GoldenEye

GoldenEye is a python-based HTTP Denial-of-Service (DoS) testing tool used to simulate web server overload conditions by generating large volumes of HTTP requests (https://github.com/jseidl/GoldenEye).

Designed to test how web servers respond to high traffic loads by:

- Opening multiple HTTP connections
- Sending concurrent requests
- Maintaining persistent sessions

So, what's the use?

- Simulates an HTTP flooding attack

!!Limitations!!

- Can crash poorly configured servers
- Exhaust Apache/Ngnix worker pools
- Trigger auto-scaling events
- Cause service downtime

Some quick admin notes:

Now, I have a local network setup with an ubuntu server.

The environment was deployed within the VirtualBox using an isolated Internal Network. This configuration ensured that all testing activity was confined to a closed Layer 2 broadcast domain and did not impact external systems or production networks.

The network used the 192.168.1.0/24 private address space (RFC 1918), with pfSense acting as the default gateway (192.168.1.1). Two Kali Linux machines were deployed, one configured as the testing host and the other for auxiliary analysis. An Ubuntu Server VM hosting Apache (**192.168.1.50**) functioned as the target web server.

The Apache web server was intentionally deployed within a controlled lab environment for the purpose of observing server behavior under high-volume HTTP request conditions. The server was not connected to any external network and did not host production data, ensuring compliance with ethical testing principles (PTES). This configuration prevents traffic from reaching the host operating system or external networks, thereby mitigating the risk of unintended service disruption.

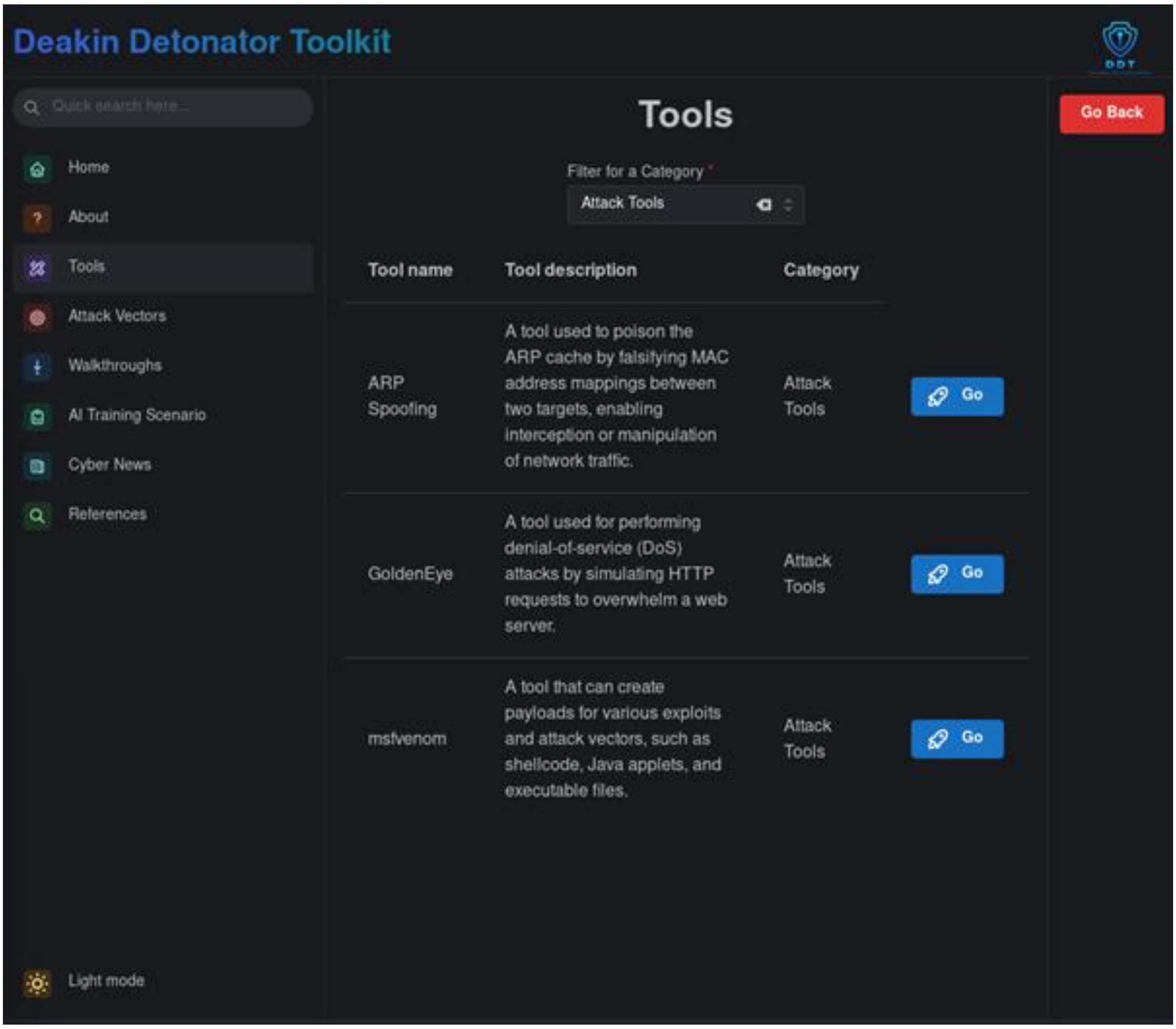
The target URL specified for testing was the internal Apache server hosted on 192.168.1.50 within the isolated LABNET network. This server was intentionally deployed as a controlled target and contained no production data.

The GoldenEye configuration required several key fields to be defined:

- The **URL of Target** specifies the web server address that will receive the generated HTTP traffic.
- The **User-Agent list** field allows custom HTTP header identifiers to be supplied; if left blank, the tool automatically uses its default embedded User-Agent string.
- The **Number of Concurrent Workers** determines how many parallel threads generate HTTP requests, effectively simulating multiple clients.
- The **Number of Concurrent Sockets** controls how many simultaneous TCP connections each worker maintains, influencing connection concurrency.
- The **HTTP Method (GET/POST)** defines the type of request sent to the server, affecting how the application processes incoming traffic.
- Finally, the **Verify SSL Certificate** option determines whether TLS certificates are validated when targeting HTTPS services; this setting is not applicable when testing over HTTP.

The How:

Step 1: Navigate to the tools, go to GoldenEye and select

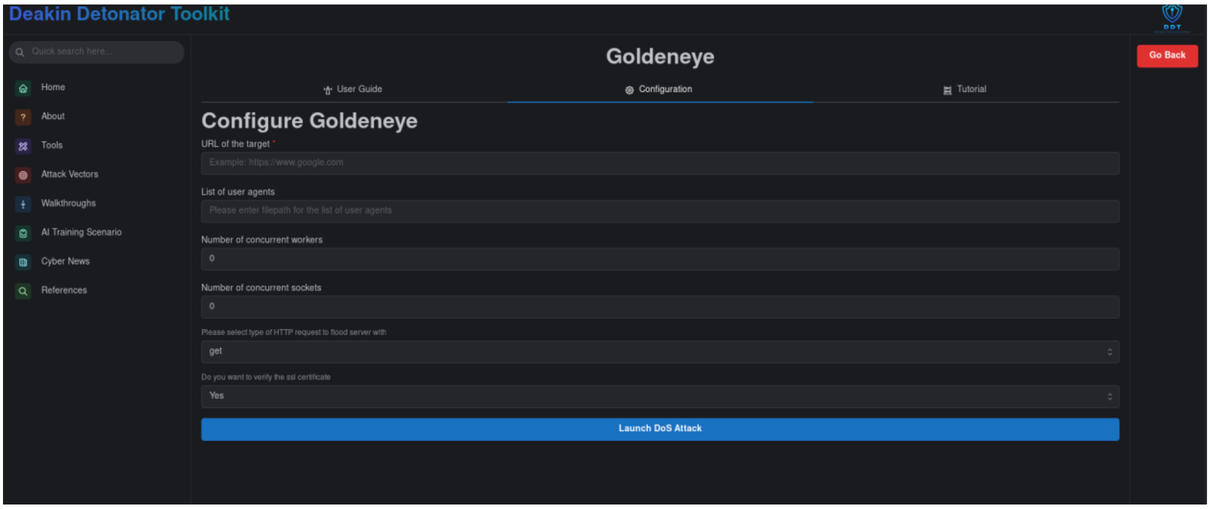


Tools

Filter for a Category *: Attack Tools

Tool name	Tool description	Category
ARP Spoofing	A tool used to poison the ARP cache by falsifying MAC address mappings between two targets, enabling interception or manipulation of network traffic.	Attack Tools
GoldenEye	A tool used for performing denial-of-service (DoS) attacks by simulating HTTP requests to overwhelm a web server.	Attack Tools
msfvenom	A tool that can create payloads for various exploits and attack vectors, such as shellcode, Java applets, and executable files.	Attack Tools

Step 2: Fill out the fields:



Goldeneye

Configure Goldeneye

URL of the target *: Example: https://www.google.com

List of user agents: Please enter filepath for the list of user agents

Number of concurrent workers: 0

Number of concurrent sockets: 0

Please select type of HTTP request to flood server with: get

Do you want to verify the ssl certificate: Yes

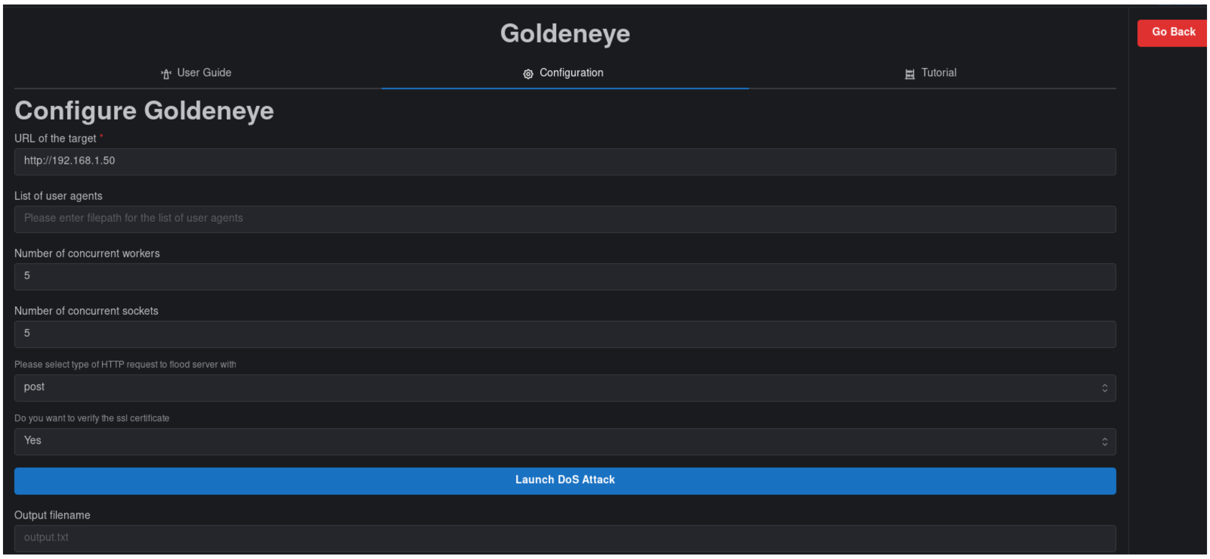
The **User-Agent parameter** was intentionally left blank, allowing GoldenEye to utilise its default embedded User-Agent header. This ensured consistent request formatting and eliminated potential parsing errors associated with externally supplied header lists.

However, if you have any customization or requirements, you can always create a file with a list of agents and enter that file path into the field to be used.

Testing was conducted incrementally, beginning with **5 concurrent workers and 5 sockets** to establish baseline server behavior. You can increase the load to 30 workers and 20 sockets while monitoring CPU utilization, memory consumption, and active TCP connections, however for the purpose of this tutorial I will stick to small numbers. This incremental methodology will ensure controlled observation of performance degradation.

https://datatracker.ietf.org/doc/html/rfc9110#name-user-agents

The **SSL certificate verification option was left enabled**; however, testing was conducted over HTTP rather than HTTPS. Since the target Apache server did not implement TLS encryption, certificate validation was not applicable to the experiment.



Configure Goldeneye

URL of the target *
http://192.168.1.50

List of user agents
Please enter filepath for the list of user agents

Number of concurrent workers
5

Number of concurrent sockets
5

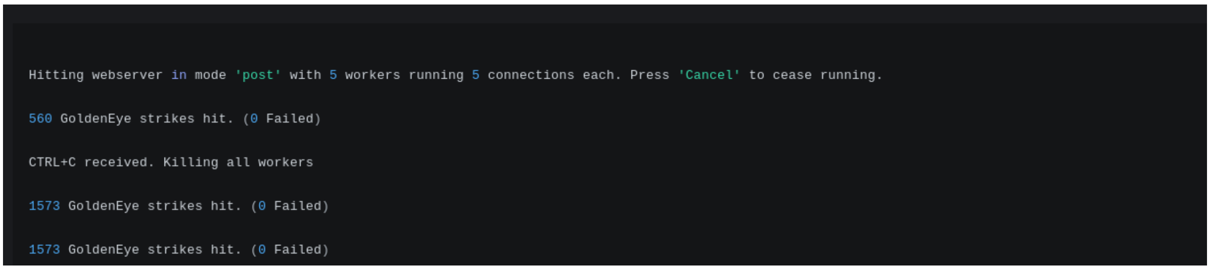
Please select type of HTTP request to flood server with
post

Do you want to verify the ssl certificate
Yes

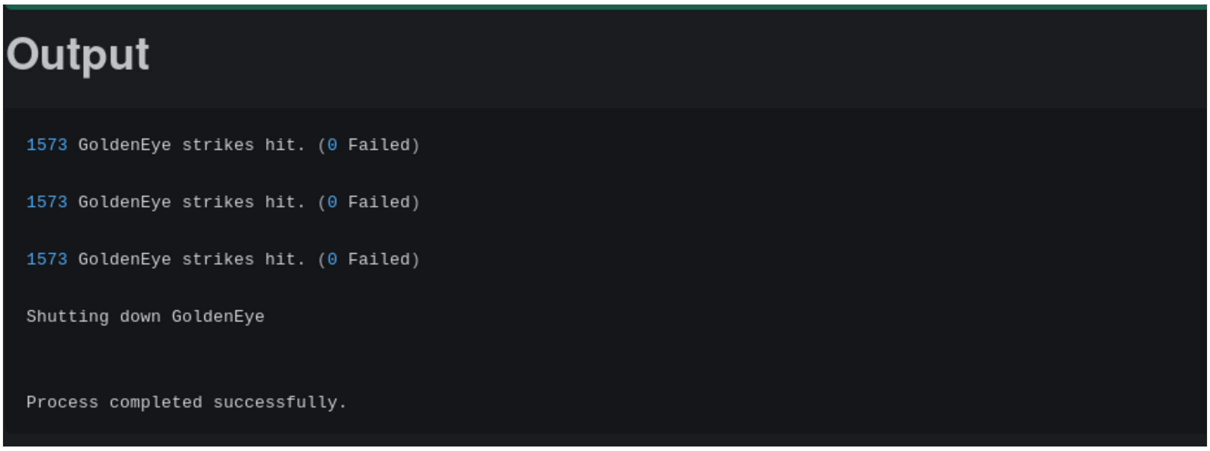
Output filename
output.txt

Now click on Launch DoS Attack!

Step 3: Understanding the output



```
Hitting webserver in mode 'post' with 5 workers running 5 connections each. Press 'Cancel' to cease running.

560 GoldenEye strikes hit. (0 Failed)

CTRL+C received. Killing all workers

1573 GoldenEye strikes hit. (0 Failed)

1573 GoldenEye strikes hit. (0 Failed)
```

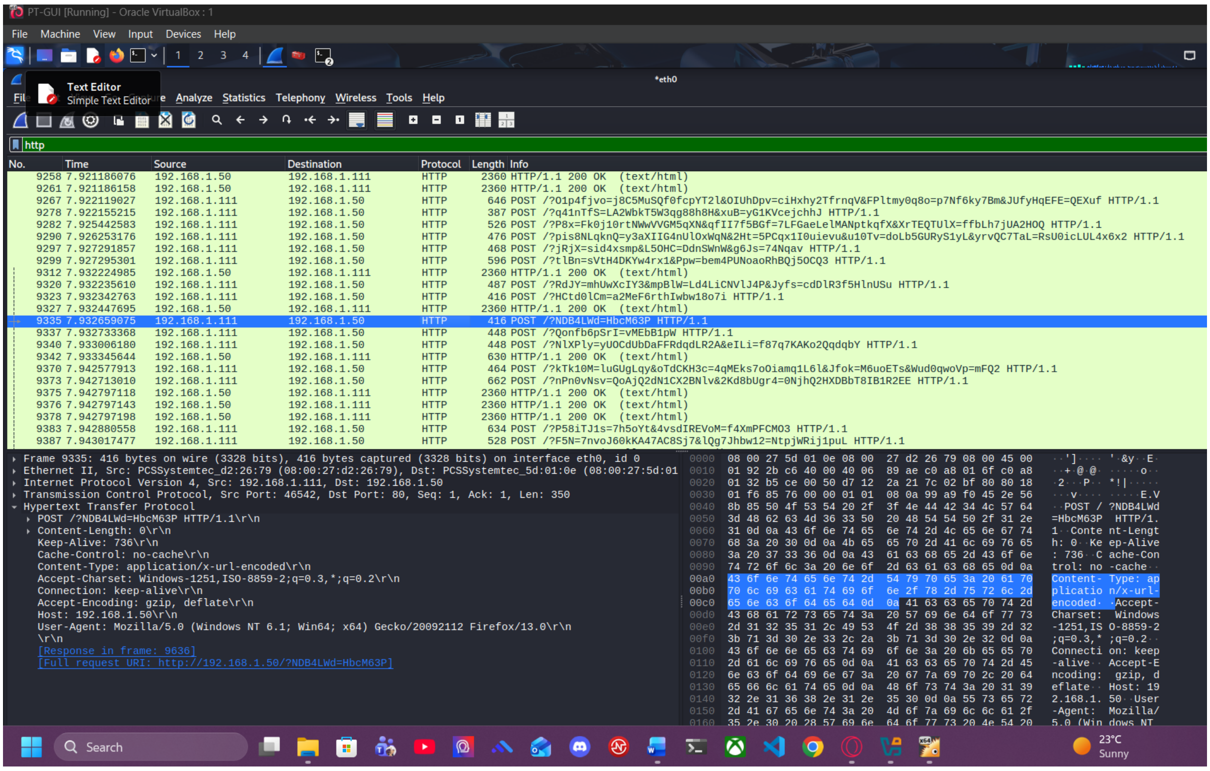


Output

```
1573 GoldenEye strikes hit. (0 Failed)

1573 GoldenEye strikes hit. (0 Failed)

1573 GoldenEye strikes hit. (0 Failed)

Shutting down GoldenEye


Process completed successfully.
```

We have now successfully launched a DoS attack on the Ubuntu server.

We can use wireshark to analyze the traffic:



PT-GUI [Running] - Oracle VirtualBox : 1
File Machine View Input Devices Help
1 2 3 4
*eth0
Text Editor
Simple Text Editor
Analyze Statistics Telephony Wireless Tools Help
http
Search
23°C
Sunny

```
No.   Time        Source         Destination    Protocol Length Info
 9258 7.921186076 192.168.1.50   192.168.1.111  HTTP     2360 HTTP/1.1 200 OK  (text/html)
 9261 7.921186158 192.168.1.50   192.168.1.111  HTTP     2360 HTTP/1.1 200 OK  (text/html)
 9267 7.922119027 192.168.1.111  192.168.1.50   HTTP      646 POST /?O1p4fjvo=j8C5MuSQf0fcpYT2l&OIUhDpv=ciHxhy2TfrnqV&FPltmy0q8o=p7Nf6ky7Bm&JUfyHqEFE=QEXuf HTTP/1.1
 9278 7.922155215 192.168.1.111  192.168.1.50   HTTP      387 POST /?q41nTfS=LA2WbkT5W3qg88h8H&xuB=yG1KVcejchhJ HTTP/1.1
 9282 7.925442583 192.168.1.111  192.168.1.50   HTTP      526 POST /?P8x=Fk0j10rtNWwVVGM5qXN&qfII7f5BGf=7LFGaeLelMANptkqfX&XrTEQTUlX=ffbLh7jUA2HOQ HTTP/1.1
 9290 7.926253176 192.168.1.111  192.168.1.50   HTTP      476 POST /?pis8NLqknQ=y3aXIIG4nUlOxWqN&2Ht=5PCqx1I0uievu&u10Tv=doLb5GURyS1yL&yrvQC7TaL=RsU0icLUL4x6x2 HTTP/1.1
 9297 7.927291857 192.168.1.111  192.168.1.50   HTTP      468 POST /?jRjX=sid4xsmp&L5OHC=DdnSWnW&g6Js=74Nqav HTTP/1.1
 9299 7.927295301 192.168.1.111  192.168.1.50   HTTP      596 POST /?tlBn=sVtH4DKYw4rx1&Ppw=bem4PUNoaoRhBQj5OCQ3 HTTP/1.1
 9312 7.932224985 192.168.1.50   192.168.1.111  HTTP     2360 HTTP/1.1 200 OK  (text/html)
 9320 7.932235610 192.168.1.111  192.168.1.50   HTTP      487 POST /?RdJY=mhUwXcIY3&mpBlW=Ld4LiCNVlJ4P&Jyfs=cdDlR3f5HlnUSu HTTP/1.1
 9323 7.932342763 192.168.1.111  192.168.1.50   HTTP      416 POST /?HCtd0lCm=a2MeF6rthIwbw18o7i HTTP/1.1
 9327 7.932447695 192.168.1.50   192.168.1.111  HTTP     2360 HTTP/1.1 200 OK  (text/html)
 9335 7.932659075 192.168.1.111  192.168.1.50   HTTP      416 POST /?NDB4LWd=HbcM63P HTTP/1.1
 9337 7.932733368 192.168.1.111  192.168.1.50   HTTP      448 POST /?Qonfb6pSrI=vMEbB1pW HTTP/1.1
 9340 7.933006180 192.168.1.111  192.168.1.50   HTTP      448 POST /?NlXPly=yUOCdUbDaFFRdqdLR2A&eILi=f87q7KAKo2QqdqbY HTTP/1.1
 9342 7.933345644 192.168.1.50   192.168.1.111  HTTP      630 HTTP/1.1 200 OK  (text/html)
 9370 7.942577913 192.168.1.111  192.168.1.50   HTTP      464 POST /?kTk10M=luGUgLqy&oTdCKH3c=4qMEks7oOiamq1L6l&Jfok=M6uoETs&Wud0qwoVp=mFQ2 HTTP/1.1
 9373 7.942713010 192.168.1.111  192.168.1.50   HTTP      662 POST /?nPn0vNsv=QoAjQ2dN1CX2BNlv&2Kd8bUgr4=0NjhQ2HXDBbT8IB1R2EE HTTP/1.1
 9375 7.942797118 192.168.1.50   192.168.1.111  HTTP     2360 HTTP/1.1 200 OK  (text/html)
 9376 7.942797143 192.168.1.50   192.168.1.111  HTTP     2360 HTTP/1.1 200 OK  (text/html)
 9378 7.942797198 192.168.1.50   192.168.1.111  HTTP     2360 HTTP/1.1 200 OK  (text/html)
 9383 7.942880558 192.168.1.111  192.168.1.50   HTTP      634 POST /?P58iTJ1s=7h5oYt&4vsdIREVoM=f4XmPFCMO3 HTTP/1.1
 9387 7.943017477 192.168.1.111  192.168.1.50   HTTP      528 POST /?F5N=7nvoJ60kKA47AC8Sj7&lQg7Jhbw12=NtpjWRij1puL HTTP/1.1
```

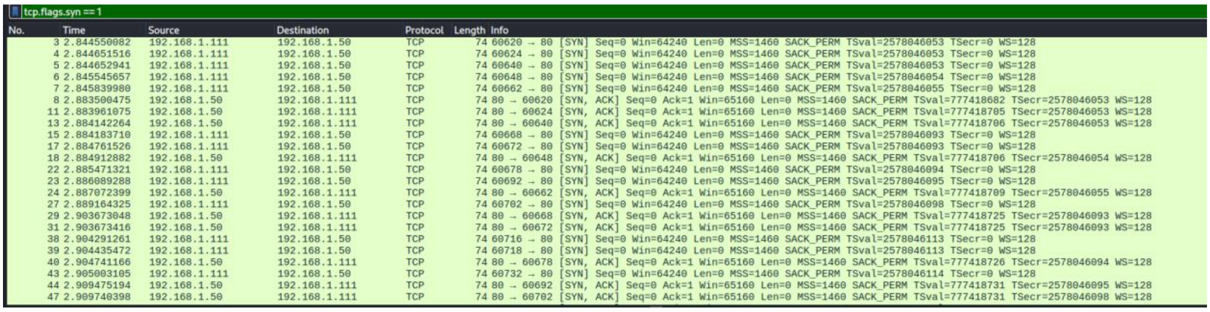
```
› Frame 9335: 416 bytes on wire (3328 bits), 416 bytes captured (3328 bits) on interface eth0, id 0
› Ethernet II, Src: PCSSystemtec_d2:26:79 (08:00:27:d2:26:79), Dst: PCSSystemtec_5d:01:0e (08:00:27:5d:01
› Internet Protocol Version 4, Src: 192.168.1.111, Dst: 192.168.1.50
› Transmission Control Protocol, Src Port: 46542, Dst Port: 80, Seq: 1, Ack: 1, Len: 350
⌄ Hypertext Transfer Protocol
  › POST /?NDB4LWd=HbcM63P HTTP/1.1\r\n
  › Content-Length: 0\r\n
    Keep-Alive: 736\r\n
    Cache-Control: no-cache\r\n
    Content-Type: application/x-url-encoded\r\n
    Accept-Charset: Windows-1251,ISO-8859-2;q=0.3,*;q=0.2\r\n
    Connection: keep-alive\r\n
    Accept-Encoding: gzip, deflate\r\n
    Host: 192.168.1.50\r\n
    User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) Gecko/20092112 Firefox/13.0\r\n
    \r\n
    [Response in frame: 9636]
    [Full request URI: http://192.168.1.50/?NDB4LWd=HbcM63P]
```

```
0000  08 00 27 5d 01 0e 08 00  27 d2 26 79 08 00 45 00   ··']···· '·&y··E·
0010  01 92 2b c6 40 00 40 06  89 ae c0 a8 01 6f c0 a8   ··+·@·@· ·····o··
0020  01 32 b5 ce 00 50 d7 12  2a 21 7c 02 bf 80 80 18   ·2···P·· *!|·····
0030  01 f6 85 76 00 00 01 01  08 0a 99 a9 f0 45 2e 56   ···v···· ·····E.V
0040  8b 85 50 4f 53 54 20 2f  3f 4e 44 42 34 4c 57 64   ··POST / ?NDB4LWd
0050  3d 48 62 63 4d 36 33 50  20 48 54 54 50 2f 31 2e   =HbcM63P  HTTP/1.
0060  31 0d 0a 43 6f 6e 74 65  6e 74 2d 4c 65 6e 67 74   1··Conte nt-Lengt
0070  68 3a 20 30 0d 0a 4b 65  65 70 2d 41 6c 69 76 65   h: 0··Ke ep-Alive
0080  3a 20 37 33 36 0d 0a 43  61 63 68 65 2d 43 6f 6e   : 736··C ache-Con
0090  74 72 6f 6c 3a 20 6e 6f  2d 63 61 63 68 65 0d 0a   trol: no -cache··
00a0  43 6f 6e 74 65 6e 74 2d  54 79 70 65 3a 20 61 70   Content- Type: ap
00b0  70 6c 69 63 61 74 69 6f  6e 2f 78 2d 75 72 6c 2d   plicatio n/x-url-
00c0  65 6e 63 6f 64 65 64 0d  0a 41 63 63 65 70 74 2d   encoded· ·Accept-
00d0  43 68 61 72 73 65 74 3a  20 57 69 6e 64 6f 77 73   Charset:  Windows
00e0  2d 31 32 35 31 2c 49 53  4f 2d 38 38 35 39 2d 32   -1251,IS O-8859-2
00f0  3b 71 3d 30 2e 33 2c 2a  3b 71 3d 30 2e 32 0d 0a   ;q=0.3,* ;q=0.2··
0100  43 6f 6e 6e 65 63 74 69  6f 6e 3a 20 6b 65 65 70   Connecti on: keep
0110  2d 61 6c 69 76 65 0d 0a  41 63 63 65 70 74 2d 45   -alive·· Accept-E
0120  6e 63 6f 64 69 6e 67 3a  20 67 7a 69 70 2c 20 64   ncoding:  gzip, d
0130  65 66 6c 61 74 65 0d 0a  48 6f 73 74 3a 20 31 39   eflate·· Host: 19
0140  32 2e 31 36 38 2e 31 2e  35 30 0d 0a 55 73 65 72   2.168.1. 50··User
0150  2d 41 67 65 6e 74 3a 20  4d 6f 7a 69 6c 6c 61 2f   -Agent:  Mozilla/
0160  35 2e 30 20 28 57 69 6e  64 6f 77 73 20 4e 54 20   5.0 (Win dows NT
```

Here we can also see one of the user-agents being used:

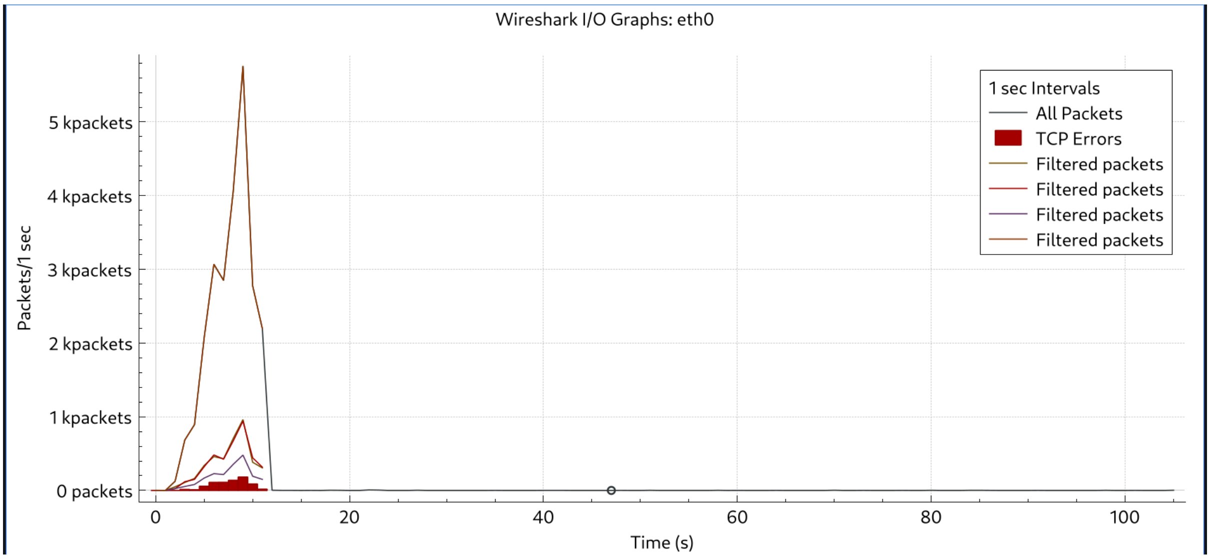
Mozilla/5.0 (Windows NT 6.1; Wind64; x64) Gecko/20092112 Firefox/13.0\r\n\r\n

We see a huge burst in HTTP traffic with HTTP POST.



```
tcp.flags.syn == 1
No.   Time          Source          Destination     Protocol  Length Info
    3 2.844550082   192.168.1.111   192.168.1.50    TCP       74 60620 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2578046053 TSecr=0 WS=128
    4 2.844651516   192.168.1.111   192.168.1.50    TCP       74 60624 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2578046053 TSecr=0 WS=128
    5 2.844652941   192.168.1.111   192.168.1.50    TCP       74 60640 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2578046053 TSecr=0 WS=128
    6 2.845545657   192.168.1.111   192.168.1.50    TCP       74 60648 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2578046054 TSecr=0 WS=128
    7 2.845839980   192.168.1.111   192.168.1.50    TCP       74 60662 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2578046055 TSecr=0 WS=128
    8 2.883500475   192.168.1.50    192.168.1.111   TCP       74 80 → 60620 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=777418682 TSecr=2578046053 WS=128
   11 2.883961075   192.168.1.50    192.168.1.111   TCP       74 80 → 60624 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=777418705 TSecr=2578046053 WS=128
   13 2.884142264   192.168.1.50    192.168.1.111   TCP       74 80 → 60640 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=777418706 TSecr=2578046053 WS=128
   15 2.884183710   192.168.1.111   192.168.1.50    TCP       74 60668 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2578046093 TSecr=0 WS=128
   17 2.884761526   192.168.1.111   192.168.1.50    TCP       74 60672 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2578046093 TSecr=0 WS=128
   18 2.884912882   192.168.1.50    192.168.1.111   TCP       74 80 → 60648 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=777418706 TSecr=2578046054 WS=128
   22 2.885471321   192.168.1.111   192.168.1.50    TCP       74 60678 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2578046094 TSecr=0 WS=128
   23 2.886089288   192.168.1.111   192.168.1.50    TCP       74 60692 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2578046095 TSecr=0 WS=128
   24 2.887072399   192.168.1.50    192.168.1.111   TCP       74 80 → 60662 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=777418709 TSecr=2578046055 WS=128
   27 2.889164325   192.168.1.111   192.168.1.50    TCP       74 60702 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2578046098 TSecr=0 WS=128
   29 2.903673048   192.168.1.50    192.168.1.111   TCP       74 80 → 60668 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=777418725 TSecr=2578046093 WS=128
   31 2.903673416   192.168.1.50    192.168.1.111   TCP       74 80 → 60672 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=777418725 TSecr=2578046093 WS=128
   38 2.904291261   192.168.1.111   192.168.1.50    TCP       74 60716 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2578046113 TSecr=0 WS=128
   39 2.904435472   192.168.1.111   192.168.1.50    TCP       74 60718 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2578046113 TSecr=0 WS=128
   40 2.904741166   192.168.1.50    192.168.1.111   TCP       74 80 → 60678 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=777418726 TSecr=2578046094 WS=128
   43 2.905003105   192.168.1.111   192.168.1.50    TCP       74 60732 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2578046114 TSecr=0 WS=128
   44 2.909475194   192.168.1.50    192.168.1.111   TCP       74 80 → 60692 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=777418731 TSecr=2578046095 WS=128
   47 2.909740398   192.168.1.50    192.168.1.111   TCP       74 80 → 60702 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=777418731 TSecr=2578046098 WS=128
```

We can see all the TCP sessions opening.



I have filtered and generated a graph showing the HTTP flood traffic. So clearly our attack has worked!