Challenge: Malware Traffic Analysis 2 Lab

Platform: CyberDefenders

Category: Network Forensics

Difficulty: Medium

Tools Used: Wireshark, Zui, NetworkMiner, VirusTotal

Summary: This lab involved investigating a Windows VM that was infected with the Sweet Orange exploit kit. I found it relatively difficult, mainly since it's done over a VM, meaning you are limited on tools along with not being able to analyse extracted files in sandbox environments. Nonetheless, it was enjoyable and is a great lab for practicing your network forensic skills.

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.

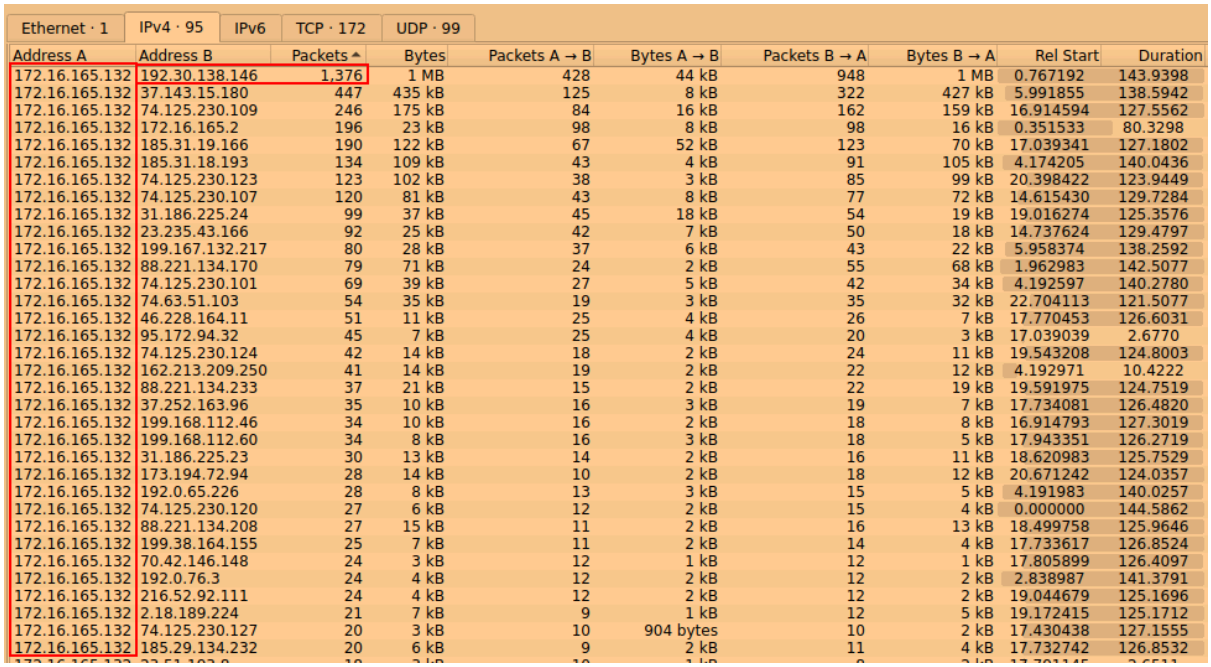
What is the IP address of the Windows VM that gets infected?

TLDR: Navigate to Statistics > Conversations > IPv4 and focus on what IP appears in all conversations.

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	4682	100.0	2833334
Ethernet	100.0	4682	2.5	69713
Internet Protocol Version 4	100.0	4682	3.3	93640
User Datagram Protocol	4.2	197	0.1	1576
Network Time Protocol	0.0	1	0.0	48
Domain Name System	4.2	196	0.5	14964
Transmission Control Protocol	95.8	4485	93.6	2653393
Transport Layer Security	4.6	215	9.4	266680
Hypertext Transfer Protocol	11.2	526	76.1	2157039
Portable Network Graphics	0.7	32	22.0	622595
Malformed Packet	0.0	1	0.0	0
Media Type	0.3	12	5.7	162402
Line-based text data	2.0	92	38.2	1082234
JPEG File Interchange Format	0.1	7	15.7	444075
Data	0.0	1	13.0	369056
Compuserve GIF	1.5	69	0.2	5380

Here we can see that HTTP makes up a large portion of the traffic. Let's now navigate to Statistics > Conversations > IPv4 to get an understanding of the hosts within this PCAP:



Ethernet · 1 | IPv4 · 95 | IPv6 | TCP · 172 | UDP · 99

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration
172.16.165.132	192.30.138.146	1,376	1 MB	428	44 kB	948	1 MB	0.767192	143.9398
172.16.165.132	37.143.15.180	447	435 kB	125	8 kB	322	427 kB	5.991855	138.5942
172.16.165.132	74.125.230.109	246	175 kB	84	16 kB	162	159 kB	16.914594	127.5562
172.16.165.132	172.16.165.2	196	23 kB	98	8 kB	98	16 kB	0.351533	80.3298
172.16.165.132	185.31.19.166	190	122 kB	67	52 kB	123	70 kB	17.039341	127.1802
172.16.165.132	185.31.18.193	134	109 kB	43	4 kB	91	105 kB	4.174205	140.0436
172.16.165.132	74.125.230.123	123	102 kB	38	3 kB	85	99 kB	20.398422	123.9449
172.16.165.132	74.125.230.107	120	81 kB	43	8 kB	77	72 kB	14.615430	129.7284
172.16.165.132	31.186.225.24	99	37 kB	45	18 kB	54	19 kB	19.016274	125.3576
172.16.165.132	23.235.43.166	92	25 kB	42	7 kB	50	18 kB	14.737624	129.4797
172.16.165.132	199.167.132.217	80	28 kB	37	6 kB	43	22 kB	5.958374	138.2592
172.16.165.132	88.221.134.170	79	71 kB	24	2 kB	55	68 kB	1.962983	142.5077
172.16.165.132	74.125.230.101	69	39 kB	27	5 kB	42	34 kB	4.192597	140.2780
172.16.165.132	74.63.51.103	54	35 kB	19	3 kB	35	32 kB	22.704113	121.5077
172.16.165.132	46.228.164.11	51	11 kB	25	4 kB	26	7 kB	17.770453	126.6031
172.16.165.132	95.172.94.32	45	7 kB	25	4 kB	20	3 kB	17.039039	2.6770
172.16.165.132	74.125.230.124	42	14 kB	18	2 kB	24	11 kB	19.543208	124.8003
172.16.165.132	162.213.209.250	41	14 kB	19	2 kB	22	12 kB	4.192971	10.4222
172.16.165.132	88.221.134.233	37	21 kB	15	2 kB	22	19 kB	19.591975	124.7519
172.16.165.132	37.252.163.96	35	10 kB	16	3 kB	19	7 kB	17.734081	126.4820
172.16.165.132	199.168.112.46	34	10 kB	16	2 kB	18	8 kB	16.914793	127.3019
172.16.165.132	199.168.112.60	34	8 kB	16	3 kB	18	5 kB	17.943351	126.2719
172.16.165.132	31.186.225.23	30	13 kB	14	2 kB	16	11 kB	18.620983	125.7529
172.16.165.132	173.194.72.94	28	14 kB	10	2 kB	18	12 kB	20.671242	124.0357
172.16.165.132	192.0.65.226	28	8 kB	13	3 kB	15	5 kB	4.191983	140.0257
172.16.165.132	74.125.230.120	27	6 kB	12	2 kB	15	4 kB	0.000000	144.5862
172.16.165.132	88.221.134.208	27	15 kB	11	2 kB	16	13 kB	18.499758	125.9646
172.16.165.132	199.38.164.155	25	7 kB	11	2 kB	14	4 kB	17.733617	126.8524
172.16.165.132	70.42.146.148	24	3 kB	12	1 kB	12	1 kB	17.805899	126.4097
172.16.165.132	192.0.76.3	24	4 kB	12	2 kB	12	2 kB	2.838987	141.3791
172.16.165.132	216.52.92.111	24	4 kB	12	2 kB	12	2 kB	19.044679	125.1696
172.16.165.132	2.18.189.224	21	7 kB	9	1 kB	12	5 kB	19.172415	125.1712
172.16.165.132	74.125.230.127	20	3 kB	10	904 bytes	10	2 kB	17.430438	127.1555
172.16.165.132	185.29.134.232	20	6 kB	9	2 kB	11	4 kB	17.732742	126.8532

Immediately we can see that 172.16.165.132 is present in all conversations and is within the private IP address space, which suggests that it's the Windows VM. We can also see many packets being sent from several hosts that warrant further investigation.

Answer: 172.16.165.132

What are the IP address and port number that delivered the exploit kit and malware?

TLDR: Query for notice alerts in Zui and focus on the one concerning "Malware Detected". Alternatively, look for GET requests made by the VM to suspicious IPs (IPs that geolocate to countries like Russia).

If you load the PCAP into Zui, we can use the following query to look for anything interesting:

- `_path=="notice"`

Here we can find a record for an executable downloaded from an IP that geolocates to Moscow Russia:

```
id: {
  orig_h: 172.16.165.132,
  orig_p: 49398 (port=(uint16)),
  resp_h: 37.143.15.180,
  resp_p: 51439 (port=(uint16))
},
fuid: "FScqah49B1diuyxmUf",
file_mime_type: "application/x-dosexec",
file_desc: "http://h.trinketking.com:51439
proto: "tcp" (zenum),
note: "TeamCymruMalwareHashRegistry::Match
msg: "Malware Hash Registry Detection rate
sub: "https://www.virustotal.com/gui/searc
src: 172.16.165.132,
dst: 37.143.15.180,
p: 51439 (port=(uint16)),
n: null,
peer_descr: null,
actions: |[
  0: "Notice::ACTION_LOG" (zenum),
  1: "Notice::ACTION_ADD_GEODATA" (zenum)
]|,
email_dest: |[
]|,
suppress_for: 1h,
remote_location: {
  country_code: "RU",
  region: "MOW",
```

Using the following display filter in Wireshark:

- `ip.addr==172.16.165.132 && ip.addr==37.143.15.180 && http`

We can see the HTTP requests and responses between the infected VM and a suspicious IP:

Source	Destination	Protocol	Length	Info
172.16.165.132	37.143.15.180	HTTP	383	GET /consumer/empty/birds.php?winter=3 HTTP/1.1
37.143.15.180	172.16.165.132	HTTP	686	HTTP/1.1 200 OK (text/html)
172.16.165.132	37.143.15.180	HTTP	289	GET /cars.php?honda=1185&proxy=2442&timeline=4&jobs=823&image=171&join=757&list=679 HTTP/1.1
37.143.15.180	172.16.165.132	HTTP	747	HTTP/1.1 200 OK
172.16.165.132	37.143.15.180	HTTP	413	GET /consumer/empty/ENFWAKJWN2NOB3 HTTP/1.1
37.143.15.180	172.16.165.132	HTTP	215	HTTP/1.1 404 Not Found (text/html)

If you view the HTTP/TCP stream of the GET request to /cars.php, we can see that this is an executable file, as indicated by the MZ file header:

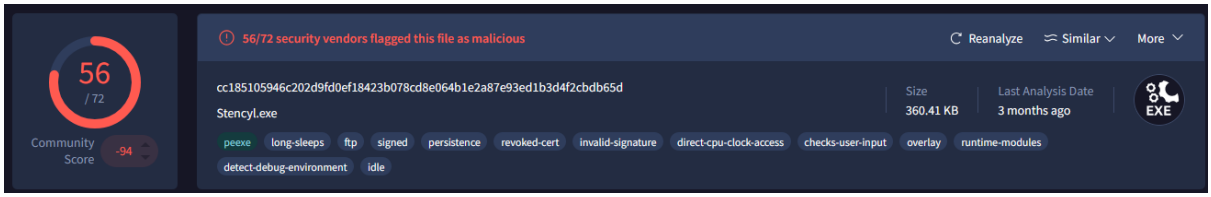
```
HTTP/1.1 200 OK
Server: nginx/1.6.2
Date: Sun, 23 Nov 2014 00:58:36 GMT
Content-Type: application/octet-stream
Content-Length: 369056
Connection: keep-alive

MZ.....................@.....................................        .!..L.!This program cannot be run in DOS mode.
```

What also stands out is the destination port:

```
Transmission Control Protocol, Src Port: 49398, Dst Port: 51439
   Source Port: 49398
   Destination Port: 51439
```

Typically, you would see HTTP over port 80. Given this, and the fact that the hash submitted to VirusTotal reveals 56/72 detections, it's safe to say that this binary is malicious:



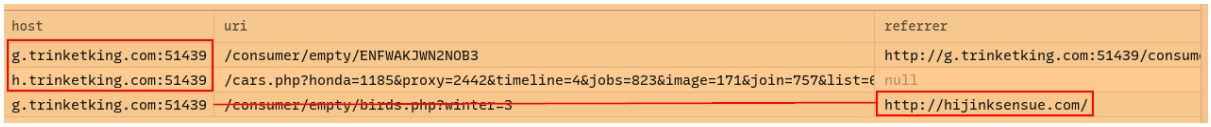
Answer: 37.143.15.180:51439

What are the two FQDN's that delivered the exploit kit? comma-separated in alphabetical order.

TDLR: Filter for the IP discovered earlier that we attribute to delivering the exploit kit. You can do so within Zui, Wireshark, or NetworkMiner. Focus on the host field, that shows the FQDN that delivered the exploit kit.

My preferred method of determining this is through Zui. We know that 37.143.15.180 is attributed with being the source of the exploit kit, so we can filter for all responses from this host and display some key fields:

- `_path=="http" | id.resp_h==37.143.15.180 | cut host, uri, referrer`



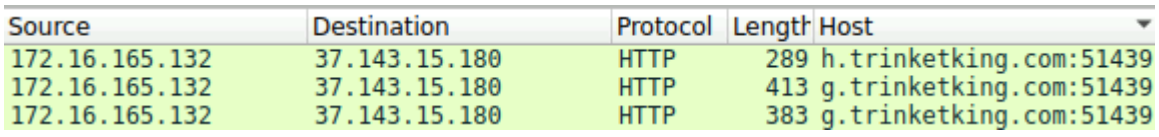
host	uri	referrer
g.trinketking.com:51439	/consumer/empty/ENFWAKJWN2NOB3	http://g.trinketking.com:51439/consum
h.trinketking.com:51439	/cars.php?honda=1185&proxy=2442&timeline=4&jobs=823&image=171&join=757&list=6	null
g.trinketking.com:51439	/consumer/empty/birds.php?winter=3	http://hijinksensue.com/

Here we can see two unique FQDN’s associated with the malicious IP. We can also see the referrer URL; this likely suggests that http://hijinksensue.com/ is compromised.

Alternatively, in Wireshark, if you apply the http.host field as a column and enter the following display filter:

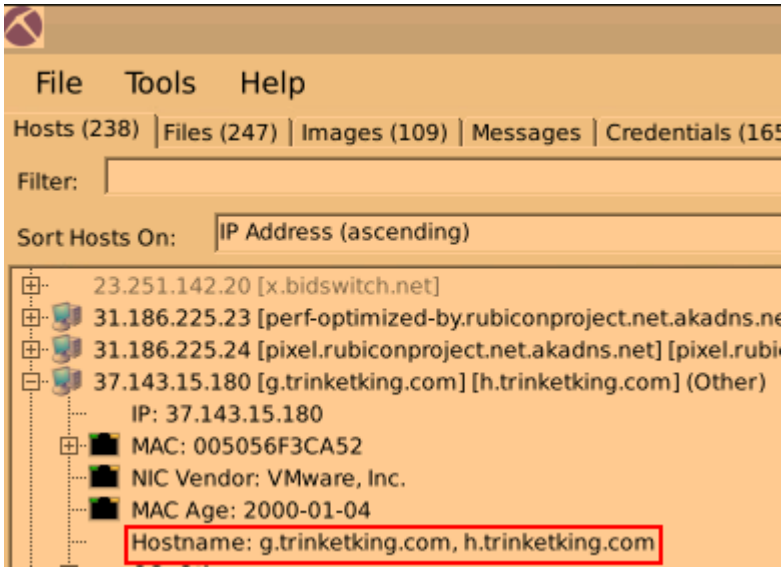
- `ip.addr==37.143.15.180 && http`

You can find the two FQDN’s:



Source	Destination	Protocol	Length	Host
172.16.165.132	37.143.15.180	HTTP	289	h.trinketking.com:51439
172.16.165.132	37.143.15.180	HTTP	413	g.trinketking.com:51439
172.16.165.132	37.143.15.180	HTTP	383	g.trinketking.com:51439

Another way of identifying this is through NetworkMiner, if you navigate to the suspicious IP in the Hosts tab, you can find the two FQDN’s:



Answer: g.trinketking.com,h.trinketking.com

What is the FQDN of the compromised website?

In the previous question, we determined that http://hijinksensue.com/ is the referrer for the request to g.trinketking.com. This suggests that it is compromised:

- `http.host=="hijinksensue.com"`

Source	Destination	Protocol	Length	Info
172.16.165.132	192.30.138.146	HTTP	525	GET / HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	365	GET /wp-content/themes/comicpress-hijinks-2011/style.css HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	384	GET /wp-content/plugins/eshop-order-emailer/css/plugin_styles.css?ver=4.0.1 HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	397	GET /wp-content/themes/comicpress-hijinks-2011/images/nav/hijinks/navstyle.css?ver=4.0.1 HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	380	GET /wp-content/plugins/wp-lightbox-2/styles/lightbox.min.css?ver=1.3.4 HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	366	GET /wp-content/plugins/jetpack/css/jetpack.css?ver=3.2.1 HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	373	GET /wp-content/plugins/comic-easel/css/comiceasel.css?ver=4.0.1 HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	386	GET /wp-content/plugins/comic-easel/images/nav/comical/navstyle.css?ver=4.0.1 HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	354	GET /wp-content/uploads/eshop_files/eshop.css HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	382	GET /wp-content/plugins/mf-gig-calendar/css/mf_gig_calendar.css?ver=4.0.1 HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	381	GET /wp-includes/js/jquery/jquery.js?ver=1.11.1 HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	392	GET /wp-includes/js/jquery/jquery-migrate.min.js?ver=1.2.1 HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	412	GET /wp-content/plugins/google-analyticator/external-tracking.min.js?ver=6.4.8 HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	409	GET /assets/misc/Patreon-Patron-Homepage-Banner-button.png HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	433	GET /wp-content/uploads/2014/11/2014-11-12-the-objectification-of-my-affection.jpg HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	428	GET /wp-content/themes/comicpress-hijinks-2011/images/layout/spacer-100x3.png HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	382	GET /assets/misc/hive_small.png HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	426	GET /wp-content/uploads/2014/09/dalek-earrings-etsy-science-and-fiction.png HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	352	GET /wp-content/themes/comicpress/style.css HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	429	GET /wp-content/uploads/2014/11/hijinks-ensue-explosm-store-banner-closing.png HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	432	GET /wp-content/uploads/2014/10/2014-10-09-hijinks-ensue-shut-up-forever-nycc.jpg HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	432	GET /wp-content/uploads/2014/10/potter-and-daughter-podcast-logo-hijink-ensue.png HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	407	GET /wp-content/uploads/2014/03/becomepatron-300x132.png HTTP/1.1
172.16.165.132	192.30.138.146	HTTP	374	GET /assets/verts/hiveworks/ad3.html HTTP/1.1

Answer: hijinksensue.com

What is the name exploit kit (EK) that delivered the malware? (two words)

Unfortunately, I was unable to find the name of the exploit kit. I started by submitting the hash of the exploit kit to Virus Total, but this mentions nothing about an exploit kit. The hint recommends using PacketTotal, however, the VM has no internet connection, so this can't work. I recommend exploring other writeups to see how they used PacketTotal to get the answer.

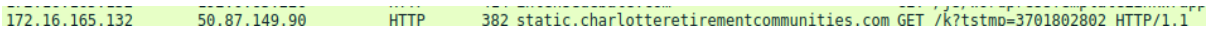
Answer: Sweet Orange

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

Using the following display filter:

- `frame contains "hijinksensue.com"`

We can see a request from our compromised VM to static.charlotteretirementcommunities.com:



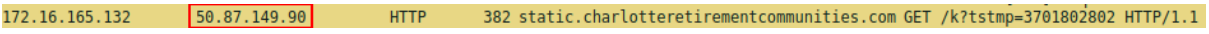
If you view the HTTP stream of this request, we can see suspicious JavaScript:

This confirms that the compromised website is hijinksensue.com due to it being the referrer, it likely contained a hidden or injected <script> tag that points to static.charlotteretirementcommunities.com. Furthermore, if you search for Sweet Orange and look at detections, the GET request format of /k?stmp has been observed in multiple incidents.

Answer: static.charlotteretirementcommunities.com/k?tstmp=3701802802

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

We discovered this earlier in the request made to static.charlotteretirementcommunities.com:



Answer: 50.87.149.90

Extract the malware payload (PE file) from the PCAP. What is the MD5 hash?

TLDR: Navigate to File > Export Objects > HTTP and search for h.trinketking.com.

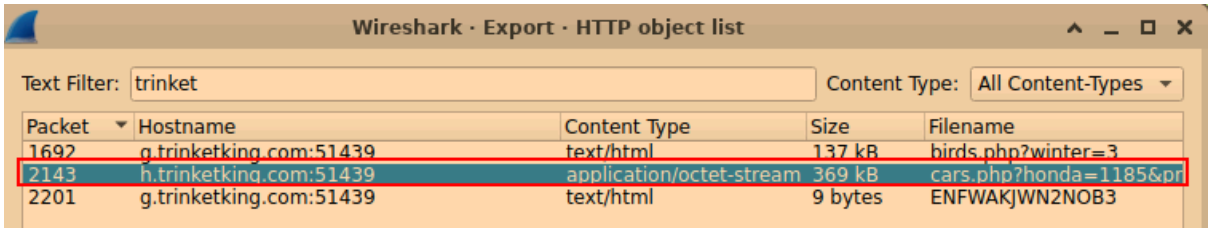
Recall earlier how we discovered a suspicious GET request made to h.trinketking.com over port 51439 that responded with a PE file:

```
GET /cars.php?honda=1185&proxy=2442&timeline=4&jobs=823&image=171&join=757&list=679 HTTP/1.1
Connection: Keep-Alive
Accept: */*
User-Agent: Mozilla/4.0 (compatible; Win32; WinHttp.WinHttpRequest.5)
Host: h.trinketking.com:51439

HTTP/1.1 200 OK
Server: nginx/1.6.2
Date: Sun, 23 Nov 2014 00:58:36 GMT
Content-Type: application/octet-stream
Content-Length: 369056
Connection: keep-alive

MZ.....................@.....................................        .!..L.!This program cannot be run in DOS mode.
```

In Wireshark, if you navigate to File > Export Objects > HTTP and filter for trinket, we can export this executable:



You can then run the md5sum command against this file:

```
ubuntu@ip-172-31-23-180:~/Desktop$ md5sum 'cars.php%3fhonda=1185&proxy=2442&timeline=4&jobs=823&image=171&join=757&list=679'
1408275c2e2c8fe5e83227ba371ac6b3  cars.php%3fhonda=1185&proxy=2442&timeline=4&jobs=823&image=171&join=757&list=679
```

This approach is unnecessary, as we found the hash within the notice log of Zui previously, however, I think it's important to understand how to do it manually. For some unknown reason, NetworkMiner was unable to extract this file.

Answer: 1408275c2e2c8fe5e83227ba371ac6b3

What is the CVE of the exploited vulnerability?

If you Google CVE Sweet Orange, you will come across advisories regarding CVE-2014-6332:

CVE-2014-6332 Detail

DEFERRED

This CVE record is not being prioritized for NVD enrichment efforts due to resource or other concerns.

Description

OleAut32.dll in OLE in Microsoft Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8, Windows 8.1, Windows Server 2012 Gold and R2, and Windows RT Gold and 8.1 allows remote attackers to execute arbitrary code via a crafted web site, as demonstrated by an array-redimensioning attempt that triggers improper handling of a size value in the SafeArrayDimen function, aka "Windows OLE Automation Array Remote Code Execution Vulnerability."

There are also posts that indicate Sweet Orange exploiting CVE-2014-6332:

Many other common programs like Adobe Reader and Microsoft Internet Explorer are affected. However, some exploit kits utilize new and unique attack vectors such as Sweet Orange's Visual Basic Script exploit (CVE-2014-6332).

Answer: CVE-2014-6332

What was the referrer for the visited URI that returned the file "f.txt"?

Using the following display filter, we can search for any packets that contain f.txt:

- `frame contains "f.txt"`

Source	Destination	Protocol	Length
74.125.230.109	172.16.165.132	TCP	1514
74.125.230.109	172.16.165.132	TCP	1409
74.125.230.109	172.16.165.132	TCP	1514

Here we can see three responses from 74.125.230.109. If you follow the TCP stream of the first request, you can see the referrer within the HTTP header:

```
GET /pagead/js/adsbygoogle.js HTTP/1.1
Accept: application/javascript, */*;q=0.8
Referer: http://hijinksensue.com/assets/verts/hiveworks/ad1.html
Accept-Language: en-US
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)
Accept-Encoding: gzip, deflate
Host: pagead2.googlesyndication.com
Connection: Keep-Alive
```

Answer: http://hijinksensue.com/assets/verts/hiveworks/ad1.html