Challenge: [Malware Traffic Analysis 3 Lab]

Platform: CyberDefenders

Category: Network Forensics

Difficulty: Medium

Tools Used: Wireshark, Zui, NetworkMiner, VirusTotal, GHex, pesec, Python

Summary: This lab involved investigating a Windows host compromised by the Angler exploit kit (EK). It involves investigating a PCAP to identify suspicious activity like malicious redirects, obfuscated payload delivery, and more. I found this challenge quite difficult, some questions were basic, especially the first 8, however, the rest of the questions required a solid understanding of malware deobfuscation.

Scenario: The attached PCAP belongs to an Exploitation Kit infection. As a security blue team member, analyze it using your favorite tool and answer the challenge questions.

What is the IP address of the infected Windows host?

TLDR: Navigate to Statistics > Conversations > IPv4 and look at what host appears in every conversation. You can also verify by navigating to the hosts tab within NetworkMiner and seeing what host is running Windows.

When approaching network forensics, I like to begin by baselining the traffic, which involves getting an understanding of the traffic within the PCAP (protocol usage, traffic volume, hosts, etc). Wireshark provides a great feature called Statistics that enables you to do so. Let's start by scoping out the protocols within the PCAP by navigating to Statistics > Protocol Hierarchy:

Protocol	Percent Packets	Packets	Percent Bytes	Bytes
Frame	100.0	5141	100.0	3428271
Ethernet	100.0	5141	2.4	80998
Internet Protocol Version 4	100.0	5141	3.0	102820
User Datagram Protocol	3.8	197	0.0	1576
Dynamic Host Configuration Protocol	0.0	2	0.0	619
Domain Name System	3.8	195	0.6	18932
Transmission Control Protocol	96.2	4944	94.0	3223326
Transport Layer Security	18.1	931	42.5	1455364
Hypertext Transfer Protocol	4.5	232	42.6	1461858
eXtensible Markup Language	0.0	2	2.2	76708
Portable Network Graphics	0.2	9	0.6	19196
Media Type	0.1	6	5.2	179151
Line-based text data	1.1	58	49.6	1699061
JavaScript Object Notation	0.0	1	0.0	65
JPEG File Interchange Format	0.1	6	5.8	198473
HTML Form URL Encoded	0.0	1	0.0	219
Data	0.0	1	2.5	84705
Compuserve GIF	0.3	15	0.3	9987

We can see that most of the traffic is HTTP. Let's now navigate to Statistics > Conversations > IPv4 to get an understanding of the hosts within the PCAP:

Ethernet · 2 | IPv4 · 45 | IPv6 | TCP · 137 | UDP · 37

Address A	Address B	Packets ▴	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration
192.168.137.62	216.9.81.189	904	612 kB	397	39 kB	507	573 kB	14.894231	107.5306
192.168.137.62	74.125.232.69	641	511 kB	267	20 kB	374	490 kB	6.743227	115.6812
192.168.137.62	209.126.97.209	606	511 kB	238	17 kB	368	493 kB	47.859518	80.5823
192.168.137.62	173.194.116.111	581	418 kB	234	45 kB	347	373 kB	1.200144	75.6880
192.168.137.62	192.99.198.158	300	252 kB	111	8 kB	189	244 kB	33.607766	92.1133
192.168.137.62	192.168.137.1	196	27 kB	36	3 kB	160	24 kB	0.997513	63.8571
192.168.137.62	173.194.116.109	188	149 kB	80	19 kB	108	130 kB	22.469684	74.9240
192.168.137.62	54.230.94.110	176	142 kB	69	6 kB	107	136 kB	21.025344	100.8712
192.168.137.62	173.194.116.121	157	110 kB	67	11 kB	90	100 kB	31.834763	65.0569
192.168.137.62	173.194.78.94	119	87 kB	48	4 kB	71	83 kB	5.216420	91.6764
192.168.137.62	198.41.215.186	105	85 kB	40	3 kB	65	82 kB	60.910963	51.2923
192.168.137.62	23.55.232.11	90	82 kB	32	2 kB	58	80 kB	44.268819	114.3993
192.168.137.62	54.231.18.161	78	61 kB	31	2 kB	47	59 kB	53.880615	68.5438
192.168.137.62	173.194.67.94	68	41 kB	30	3 kB	38	38 kB	4.810966	72.6168
192.168.137.62	94.31.29.154	67	48 kB	27	2 kB	40	46 kB	31.647157	14.1300
192.168.137.62	173.194.116.98	57	31 kB	28	3 kB	29	28 kB	55.172986	67.2515
192.168.137.62	2.21.90.227	51	10 kB	28	4 kB	23	7 kB	62.457326	24.4317
192.168.137.62	54.231.244.0	51	18 kB	24	3 kB	27	15 kB	54.264355	17.6206
192.168.137.62	74.125.136.95	51	37 kB	21	2 kB	30	35 kB	22.366545	65.0785
192.168.137.62	173.194.78.103	47	20 kB	25	3 kB	22	17 kB	5.928611	91.4651
192.168.137.62	74.125.232.70	46	21 kB	23	2 kB	23	19 kB	31.434478	65.4576
192.168.137.62	2.21.99.146	41	12 kB	22	4 kB	19	8 kB	62.457328	69.4408
192.168.137.62	54.243.173.6	38	11 kB	20	2 kB	18	9 kB	54.264743	67.6306
192.168.137.62	74.125.71.100	38	19 kB	20	5 kB	18	14 kB	23.325752	99.0987
192.168.137.62	74.125.206.94	35	22 kB	16	2 kB	19	19 kB	33.747616	63.3364
192.168.137.62	91.225.248.129	35	8 kB	19	2 kB	16	6 kB	62.849894	69.5394
192.168.137.62	50.16.196.229	34	12 kB	18	2 kB	16	9 kB	60.602145	71.2960
192.168.137.62	37.252.170.143	33	8 kB	18	3 kB	15	5 kB	57.628892	19.7986
192.168.137.62	74.125.232.84	31	19 kB	14	1 kB	17	17 kB	22.365728	65.0793
192.168.137.62	74.125.232.90	28	7 kB	17	3 kB	11	4 kB	63.285404	69.1039
192.168.137.62	217.163.21.35	24	4 kB	14	2 kB	10	2 kB	63.468901	23.6095
192.168.137.62	64.233.167.95	23	6 kB	14	3 kB	9	2 kB	32.696473	64.6972
192.168.137.62	152.163.13.77	22	3 kB	13	2 kB	9	2 kB	63.467331	23.9776
192.168.137.62	23.21.77.81	21	3 kB	12	2 kB	9	1 kB	23.618013	98.8062
192.168.137.62	173.241.240.220	20	3 kB	10	1 kB	10	1 kB	63.286069	0.8535
192.168.137.62	173.201.198.128	19	2 kB	11	1 kB	8	976 bytes	32.537499	54.9074
192.168.137.62	23.55.234.99	18	3 kB	11	1 kB	7	2 kB	63.497718	68.3997
192.168.137.62	23.235.43.166	18	3 kB	10	1 kB	8	2 kB	63.467962	13.6064

We can notice a pattern of 192.168.137.62 being present in all conversations. This suggests that it's the target host, along with the fact that's it's within the private IP address space. Furthermore, if you navigate to NetworkMiner, it detects this host as a Windows machine:

- 192.168.137.62 [StevePowers-PC] (Windows)
  - IP: 192.168.137.62
  - MAC: 001B21CAFED7
  - NIC Vendor: Intel Corporate
  - MAC Age: 2007-01-16
  - Hostname: StevePowers-PC
  - OS: Windows

Answer: 192.168.137.62

What is the Exploit kit (EK) name? (two words)

TLDR: Look for any suspicious file downloads, you can find a malware alert for a file within the notice logs from Zui.

An Exploit Kit (EK) is designed to exploit vulnerabilities in software like web browsers to gain unauthorised access to a system. These EKs are commonly used in drive-by download attacks, whereby a user is infected by visiting a compromised website. Once EKs exploit a vulnerability, they often drop additional malware payloads, like ransomware, crypto miners, and more.

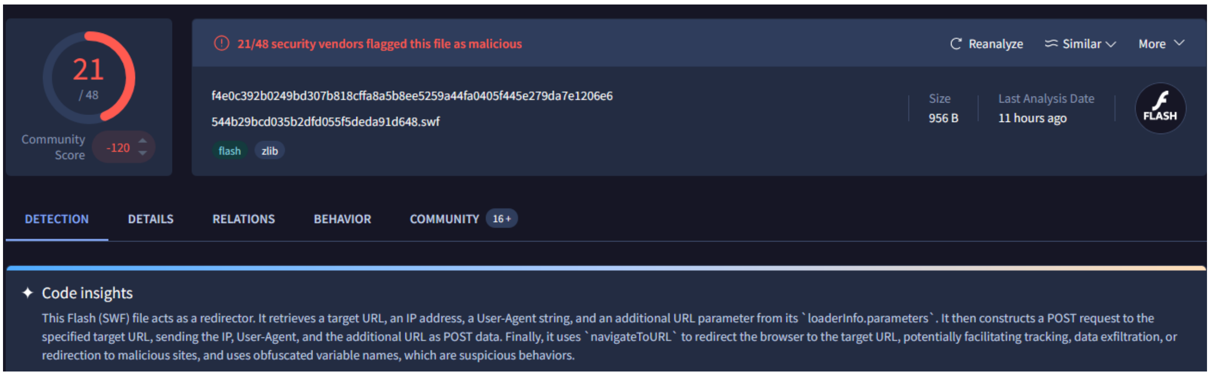
If you inspect the notice alerts in Zui:

- `_path=="notice"`

One notice really stands out, as it's a malware detection for a flash file:


```
{
  _path: notice,
  ts: 2014-12-04T18:27:31.306711Z,
  uid: "CUUw2SoQ7vTaEYpQe",
  id: {
    orig_h: 192.168.137.62,
    orig_p: 50450 (port=(uint16)),
    resp_h: 93.114.64.118,
    resp_p: 80 (port=(uint16))
  },
  fuid: "FVnXWT3w5DR5t2BhU6",
  file_mime_type: "application/x-shockwave-flash",
  file_desc: "http://adstairs.ro/544b29bcd035b2dfd055f5deda91d648.swf",
  proto: "tcp" (zenum),
  note: "TeamCymruMalwareHashRegistry::Match" (zenum),
  msg: "Malware Hash Registry Detection rate: 27%  Last seen: 2019-01-03 19:40:14",
  sub: "https://www.virustotal.com/gui/search/af552b8d2cc1445dee56c8bd830fb322fc9e583a",
  src: 192.168.137.62,
  dst: 93.114.64.118,
  p: 80 (port=(uint16)),
```


If you visit the provided VirusTotal link, we can see that it receives 21/48 detections which suggests that it is malicious. We are also given some code insights which are helpful:



Exploit bits (EKs) frequently abuse SWF files because of how Flash executed code and interacted with browsers. Exploit kits like Angler, Nuclear, Magnitude, and Neutrino have been observed hosting malicious SWFs on websites. When a victim visits a compromised site or malicious ad (malvertising), the EK profiles the browser and plugin versions, if flash is outdated, it delivers a malicious SWF exploiting some sort of vulnerability, upon success, it can drop and execute a payload like ransomware, etc.

If you search for this hash online, you will come across multiple posts. In this instance, I came across a post which talked about the Angler exploit kit:

Re: [Emerging-Sigs] Malicious swf sig

From: James Lay <jlay () slave-tothe-box net>
Date: Wed, 10 Dec 2014 14:29:29 -0700

```
On 2014-12-10 01:58 PM, Will Metcalf wrote:

| Will check into
those on the ET side. For some reason I think I've seen leading dir
sometimes could be wrong though..
| Regards,
| Will
|
| On Wed, Dec
10, 2014 at 1:09 PM, James Lay <jlay () slave-tothe-box net [15]> wrote:
|
| |
| | On 2014-12-10 11:11 AM, Shefferman, Ian wrote:
| |
| | | So far I've
seen these Flash files used primarily (and probably
| | | solely) to
redirect to Angler exploit kit "32x32" gates. A typical
| | | chain is as
follows:
| | |
| | | (Source:
http://malware-traffic-analysis.net/2014/10/30/index.html [1])
| | |
```

Answer: Angler EK

What is the FQDN that delivered the exploit kit?

TLDR: Using Zui, search for downloaded flash files. Once you identify the IP associated with the exploit kit download, you can filter for this IP within the HTTP logs to identify the FQDN.

We know that one of the FQDN's that delivered a malicious SWF file is adstairs.ro:

```
GET /544b29bcd035b2dfd055f5deda91d648.swf HTTP/1.1
Accept: */*
Accept-Language: en-US
Referer: http://www.earsurgery.org/
x-flash-version: 11,4,402,287
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)
Host: adstairs.ro
Connection: Keep-Alive
```

However, this didn't deliver the exploit kit. Using the following query in Zui:

- `_path=="files" mime_type=="application/x-shockwave-flash" | cut ts, id.orig_h, id.resp_h, id.resp_p, source, mime_type, md5`

We can see all SWF files observed within the PCAP. This returns two results, one seen previously, and a new one from 192.99.198.158:

ts	id	source	mime_type
2014-12-04T18:27:49.988502Z	› {orig_h: 192.168.137.62, resp_h: 192.99.198.158 …+1 }	HTTP	application/x-shockwave-flash
2014-12-04T18:27:27.786607Z	› {orig_h: 192.168.137.62, resp_h: 93.114.64.118 …+1 }	HTTP	application/x-shockwave-flash

Using the following query:

- `_path=="http" id.resp_h==192.99.198.158 | cut ts, host, uri, referrer`

We can find the FQDN associated with this IP:

ts	host	uri
2014-12-04T18:27:55.49171Z	qwe.mvdunalterableairreport.net	/i_JnzurEICi4FQgJPm53aItUwat9SekFTU9d2KwmkCuLN2dPiuEjgSqCgiP8yIMk
2014-12-04T18:27:54.443893Z	qwe.mvdunalterableairreport.net	/2nAY-xQvz4JQqjC66P7SgvZGdjIrMJheyLnsQvXjBrLitaA-_K4Uh45BR0unHcom
2014-12-04T18:27:49.261946Z	qwe.mvdunalterableairreport.net	/xPF_HAXN7TK9bMAgBjZDwQzO1-Wf5GvrN5_lIReIhbrhqHAlWyTDbaOBMPWitjnX
2014-12-04T18:27:49.204871Z	qwe.mvdunalterableairreport.net	/2fNECYxvaRhNgivqycm7mfyO70tDCcYnnkyzNqJ-9ax5HSDcERPdxHf3Ow1szmYw
2014-12-04T18:27:34.889035Z	qwe.mvdunalterableairreport.net	/680VBFhpBNBJOYXebSxgwLrtbh3g6JFUllqksWFSsGshhwsguyNL26MGul2oZ3b8
2014-12-04T18:27:29.658351Z	qwe.mvdunalterableairreport.net	/3xdz3bcxc8

Answer: qwe.mvdunalterableairreport.net

What is the redirect URL that points to the exploit kit landing page?

TLDR: Filter for HTTP traffic of the domain hosting the exploit kit. Make sure to look at the referrer field to find the redirect URL.

The redirect URL indicates the webpage the user initially visited, which subsequently redirected them to the site hosting the exploit kit. By using the same HTTP query as done in the previous question, we can see the referrer URL:


```
{
  ts: 2014-12-04T18:27:29.658351Z,
  host: "qwe.mvdunalterableairreport.net",
  uri: "/3xdz3bcxc8",
  referrer: "http://lifeinsidedetroit.com/02024870e4644b68814aadfbb58a75bc.php?q=e8bd3799ee8799332593b0b9caa1f426"
}
```


Alternatively, if you use the following Wireshark display filter:

- `http.host=="qwe.mvdunalterableairreport.net"`

And view the HTTP stream for packet number 2272, we can find the referrer:

```
GET /3xdz3bcxc8 HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Referer: http://lifeinsidedetroit.com/02024870e4644b68814aadfbb58a75bc.php?q=e8bd3799ee8799332593b0b9caa1f426
Accept-Language: en-US
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)
Accept-Encoding: gzip, deflate
Host: qwe.mvdunalterableairreport.net
Connection: Keep-Alive
```

Within the same TCP stream, we can also find a very suspicious script.

Answer:
http://lifeinsidedetroit.com/02024870e4644b68814aadfbb58a75bc.php?q=e8bd3799ee87993 32593b0b9caa1f426

What is the FQDN of the compromised website?

TLDR: Look at the referrer for adstrairs.ro which was observed delivering a malicious flash file. This will enable you to identify a compromised WordPress site that redirects users to the malicious site.

Let's start by getting an understanding of all the hosts visited in the PCAP by using the following Zui query:

- `_path=="http" | count() by host | sort -r count`

host	count
www.earsurgery.org	29
pagead2.googlesyndication.com	13
qwe.mvdunalterableairreport.net	6
googleads.g.doubleclick.net	5

We can see many visits to www.earsurgery.org. If you recall in the first question, we noticed that the infected Windows VM had a conversation which involved 904 packets with 216.9.81.189. If we filter for this IP in Wireshark:

- `ip.addr==192.168.137.62 && ip.addr==216.9.81.189 && http`

We can see that this is a WordPress site for www.earsurgery.org:

Source	Destination	Protocol	Length	Host	Info
192.168.137.62	216.9.81.189	HTTP	524	www.earsurgery.org	GET / HTTP/1.1
192.168.137.62	216.9.81.189	HTTP	431	www.earsurgery.org	GET /wp-content/plugins/easy-image-display/css/colorbox.css?ver=4.0 HTTP/1.1
192.168.137.62	216.9.81.189	HTTP	410	www.earsurgery.org	GET /wp-includes/css/dashicons.min.css?ver=4.0 HTTP/1.1
192.168.137.62	216.9.81.189	HTTP	447	www.earsurgery.org	GET /wp-includes/js/jquery/jquery-migrate.min.js?ver=1.2.1 HTTP/1.1
192.168.137.62	216.9.81.189	HTTP	423	www.earsurgery.org	GET /wp-content/plugins/page-list/css/page-list.css?ver=4.2 HTTP/1.1
192.168.137.62	216.9.81.189	HTTP	409	www.earsurgery.org	GET /wp-content/themes/esic/style.css?ver=4.0 HTTP/1.1

To determine if this website is compromised, let's think back to the beginning. In the second question, we discovered that adstrairs.ro was used to deliver a malicious SWF file. Using the following query:

- `_path=="http" host=="adstairs.ro" | cut host, uri, referrer`

We can see that the referrer for this request is http://www.earsurgery.org/:

host	uri	referrer
adstairs.ro	/544b29bcd035b2dfd055f5deda91d648.swf	http://www.earsurgery.org/

Due to this, it's clear that earsurgery.org has been compromised in some manner.

Answer: earsurgery.org

Which TCP stream shows the malware payload being delivered? Provide stream number

If you navigate to File > Export Objects > HTTP and filter for the exploit kit domain, we can find an interesting octet-stream:

Packet	Hostname	Content Type	Size	Filename
2775	qwe.mvdunalterableairreport.net	text/html	94 kB	3xdz3bcxc8
2957	qwe.mvdunalterableairreport.net	application/octet-stream	84 kB	680VBFhpBNBJOYXebSxgwLrtbh3g6JFUllqksWFSsGshhwsguyNL26MGul2oZ3b8
3728	qwe.mvdunalterableairreport.net	text/html	46 kB	xPF_HAXN7TK9bMAgBjZDwQzO1-Wf5GvrN5_lIRelhbrhqHAlWyTDbaOBMPWitjnX
3853	qwe.mvdunalterableairreport.net	application/x-shockwave-flash	44 kB	2fNECYxvaRhNgivqycm7mfyO70tDCcYnnkyzNqJ-9ax5HSDcERPdxHf3Ow1szmYw
4221	qwe.mvdunalterableairreport.net	text/html	0 bytes	2nAY-xQvz4JQqjC66P7SgvZGdjIrMJheyLnsQvXjBrLitaA-_K4Uh45BR0unHcom
4258	qwe.mvdunalterableairreport.net	text/html	0 bytes	i_JnzurEICi4FQgJPm53altUwat9SekFTU9d2KwmkCuLN2dPiuEjgSqCgiP8yIMk

Using the following display filter:

- `frame.number==2967`

We can see the payload being downloaded at 2014-12-04 18:27:36 UTC:

Source	Destination	Protocol	Length	Info
192.99.198.158	192.168.137.62	HTTP	326	HTTP/1.1 200 OK

If you right-click this packet and select Follow > HTTP stream, we can find the stream number:

Answer: 80

What is the IP address of the C&C server?

TLDR: Navigate to Statistics > Conversations > IPv4 and look for large conversations. Focus on when these conversations began, and whether it matches with when the malicious payload, identified previously, was downloaded.

If you navigate to Statistics > Conversations > IPv4, we can see what hosts the victim has communicated with:

Ethernet · 2 | IPv4 · 45 | IPv6 | TCP · 137 | UDP · 37

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration
192.168.137.62	216.9.81.189	904	612 kB	397	39 kB	507	573 kB	14.894231	107.5306
192.168.137.62	74.125.232.69	641	511 kB	267	20 kB	374	490 kB	6.743227	115.6812
192.168.137.62	209.126.97.209	606	511 kB	238	17 kB	368	493 kB	47.859518	80.5823
192.168.137.62	173.194.116.111	581	418 kB	234	45 kB	347	373 kB	1.200144	75.6880
192.168.137.62	192.99.198.158	300	252 kB	111	8 kB	189	244 kB	33.607766	92.1133
192.168.137.62	192.168.137.1	196	27 kB	36	3 kB	160	24 kB	0.997513	63.8571
192.168.137.62	173.194.116.109	188	149 kB	80	19 kB	108	130 kB	22.469684	74.9240
192.168.137.62	54.230.94.110	176	142 kB	69	6 kB	107	136 kB	21.025344	100.8712
192.168.137.62	173.194.116.121	157	110 kB	67	11 kB	90	100 kB	31.834763	65.0569
192.168.137.62	173.194.78.94	119	87 kB	48	4 kB	71	83 kB	5.216420	91.6764
192.168.137.62	198.41.215.186	105	85 kB	40	3 kB	65	82 kB	60.910963	51.2923

The conversations in the image above are relatively large. 216.9.81.189 is the compromised WordPress site, 74.125.232.69 is unknown, if you view this conversation, we can see that it beings at 18:27:02, which is before the malware payload was downloaded so it's likely not the C2 server. 209.126.97.209 on the other hand, was reached out to by the victim machine at 18:27:43, just seven seconds after the malware payload was downloaded. Due to this, and the large number of packets in this conversation, it's safe to assume that this is the C2 server.

- `ip.addr==192.168.137.62 && ip.addr==209.126.97.209`

We can also see that this server has an invalid SSL cert within the Zui logs:

- `_path=="notice" | cut id.orig_h, id.resp_h, id.resp_p, note, msg`

```
{orig_h: 192.168.137.62, resp_h: 209.126.97.209 …+1 }  SSL::Invalid_Server_Cert
```

Answer: 209.126.97.209

The malicious domain served a ZIP archive. What is the name of the DLL file included in this archive?

TLDR: Analyse the PCAP with NetworkMiner and filter for the keyword "zip" in the Files tab to find the zip archive.

There are multiple ways of approaching this question, the easiest being NetworkMiner. If you navigate to the Files tab and search for zip in the keyword filter, you can see one result for a zip file downloaded from the exploit kit website qwe.mvdunalterableairreport.net:

NetworkMiner 2.9.0

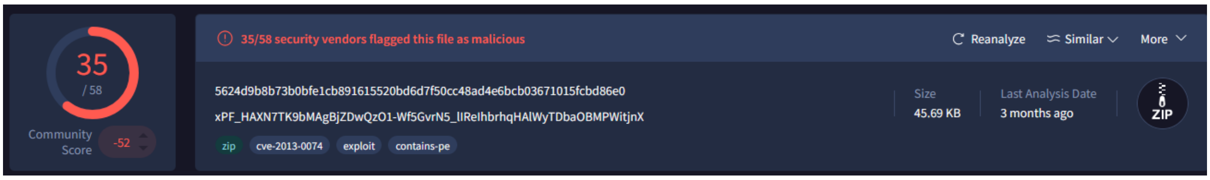
File Tools Help

Hosts (98) | Files (194) | Images (32) | Messages | Credentials (40) | Sessions (152) | DNS (244) | Parameters (4284) | Keywords | Anomalies |

Filter keyword: zip Case sensitive ExactPhrase

Frame nr.	Filename	Extension	Size	Source host	S. port	Destination host	D. port	Protocol	Timestamp
3296	xPF_HAXN7TK9bMAgBjZD.html	zip	46 789 B	192.99.198.158 [qwe.mvdunalterableairreport.net]	TCP 80	192.168.137.62	TCP 50467	HttpGetChunked	2014-12-04 18:27:49 UTC

If you right-click the result and select file details, you can extract the MD5 hash. Upon submitting this hash to VirusTotal, you can see that it receives a high number of detections:



In the comments section, a user mentions Angler EK. To extract this file, right click the result and select open folder:

Alternatively, using the following filter in Wireshark:

- `ip.src==192.168.137.62 && http.request.method==GET and http.request.uri contains "xPF"`

We can pinpoint the exact packet that contains the zip archive. If you follow the HTTP stream of the single result and scroll all the way down to the bottom, you can find the zip archive:

```
</html>GET /xPF_HAXN7TK9bMAgBjZDwQzO1-Wf5GvrN5_lIReIhbrhqHAlWyTDbaOBMPWitjnX HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)
Host: qwe.mvdunalterableairreport.net
Connection: Keep-Alive

HTTP/1.1 200 OK
Server: nginx/1.2.1
Date: Thu, 04 Dec 2014 18:27:52 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
Cache-Control: no-cache, must-revalidate, max-age=1
Expires: Sat, 26 Jul 1997 05:00:00 GMT
Last-Modified: Sat, 26 Jul 2040 05:00:00 GMT
Pragma: no-cache
Content-Encoding: gzip

PK........~.E-o'.....y.......AppManifest.xaml..A..0....s...$qpf.XA..."*.k.1.?..#.gs....
.F..X...F.q.r..D.Z..n..t...M.......J.n.>.'m.....T..djc..N.8BT.1uN.\c....(*[...vlU{..a#.)
.a...........icVsx1qBrNNdnNjRI.dll.P.teA...m.v&.q2..Lls..m....msbNl........U.......V,..
@.........;.P.8.h....../....?..:.........'.=..A....`.. .......0....c_y._.....)A.......
.....\..8...w.?...s._f.......N.F..z....}.\'....hbmg...........W.;[....~..P1....0.....?R
5..9-....(.2.
(5..0`!..H......!..:.Q.....F.C..D.IK
.e...4....... ..X...@...x..       .0....DK..6..a.3{..p
```

You can easily copy this and save it as an archive.

If you navigate to the folder which contains the ZIP archive, you can run the following command to extract it:

```
ubuntu@ip-172-31-31-107:/opt/NetworkMiner/AssembledFiles/192.99.198.158/TCP-80$ unzip xPF_HAXN7TK9bMAgBjZD.html
Archive:  xPF_HAXN7TK9bMAgBjZD.html
  inflating: AppManifest.xaml
```

This extracts a DLL file called icVsx1qBrNNdnNjRI.dll:

```
ubuntu@ip-172-31-31-107:/opt/NetworkMiner/AssembledFiles/192.99.198.158/TCP-80$ ls
2fNECYxvaRhNgivqycm7.swf   3xdz3bcxc8.html                   AppManifest.xaml             icVsx1qBrNNdnNjRI.dll
2nAY-xQvz4JQqjC66P7S.html  680VBFhpBNBJOYXebSxg.octet-stream  i_JnzurEICi4FQgJPm53.html  xPF_HAXN7TK9bMAgBjZD.html
```

Answer: icVsx1qBrNNdnNjRI.dll

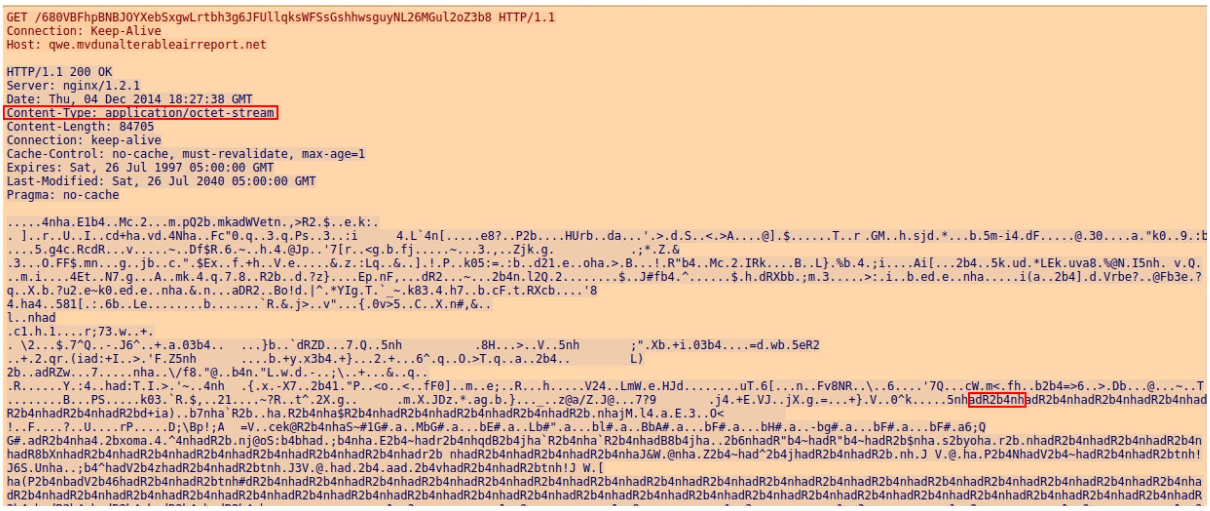
Extract the malware payload, deobfuscate it, and remove the shellcode at the beginning. This should give you the actual payload (a DLL file) used for the infection. What's the MD5 hash of the payload?

TLDR: Refer to the official walkthrough.

Using the following display filter:

- `http.host=="qwe.mvdunalterableairreport.net"`

If you follow the HTTP stream for GET /680VBFhpBNBJOYXebSxgwLrtbh3g6JFUllqksWFSsGshhwsguyNL26MGul2oZ3b8, we can see something suspicious:



The MIME type being application/octet-stream indicates that it is a binary file, meaning it could be an executable, an encrypted payload, or an archive. We can also see the string “dR2b4nh” being repeated multiple times within the payload, which could indicate XOR. To extract this file, we can use NetworkMiner:

Filter keyword: octet ☐ Case sensitive ExactPhrase

Frame nr.	Filename	Extension	Size	Source host	S. port	Destination host	D. port	Protocol	Timestamp
2839	680VBFhpBNBJOYXebSxg.octet-stream	octet-stream	84 705 B	192.99.198.158 [qwe.mvdunalterableairreport.net]	TCP 80	192.168.137.62	TCP 50473	HttpGetNormal	2014-12-04 18:27:34 UTC

To decode the payload, we can use a Python script that performs an XOR operation on the extracted file using “dR2b4nh” as the decryption key. Note! I took this script from the official walkthrough:

```python
def xor_file(input_file, output_file, key):
    key_bytes = key.encode()
    key_length = len(key_bytes)

    with open(input_file, 'rb') as infile, open(output_file, 'wb') as outfile:
        data = infile.read()
        xored_data = bytearray(len(data))

        for i in range(len(data)):
            xored_data[i] = data[i] ^ key_bytes[i % key_length]

        outfile.write(xored_data)

if __name__ == "__main__":
    input_filename  = "680VBFhpBNBJOYXebSxg.octet-stream"
    output_filename = "output.bin"
    key = "adR2b4nh"

    xor_file(input_filename, output_filename, key)
    print(f"File '{input_filename}' processed and saved as '{output_filename}'.")
```

```
ubuntu@ip-172-31-31-107:/opt/NetworkMiner/AssembledFiles/192.99.198.158/TCP-80$ python3 extractor.py
ubuntu@ip-172-31-31-107:/opt/NetworkMiner/AssembledFiles/192.99.198.158/TCP-80$ ls
2fNECYxvaRhNgivqycm7.swf   3xdz3bcxc8.html                    AppManifest.xaml  i_JnzurEICi4FQgJPm53.html  output.bin
2nAY-xQvz4JQqjC66P7S.html  680VBFhpBNBJOYXebSxg.octet-stream  extractor.py      icVsx1qBrNNdnNjRI.dll      xPF_HAXN7TK9bMAgBjZD.html
```

If you open the output file in GHex, we can see the MZ file header with a bunch of junk before it:

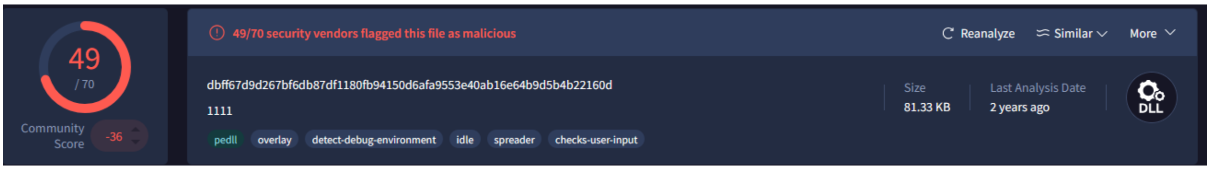
```
..................,.@...t............d.@..MZ..
.....f9.u9.H<..@r1;.w-.<.PE..u$V.. ..f.t..^t
..D.`Ev8..D.d.T.hP.~...3..U..QVW.........,.@
..}..tG.E.Pj.j..u..V ....t3.E...t,.<.. ;.s#.
u..A.P.B...P...E.P.V$....t..Q...u.3.@_^..]..
.3...QSUVW.:....t$....\$...,.@..T$.......t$.
..u.9F.t]..P.R....L$...tR.~..n.....;..D.;.t?
....t+.t$.y...........PQ...L$.....E........u
..t$..T$......3.@..3._^][Y......SUVW......l$
 ..j@h......,.@..}<...\$ .|$..wPj..S.....u.3
.......wT..G..D$$..G.UV.D$..+....D$,3....f;.
s0.\$...,.......s.....P.C...P.........[(Ou..|
$..\$.........t........t.V..P.......t...+G4P
V.0.....u.h....j.V.S..a....._^][......VW....
.....,.@......3..q.@.QQ.t$.PQQ.W...j.V.W.V.W
._^...VW......t$..... ....,.@..N<.D1(..f.T1.
u.j.j.j.V..P.W(..j.j.V....u.h....PV.W._3.^..
..t$........t.P.b.......D$.V.t$...t..T$.W..+
......BNu._^......X-..@..U.....SVW.E..E.kern
P...E.el32.E..dll.E...b.....h....V.....h&...
V.......h.jzi..V.{..p...h_p5:V.C..b...h...oV
.C..T...h.tC.V.C..F...h.+..V.C..8...h..>rV.C
..*....C..E.P.E.ntdl.E.l.dlf.E.l........h({?
PV.....h.0..V.C .....h.nMRV.C$......C(.E.P.E
.wini.E.net..E.dll.....h.=Y.V.....hf.}.V.C,.
....hb)!..C0V....._.C4..^[..]..T$.3.........
3.B....u....V.t$.3.W.|$....+.....0..9..t...t
.. :.u........u.3.@..3._^...d.0...SVW.@..p..
....t$..v0.......u....>;.u.3._^[....F...SUV.
t$.W3..F<.\0x...k ..9{.v..D....P.I...;D$.t.G
;{.r.3._^][....C$..xf..0f......C......0.....
............................................
.............PE..MZ.......................@..
............................................
.!..L.!This program cannot be run in DOS mod
e....$.......7,.SsM..sM..sM...E".qM..sM~..L.
..B".XM...Bp.uM...B#.rM...B .|M...B..SM...B!
```

This is all the shellcode we need to remove to extract the clean executable. To do so, you can execute the following command:

- `dd if=output.bin of=mz.bin bs=1 skip=1425`

```
ubuntu@ip-172-31-31-107:/opt/NetworkMiner/AssembledFiles/192.99.198.158/TCP-80$ dd if=output.bin of=mz.bin bs=1 skip=1425
83280+0 records in
83280+0 records out
83280 bytes (83 kB, 81 KiB) copied, 0.420116 s, 198 kB/s
ubuntu@ip-172-31-31-107:/opt/NetworkMiner/AssembledFiles/192.99.198.158/TCP-80$ file mz.bin
mz.bin: PE32 executable (DLL) (GUI) Intel 80386, for MS Windows, 5 sections
ubuntu@ip-172-31-31-107:/opt/NetworkMiner/AssembledFiles/192.99.198.158/TCP-80$ md5sum mz.bin
3dfa337e5b3bdb9c2775503bd7539b1c  mz.bin
```

If you submit this hash to VirusTotal, it receives a high detection rate:



Answer: 3dfa337e5b3bdb9c2775503bd7539b1c

What were the two protection methods enabled during the compilation of the PE file? (comma-separated)

Binary protections are mechanisms implemented during the compilation of an executable to make it more secure against exploitation. To find the two protection methods enabled during compilation of the PE file, we can use a tool called pesec:

- `pesec mz.bin`

```
ubuntu@ip-172-31-31-107:/opt/NetworkMiner/AssembledFiles/192.99.198.158/TCP-80$ pesec mz.bin
ASLR:                             no
DEP/NX:                           no
SEH:                              yes
Stack cookies (EXPERIMENTAL):     yes
```

Stack cookies are also known as canary, therefore the two protections enabled are SEH, and Canary.

Answer: SEH, Canary

A Flash file was used in conjunction with the redirect URL. What URL was used to retrieve this flash file?

Using the following query in Zui:

- `_path=="files" mime_type=="application/x-shockwave-flash"`

We can see that two flash files were downloaded:

_path	ts	fuid	uid	id
files	2014-12-04T18:27:49.988502Z	FJDkNn4rwZhSBnyen9	CYaVBo2ErYlNahWpQc	{ orig_h: 192.168.137.62, orig_p: 50468, resp_h: 192.99.198.158, resp_p: 80 }
files	2014-12-04T18:27:27.786607Z	FVnXWT3w5DR5t2BhU6	CUUw2SoQ7vTaEYpQe	{ orig_h: 192.168.137.62, orig_p: 50450, resp_h: 93.114.64.118, resp_p: 80 }

We can now use the following query to find the host and URI of the host that sent the flash file to the victim:

- `_path=="http" id.resp_h==93.114.64.118 | cut ts, host, uri, referrer`

ts	host	uri	referrer
2014-12-04T18:27:27.573286Z	adstairs.ro	/544b29bcd035b2dfd055f5deda91d648.swf	http://www.earsurgery.org/

The reason I am focusing on adstairs.ro and not the other IP is because the other IP is attributed to delivering the exploit kit.

Alternatively, you can navigate to File > Export Objects > HTTP in Wireshark and search for swf to find where the flash file was downloaded from:

Text Filter: swf

Packet	Hostname	Content Type	Size	Filename
1875	adstairs.ro	application/x-shockwave-flash	956 bytes	544b29bcd035b2dfd055f5deda91d648.swf

Answer: http://adstairs.ro/544b29bcd035b2dfd055f5deda91d648.swf

What is the CVE of the exploited vulnerability?

To determine the CVE of the exploited vulnerability, I researched for CVE's exploited by Angler EK and came across the following post by keysight that talks about how Angler EK was observed exploiting CVE-2013-2551.

Answer: CVE-2013-2551