

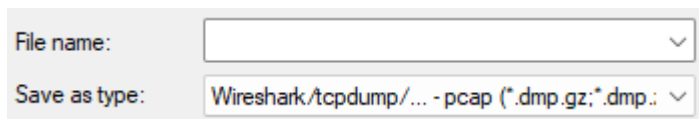
## CyberDefenders: l337 S4uc3 Lab

The following writeup is for [l337 S4uc3 Lab](#) on CyberDefenders, investigating a PCAP and memory dump using Wireshark, NetworkMiner, Brim, and Volatility2. This was the longest individual challenge I have taken part in, and I really enjoyed it. For some questions I did need to look at writeups and check the hints, but I completed it nonetheless. Those who love network and memory forensics should definitely give this a shot.

**Scenario:** Everyone has heard of targeted attacks. Detecting these can be challenging, responding to these can be even more challenging. This scenario will test your network and host-based analysis skills as a soc analyst to figure out the who, what, where, when, and how of this incident. There is sure to be something for all skill levels and the only thing you need to solve the challenge is some l337 S4uc3!

**PCAP:** Development.wse.local is a critical asset for the Wayne and Stark Enterprises, where the company stores new top-secret designs on weapons. Jon Smith has access to the website and we believe it may have been compromised, according to the IDS alert we received earlier today. First, determine the Public IP Address of the webserver?

To find the public IP address, we can analyse the PCAP using NetworkMiner. Before loading the pcap with NetworkMiner, make sure to save the pcapng file as a pcap file so it can be imported into NetworkMiner, you can do so by selecting the following file type when clicking the save as button:



Once loaded into NetworkMiner, you can find the public IP address associated with development.wse.local in the Host tab:



Answer: 74.204.41.73

**PCAP:** Alright, now we need you to determine a starting point for the timeline that will be useful in mapping out the incident. Please determine the arrival time of frame 1 in the "GrrCON.pcapng" evidence file.

This is super simple:

No.	Time
1	2013-09-10 22:51:07

Answer: 22:51:07 UTC

**PCAP: What version number of PHP is the development.wse.local server running?**

Seeing as we know the private IP associated with development.wse.local, we can create a display filter to look for the IP of interest as well as HTTP traffic associated with said address:



If you follow the TCP (or HTTP stream) of any of the results, you can find the version of PHP running in the header:

```
HTTP/1.1 200 OK
Date: Tue, 10 Sep 2013 22:51:28 GMT
Server: Apache/2.2.14 (Ubuntu)
X-Powered-By: PHP/5.3.2-1ubuntu4.20
X-Pingback: http://development.wse.local/xmlrpc.php
Vary: Accept-Encoding
Content-Length: 7174
Connection: close
Content-Type: text/html; charset=UTF-8
```

Answer: 5.3.2

**PCAP: What version number of Apache is the development.wse.local web server using?**

The version of Apache is also visible in the header seen in the above image.

Answer: 2.2.14

**IR: What is the common name of the malware reported by the IDS alert provided?**

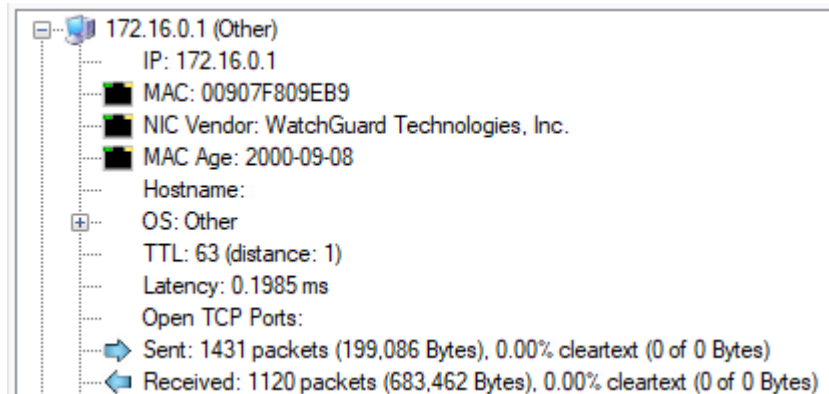
Within the references section of the alert, you can find a link that contains a query to zeus, we can likely infer that this is the name of the malware:

References	
Type	Value
url	<a href="http://www.secureworks.com/research/threats/zeus/?threat=zeus">www.secureworks.com/research/threats/zeus/?threat=zeus</a>

Answer: zeus

**PCAP: Please identify the Gateway IP address of the LAN because the infrastructure team reported a potential problem with the IDS server that could have corrupted the PCAP**

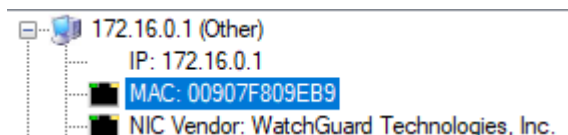
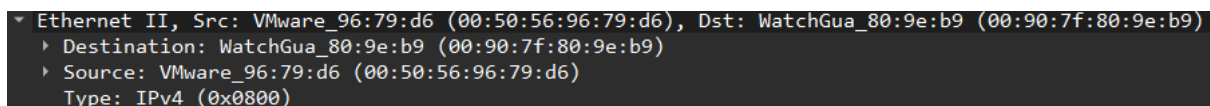
Seeing as the private IP of development.wse.local is 172.16.0.108, we can assume that the gateway address is 172.16.0.1:



If you look into WatchGuard Technologies, you can see that they produce cybersecurity products such as firewalls. Alternatively, if you take a look at Statistics > Conversations and focus on connections between 172.16.0.109 and IP outside the LAN, such as this one for example:



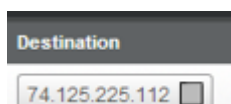
If you check the destination MAC address of 198.47.108.35, we can see that it resolves to 172.16.0.1:



Answer: 172.16.0.1

**IR: According to the IDS alert, the Zeus bot attempted to ping an external website to verify connectivity. What was the IP address of the website pinged?**

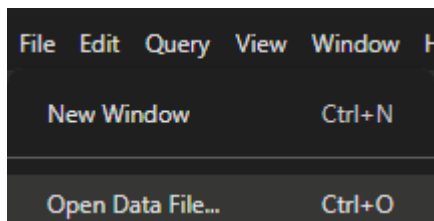
The external website is simply the destination IP address listed on the alert:



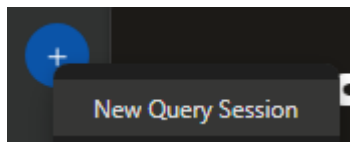
Answer: 74.125.225.112

**PCAP: It's critical to the infrastructure team to identify the Zeus Bot CNC server IP address so they can block communication in the firewall as soon as possible. Please provide the IP address?**

To answer this question, we can use a wonderful tool called Brim. Brim (currently known as Zui) is a network forensics tool that uses Zeek and Suricata to analyse supplied pcaps. To load the PCAP into Zui, simply click File > Open Data File and select the pcap file from the challenge (make sure the same file is not opened with Wireshark as Zui will be unable to get a handle to the file and load it properly):



Then click the plus (+) button and select New Query Session:



After clicking the pool we just imported, you can now analyse the Zeek and Suricata logs/alerts. In this instance, I am going to use a search query to look for Suricata Alerts:

```
1 event_type=="alert" | cut alert.category, dest_ip | uniq | sort -r alert
```

No data.	
alert	dest_ip
> {category: Malware Command and Control Activity Detected}	88.198.6.20
> {category: Malware Command and Control Activity Detected}	88.198.6.20
> {category: Malware Command and Control Activity Detected}	88.198.6.20
> {category: Malware Command and Control Activity Detected}	88.198.6.20
> {category: Malware Command and Control Activity Detected}	88.198.6.20
> {category: Malware Command and Control Activity Detected}	88.198.6.20
> {category: Malware Command and Control Activity Detected}	88.198.6.20
> {category: Malware Command and Control Activity Detected}	88.198.6.20
> {category: Malware Command and Control Activity Detected}	88.198.6.20
> {category: Malware Command and Control Activity Detected}	88.198.6.20
> {category: Malware Command and Control Activity Detected}	88.198.6.20
> {category: Malware Command and Control Activity Detected}	88.198.6.20
> {category: Malware Command and Control Activity Detected}	88.198.6.20
> {category: Malware Command and Control Activity Detected}	88.198.6.20
> {category: Malware Command and Control Activity Detected}	88.198.6.20
> {category: Malware Command and Control Activity Detected}	88.198.6.20
> {category: Malware Command and Control Activity Detected}	88.198.6.20
> {category: Malware Command and Control Activity Detected}	88.198.6.20
> {category: Malware Command and Control Activity Detected}	88.198.6.20
> {category: Malware Command and Control Activity Detected}	88.198.6.20
> {category: Malware Command and Control Activity Detected}	74.125.225.11

Here we can see several alerts related to C2 activity for 88.198.6.20.

Answer: 88.198.6.20

**PCAP: The infrastructure team also requests that you identify the filename of the “.bin” configuration file that the Zeus bot downloaded right after the infection. Please provide the file name?**

There are multiple ways to find this, the most easiest being using NetworkMiner and the Files tab. We can filter for 88.198.6.20 (the C2 address):

Filter keyword: 88.198.6.20						
Frame nr.	Filename	Extension	Size	Source host	S. port	Destination host
3472	NewDesign.jpg.exe.html	html	1 054 B	88.198.6.20 [88.198.6.20]	TCP 80	172.16.0.109
3609	cf.bin	bin	446 B	88.198.6.20 [88.198.6.20]	TCP 80	172.16.0.109

Alternatively, we can create a display filter for the known compromised server (172.16.0.109) and HTTP GET requests:

ip.addr == 172.16.0.109 and http.request.method == GET							
No.	Time	Source	Destination	Destination Port	Protocol	Info	
3224	2013-09-10 22:52:42	172.16.0.109	204.79.197.200	80	HTTP	GET /fd/sa/9_00_0_20	
3258	2013-09-10 22:52:42	172.16.0.109	204.79.197.200	80	HTTP	GET /fd/fb/mulmfg?n=	
3326	2013-09-10 22:52:43	172.16.0.109	204.79.197.200	80	HTTP	GET /rewardsapp/flyo	
3364	2013-09-10 22:52:43	172.16.0.109	204.79.197.200	80	HTTP	GET /s/as/0713125935	
3365	2013-09-10 22:52:43	172.16.0.109	204.79.197.200	80	HTTP	GET /s/rw/trial_xbox	
3409	2013-09-10 22:52:43	172.16.0.109	204.79.197.200	80	HTTP	GET /fd/ls/GLinkPing	
3410	2013-09-10 22:52:43	172.16.0.109	204.79.197.200	80	HTTP	GET /fd/ls/l?IG=8894	
3458	2013-09-10 22:52:43	172.16.0.109	204.79.197.200	80	HTTP	GET /fd/s/a/identity	
3472	2013-09-10 22:52:51	172.16.0.109	88.198.6.20	80	HTTP	GET /NewDesign.jpg.e	
3510	2013-09-10 22:53:06	172.16.0.109	88.198.6.20	80	HTTP	GET /bt.exe HTTP/1.1	
3609	2013-09-10 22:53:39	172.16.0.109	88.198.6.20	80	HTTP	GET /cf.bin HTTP/1.1	

Answer: cf.bin

**PCAP: No other users accessed the development.wse.local WordPress site during the timeline of the incident and the reports indicate that an account successfully logged in from the external interface. Please provide the password they used to log in to the WordPress page around 6:59 PM EST?**

We can use the following display filter that looks for all requests made to wp-admin (default Wordpress login portal location):

ip.addr == 172.16.0.109 and http.request.uri contains "wp-admin"							
No.	Time	Source	Destination	Destination Port	Protocol	Info	
4325	2013-09-10 22:56:32	172.16.0.109	172.16.0.108	80	HTTP	GET /wp-admin/profile.php HTTP/1.1	
4285	2013-09-10 22:56:29	172.16.0.109	172.16.0.108	80	HTTP	GET /wp-admin/ HTTP/1.1	
4198	2013-09-10 22:56:03	172.16.0.109	172.16.0.108	80	HTTP	GET /wp-admin/wp-admin.css HTTP/1.1	

If you follow the HTTP stream of the second request, we can see the password used to login to the Jsmith account:

```
log=Jsmith&pwd=wM812ugu&submit=Login+%C2%BB&redirect_to=wp-admin%2
```

Answer: wM812ugu

**PCAP:** After reporting that the WordPress page was indeed accessed from an external connection, your boss comes to you in a rage over the potential loss of confidential top-secret documents. He calms down enough to admit that the design's page has a separate access code outside to ensure the security of their information. Before storming off he provided the password to the designs page “1qBeJ2Az” and told you to find a timestamp of the access time or you will be fired. Please provide the time of the accessed Designs page?

If you check the Credentials tab in NetworkMiner, we can see the First Login timestamp:

172.16.0.1 172.16.0.108 [74.204.41.73] [development.wise.local] HTTP Cookie parameter: Jsmith,Jsmith 1qBeJ2Az Unknown 2013-09-10 23:04:04 UTC

Answer: 23:04:04 UTC

**PCAP:** What is the source port number in the shellcode exploit? Dest Port was 31708 IDS Signature GPL SHELLCODE x86 inc ebx NOOP

Let's craft a display filter that looks for the destination UDP port 31708:

udp.dstport == 31708

We can then determine the source port to be 39709:

User Datagram Protocol, Src Port: 39709, Dst Port: 31708

Answer: 39709

**PCAP:** What was the Linux kernel version returned from the meterpreter sysinfo command run by the attacker?

If you enter the following display filter, we can find frames that contain the text “sysinfo”;

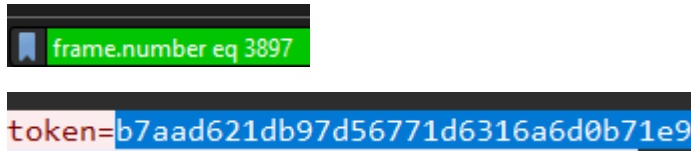
frame matches "sysinfo"						
Time	Source	Destination	Destination Port	Protocol	Info	
4077 2013-09-10 22:55:25	172.16.0.108	172.16.0.1	43614	TCP	[TCP Retransmission]	
4076 2013-09-10 22:55:25	172.16.0.108	172.16.0.1	43614	SMTP	Bind_receiver	
4075 2013-09-10 22:55:25	172.16.0.1	172.16.0.108	4444	TCP	[TCP Retransmission]	
4074 2013-09-10 22:55:25	172.16.0.1	172.16.0.108	4444	TCP	43614 → 4444 [PSH]	
4040 2013-09-10 22:55:09	172.16.0.108	172.16.0.1	43614	TCP	[TCP Retransmission]	
4039 2013-09-10 22:55:09	172.16.0.108	172.16.0.1	43614	SMTP	Bind_receiver	
4038 2013-09-10 22:55:09	172.16.0.1	172.16.0.108	4444	TCP	[TCP Retransmission]	
4037 2013-09-10 22:55:09	172.16.0.1	172.16.0.108	4444	TCP	43614 → 4444 [PSH]	
4004 2013-09-10 22:55:09	172.16.0.108	172.16.0.1	43614	TCP	[TCP Retransmission]	
4003 2013-09-10 22:55:09	172.16.0.108	172.16.0.1	43614	SMTP	Bind_receiver	

Port 4444 is already super suspicious, as it's a default port used by a lot of reverse shells including meterpreter. If you search for sysinfo in the text filter, we can see that the infected machine responds with 2.6.32-38-server after the sysinfo command is executed:

sysinfo....)....8630411363239361145667622246084.....  
...WWW01....W....Linux WWW01 2.6.32-38-server #83-Ubuntu

Answer: 2.6.32-38-server

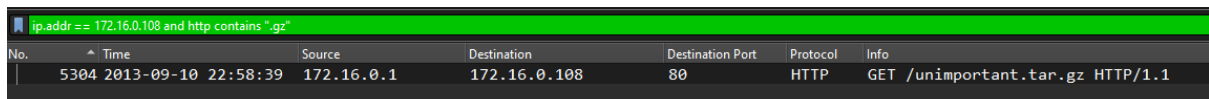
**PCAP: What is the value of the token passed in frame 3897?**



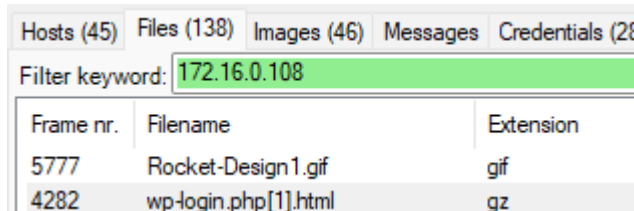
Answer: b7aad621db97d56771d6316a6d0b71e9

**PCAP: What was the tool that was used to download a compressed file from the webserver?**

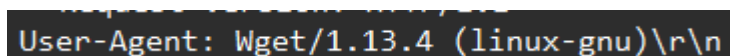
After entering the following display filter, you can find all requests that contain .gz:



I know the compressed file ends with .gz due to looking through the Files tab in NetworkMiner:



If you look at the user-agent of this single request, we can see wget was used to download the compressed file:



Answer: wget

**PCAP: What is the download file name the user launched the Zeus bot?**

Previously we determined that the configuration file Zeus downloaded was cf.bin. The source address for that download was 172.16.0.109. After looking at other requests from this host, we can see one made to bt.exe.

ip.addr == 172.16.0.109 and http.request.method == GET								
No.	Time	Source	Destination	Destination Port	Protocol	Info		
3224	2013-09-10 22:52:42	172.16.0.109	204.79.197.200	80	HTTP	GET	/fd/sa/9_00_0_206	
3258	2013-09-10 22:52:42	172.16.0.109	204.79.197.200	80	HTTP	GET	/fd/fb/mulmfg?n=1	
3326	2013-09-10 22:52:43	172.16.0.109	204.79.197.200	80	HTTP	GET	/rewardsapp/flyou	
3364	2013-09-10 22:52:43	172.16.0.109	204.79.197.200	80	HTTP	GET	/s/as/0713125935/	
3365	2013-09-10 22:52:43	172.16.0.109	204.79.197.200	80	HTTP	GET	/s/rw/trial_xbox.	
3409	2013-09-10 22:52:43	172.16.0.109	204.79.197.200	80	HTTP	GET	/fd/ls/GLinkPing.	
3410	2013-09-10 22:52:43	172.16.0.109	204.79.197.200	80	HTTP	GET	/fd/ls/l?IG=8894a	
3458	2013-09-10 22:52:43	172.16.0.109	204.79.197.200	80	HTTP	GET	/fd/s/a/identity6	
3472	2013-09-10 22:52:51	172.16.0.109	88.198.6.20	80	HTTP	GET	/NewDesign.jpg.ex	
3510	2013-09-10 22:53:06	172.16.0.109	88.198.6.20	80	HTTP	GET	/bt.exe HTTP/1.1	

Unfortunately I cant download the file and hash it to check it on VirusTotal.

Answer: bt.exe

**Memory: What is the full file path of the system shell spawned through the attacker's meterpreter session?**

First you need to move the supplied profile to the volatility/plugins/overlays/linux directory like as follows:

```
cd /usr/local/lib/python2.7/dist-packages/volatility/plugins/overlays/linux/

remnux@remnux:/usr/local/lib/python2.7/dist-packages/volatility/plugins/overlays/linux$ sudo mv /home/remnux/Documents/Ubuntu10-4/DFIRwebsvr.zip /usr/local/lib/python2.7/dist-packages/volatility/plugins/overlays/linux/

remnux@remnux:/usr/local/lib/python2.7/dist-packages/volatility/plugins/overlays/linux$ ls
DFIRwebsvr.zip  elf.py  elf.pyc  __init__.py  __init__.pyc  linux.py  linux.pyc
```

We can now use the linux\_psaux plugin to list processes and their command line arguments:

```
vol.py -f webserver.vms --profile=LinuxDFIRwebsvr64 linux_psaux
```

We can see that the full file path of the system shell spawned is /bin/sh:

```
1274  33  33  sh -c /bin/sh
1275  33  33  /bin/sh
```

Answer: /bin/sh

**Memory: What is the Parent Process ID of the two 'sh' sessions?**

To find the PPID of the two sh session, we can use the linux\_pstree plugin:

```
vol.py -f webserver.vms --profile=LinuxDFIRwebsvr64 linux_pstree

. apache2 1032
.. apache2 1040 33
.. apache2 1042 33
... sh 1274 33
.... sh 1275 33
```



Here we can see that the PPID of the two sh session is 1042.

Answer: 1042

### Memory: What is the latency\_record\_count for PID 1274?

```
remnux@remnux:~/Documents/Ubuntu10-4$ vol.py -f webserver.vms --profile=LinuxDFIRwebsvr64 linux_volshell
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cryptography
  is deprecated in cryptography, and will be removed in the next release.
  from cryptography.hazmat.backends.openssl import backend
Current context: process init, pid=1 DTB=0x176ba000
Welcome to volshell! Current memory image is:
file:///home/remnux/Documents/Ubuntu10-4/webserver.vms
To get help, type 'hh()'
>>> cc(pid=1274)
Current context: process sh, pid=1274 DTB=0x6d94000
>>> proc()
[task_struct task_struct] @ 0xFFFF880006DD8000
>>> dti("task_struct",0xFFFF880006DD8000)
```

```
0x7b0 : latency_record_count      0
0x7b8 : latency_record            -
```

In the above image, we can see that the linux\_volshell plugin is being used. This launches an interactive Volatility shell for manual forensic analysis. The cc commands switches the context to a process with a PID of 1274. Proc() retrieves information about the current process, and the task\_struct line is used to show a detailed structure of the specified address.

Answer: 0

### Memory: For the PID 1274, what is the first mapped file path?

To find the first mapped file path for the PID 1274, we can use the linux\_proc\_maps plugin, which shows details of process memory, including heaps, stacks, and shared libraries.

```
vol.py -f webserver.vms --profile=LinuxDFIRwebsvr64 linux_proc_maps --pid 1274
```

```
File Path
-----
/bin/dash
/bin/dash
/bin/dash
```

As you can see, the first mapped file path is /bin/dash.

Answer: /bin/dash

**Memory:What is the md5hash of the receive.1105.3 file out of the per-process packet queue?**

To find the md5hash of the receive.1105.3 file out of the per-process packet queue, we can use the linux\_pkt\_queues plugin. As stated in the volatility wiki, “When a socket is attempting to send packets out onto the network at rates that the network cannot handle, or when the kernel has processed received packets that the corresponding userland service has not yet picked up, these packets are placed on per-socket send and receive queues.

The linux\_pkt\_queues plugin enumerates these queues for each active socket in the kernel and writes the recovered packets to disk.”

```
mkdir packets
vol.py -f webserver.vms --profile=LinuxDFIRwebsvr64 linux_pkt_queues -D packets

remnux@remnux:~/Documents/Ubuntu10-4$ md5sum packets/receive.1105.3
184c8748cfcfe8c0e24d7d80cac6e9bd  packets/receive.1105.3
```

Answer: 184c8748cfcfe8c0e24d7d80cac6e9bd