannergrap){
    print FH " <B><A
HREF=\"\".$savep_dirname.\"/\".$savep_filename.$savep_count.\".html\">#</A>] –
<A
HREF=\"http://$target:$port/$prepend$trav$buffer$append\">http://$target:$port/$prepend$trav$buffer$append>\r\n";
    } else {
    print FH " <B><A
HREF=\"http://$buffer:$port\">http://$buffer:$port>\r\n";
    }
}
print FH " </TD>\r\n";
print FH " <TD><LEFT>\r\n";
if(!$bannergrap){
print FH " <B>$response_code</B>\r\n";
} else {
print FH " <B>$banner</B>\r\n";
}
print FH " </LEFT></TD>\r\n";
print FH " </TR>\r\n";
}
close FH;
}
}

#ooOOoo
#ooOOoo
# Writes the end to the log

sub end_log {
if($nolog){

```

```

    }
    $timestamp = localtime;
    open(FH, ">>", $log);
    if($textlog){
    print FH "\n\tooOOooooo";
    print FH "\n\tooOOooooo\n\n";
    print FH "\tThe Scan completed: $timestamp\n";
    print FH "\tSuccesfull findings: $success\n";
    print FH "\tTotal attempts: $count\n";
    }

    if($html_log){
    print FH "\r\n";
    print FH " </TABLE>\r\n";
    print FH " \r\n\r\n";
    print FH "<BR><TABLE WIDTH=80% BGCOLOR=black CELLSPACING=0 CELLPADDING=2
    BORDER=0><TR><TD><CENTER><FONT FACE=Tahoma COLOR=white SIZE=+1><B>Rapport
    Summary</B></FONT><</LEFT></TD></TABLE><BR>\r\n";
    print FH "<TABLE WIDTH=90% BGCOLOR=white CELLSPACING=0 CELLPADDING=0
    BORDER=0><TR ALIGN=left>\r\n";
    print FH " <TR>\r\n";
    print FH " <TD>\r\n";
    print FH " <B>The Scan completed:</B>\r\n";
    print FH " </TD>\r\n";
    print FH " <TD><LEFT>\r\n";
    print FH " <B>$timestamp<B>\r\n";
    print FH " </LEFT></TD>\r\n";
    print FH " </TR>\r\n";
    if(!$bannergrap){
    print FH "\r\n";
    print FH " <TR>\r\n";
    print FH " <TD>\r\n";
    print FH " <B>Succesfull findings:</B>\r\n";
    print FH " </TD>\r\n";
    print FH " <TD><LEFT>\r\n";
    print FH " <B>$success\r\n";
    print FH " </LEFT></TD>\r\n";
    print FH " </TR>\r\n";
    print FH "\r\n";
    print FH " <TR>\r\n";
    print FH " <TD>\r\n";
    print FH " <B>Total attempts:</B>\r\n";
    print FH " </TD>\r\n";
    print FH " <TD><LEFT>\r\n";

```