SANS DFIR challenge (Anns Aurora) writeup

By: Andrew Mayer

This was a fun, albeit shorter than I'd like challenge that my mentor gave me to get some practice with. It gives the analyst a singular .pcap file to work with, along with a series of 10 questions, and a few additional subquestions. I went through this fairly quickly and easily, aside from some MD5 Hashes I could not get right for the life of me. My analysis and answering went as follows:

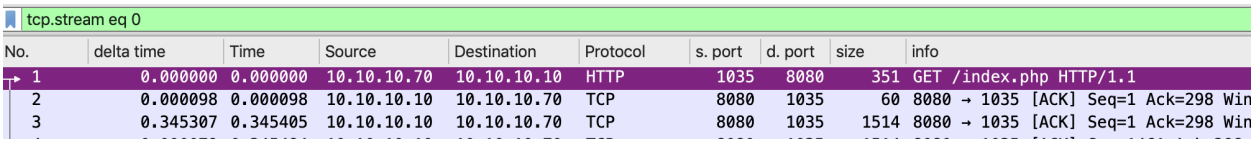
Ann Dercover is after SaucyCorp's Secret Sauce recipe. She's been trailing the lead developer, Vick Timmes, to figure out how she can remotely access SaucyCorp's servers. One night, while conducting reconnaissance, she sees him log into his laptop (10.10.10.70) and VPN into SaucyCorp's headquarters.

Leveraging her connections with international hacking organizations, Ann obtains a 0-day exploit for Internet Explorer and launches a client-side spear phishing attack against Vick Timmes. Ann carefully crafts an email to Vick containing tips on how to improve secret sauce recipes and sends it. Seeing an opportunity that could get him that Vice President of Product Development title (and corner office) that he's been coveting, Vick clicks on the link. Ann is ready to strike…

You are the forensic investigator. Your mission is to analyze the packet capture containing Ann's exploit, build a timeline, and submit your evidence including...

1. **What was the full URI of Vick Timmes' original web request? (Please include the port in your URI.)**

Theory: I began this by setting up my Wireshark profile to my liking and getting to work with some basic looking into the interesting packets. I have very useful coloring for TCP and HTTP traffic, so I had some things to look at right away. I could see a few established TCP sessions, a few windows executables being sent, and a few get requests right at the top of the capture. This first answer was very easy to find, as it was the first packet in the capture.



A: 10.10.10.10:8080/index.php

2. **In response, the malicious web server sent back obfuscated JavaScript. Near the beginning of this code, the attacker created an array with 1300 elements labeled "COMMENT", then filled their data element with a string. What was the value of this string?**

Theory: The wording of the question made me think a bit. I am familiar with JS, but not the most fluent. Knowing that this was a response to the original request, I found the first response, and followed the TCP

stream. By examining the JS code for a bit, I was able to make up what it is saying, and the value it is

```
<script>
var UWnHADOfYHiHDDXj = "COMMENT";
var qSNgVkOrdIjaiFpPTfDjbPHQppHSGtzpmOOyqEbLEFxNqAxicRyZKKWiRWmUaDHFOuzHPHqLrRFSzQuPusTnQyqpQwVpARdlR = new Array();
for (i = 0; i < 1300; i++)
{
  qSNgVkOrdIjaiFpPTfDjbPHQppHSGtzpmOOyqEbLEFxNqAxicRyZKKWiRWmUaDHFOuzHPHqLrRFSzQuPusTnQyqpQwVpARdlR[i] = document.createElement(UWnHADOfYHiHDDXj);
  qSNgVkOrdIjaiFpPTfDjbPHQppHSGtzpmOOyqEbLEFxNqAxicRyZKKWiRWmUaDHFOuzHPHqLrRFSzQuPusTnQyqpQwVpARdlR[i].data = "vEI";
}
var dccahTiTDPLKAIGwlKLcqCTAoePblbBvEYSuWFWodqQrztYYizoWxwqlMJzsyPhKfhPwpLGFWIflTpaOhwy = null;
var VTWgXjXYoSTHY = new Array();
var lwFZUGyxMIWTGMBfMqyKxGTRyRuDasLmlzjI = unescape;
function XyFfBwnvNTHaNNqCqPKVxmwOjqwptvyJsLDUgweSdLDgm()
{
```

giving to the COMMENT variable:
A:vEI

3. **Vick's computer made a second HTTP request for an object.**
    a. **What was the filename of the object that was requested?**

Theory: Again, my preset filters and knowing what to look for made it easy to find the second http request:

3	0.345307	0.345405	10.10.10.10	10.10.10.70	TCP	8080	1035	1514	8080 → 1035 [ACK] Seq=1 Ack=298 Win=8576 Len=1460 [TCP segment
4	0.000079	0.345484	10.10.10.10	10.10.10.70	TCP	8080	1035	1514	8080 → 1035 [ACK] Seq=1461 Ack=298 Win=8576 Len=1460 [TCP segme
5	0.000124	0.345608	10.10.10.10	10.10.10.70	TCP	8080	1035	1514	8080 → 1035 [ACK] Seq=2921 Ack=298 Win=8576 Len=1460 [TCP segme
6	0.000121	0.345729	10.10.10.10	10.10.10.70	TCP	8080	1035	1514	8080 → 1035 [ACK] Seq=4381 Ack=298 Win=8576 Len=1460 [TCP segme
7	0.000095	0.345824	10.10.10.70	10.10.10.10	TCP	1035	8080	60	1035 → 8080 [ACK] Seq=298 Ack=5841 Win=65535 Len=0
8	0.000105	0.345929	10.10.10.10	10.10.10.70	HTTP	8080	1035	86	HTTP/1.1 200 OK (text/html)
9	0.116183	0.462112	10.10.10.70	10.10.10.10	HTTP	1035	8080	415	GET /index.phpmfKSxSANkeTeNrah.gif HTTP/1.1
10	0.000117	0.462229	10.10.10.10	10.10.10.70	TCP	8080	1035	60	8080 → 1035 [ACK] Seq=5873 Ack=659 Win=9648 Len=0
11	0.104914	0.567143	10.10.10.10	10.10.10.70	HTTP	8080	1035	201	HTTP/1.1 200 OK (GIF89a)
12	0.170074	0.737217	10.10.10.70	10.10.10.10	TCP	1035	8080	60	1035 → 8080 [ACK] Seq=659 Ack=6020 Win=65356 Len=0

A: index.phpmfKSxSANkeTeNrah.gif

    b. **What is the MD5sum of the object that was returned?**

A: A simple export object, and MD5 command revealed the sum to be df3e567d6f16d040326c7a0ea29a4f41

4. **When was the TCP session on port 4444 opened? (Provide the number of seconds since the beginning of the packet capture, rounded to tenths of a second. ie, 49.5 seconds)**

Theory: a simple filter of tcp.srcport == 4444 revealed that the server was sending a synack, meaning it was being requested a connection. The tcp.dstport == 444 revealed the time that the host ACK'd the tcp handshake.

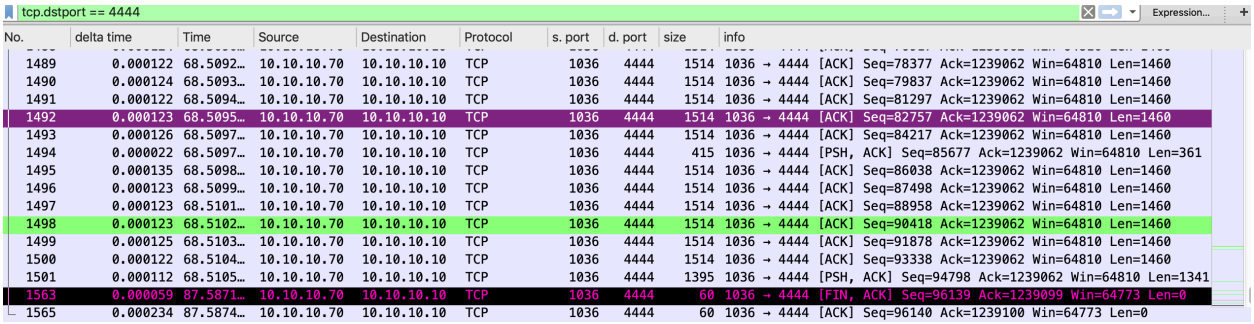
tcp.dstport == 4444

No.	delta time	Time	Source	Destination	Protocol	s. port	d. port	size	info
13	0.528634	1.265851	10.10.10.70	10.10.10.10	TCP	1036	4444	62	1036 → 4444 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
15	0.000296	1.266218	10.10.10.70	10.10.10.10	TCP	1036	4444	60	1036 → 4444 [ACK] Seq=1 Ack=1 Win=65535 Len=0
19	0.000322	1.530178	10.10.10.70	10.10.10.10	TCP	1036	4444	60	1036 → 4444 [ACK] Seq=1 Ack=2925 Win=65535 Len=0
24	0.000142	1.530931	10.10.10.70	10.10.10.10	TCP	1036	4444	60	1036 → 4444 [ACK] Seq=1 Ack=8765 Win=65535 Len=0

A: 1.3s

5. **When was the TCP session on port 4444 closed? (Provide the number of seconds since the beginning of the packet capture, rounded to tenths of a second. ie, 49.5 seconds)**

A: using the same filter and my coloring rules, I was able to easily find the FINACK and ACK messages between the client and server. 87.6s



6. **In packet 17, the malicious server sent a file to the client.**
    a. **What type of file was it? Choose one:**
        - ************Windows executable**
        - **GIF image**
        - **PHP script**
        - **Zip file**
        - **Encrypted data**
    b. **What was the MD5sum of the file?**

Theory: the first part of this was fairly easy for me. Once the content was examined, I was able to see the MZ header and the famous “DOS mode” phrase that signifies windows executable headers. The md5 sum was one I had issue with, and took a while to get, with some help from learning new tools and using walkthroughs, because I wanted to learn it and the reasoning behind using specific tools to find the correct answer.
I first used tcpflow to separate out the pcap to different connections in terminal form, and got all 6 connections:

```
$ ls -la
total 4492
drwxr-xr-x  3 sansforensics sansforensics    4096 Oct  3 12:24 .
drwxr-xr-x 19 sansforensics sansforensics    4096 Oct  3 12:20 ..
-rw-r--r--  1 root          root          1239098 Apr 28  2010 010.010.010.010.04444-010.010.010.070.01036
-rw-r--r--  1 root          root           833091 Apr 28  2010 010.010.010.010.04445-010.010.010.070.01044
-rw-r--r--  1 root          root             6019 Apr 28  2010 010.010.010.010.08080-010.010.010.070.01035
-rw-r--r--  1 root          root              658 Apr 28  2010 010.010.010.070.01035-010.010.010.010.08080
-rw-r--r--  1 root          root            96138 Apr 28  2010 010.010.010.070.01036-010.010.010.010.04444
-rw-r--r--  1 root          root          1144419 Apr 28  2010 010.010.010.070.01044-010.010.010.010.04445
-rwxr-x---  1 root          root          2371708 Oct  3 12:21 evidence06.pcap
```

Note that I tried using the tool **foremost** to carve out the file first, but it proved to take too long and not quite give me the answer I needed. After getting the specific connection I needed separated out with TCPflow, I was able to run the command foremost on that specific file, and the output came to a .dll file. Md5sum was then ran, and answer was found: b062cb8344cd3e296d8868fbef289c7c

```
sansforensics@siftworkstation: ~/Documents
$ foremost -t exe 010.010.010.010.04444-010.010.010.070.01036
Processing: 010.010.010.010.04444-010.010.010.070.01036
|*|
sansforensics@siftworkstation: ~/Documents
$ ls
010.010.010.010.04444-010.010.010.070.01036  010.010.010.070.01035-010.010.010.010.08080  evidence06.pcap
010.010.010.010.04445-010.010.010.070.01044  010.010.010.070.01036-010.010.010.010.04444  output
010.010.010.010.08080-010.010.010.070.01035  010.010.010.070.01044-010.010.010.010.04445  report.xml
sansforensics@siftworkstation: ~/Documents
$ cat output/
audit.txt  dll/
sansforensics@siftworkstation: ~/Documents
$ cat output/
audit.txt  dll/
sansforensics@siftworkstation: ~/Documents
$ cat output/
audit.txt  dll/
sansforensics@siftworkstation: ~/Documents
$ md5sum output/dll/00000000.dll
b062cb8344cd3e296d8868fbef289c7c  output/dll/00000000.dll
```

A:

7. **Vick's computer repeatedly tried to connect back to the malicious server on port 4445, even after the original connection on port 4444 was closed. With respect to these repeated failed connection attempts:**
   a. **How often does the TCP initial sequence number (ISN) change? (Choose one.)**
      - **Every packet**
      - *****Every third packet**
      - **Every 10-15 seconds**
      - **Every 30-35 seconds**
      - **Every 60 seconds**

   b. **How often does the IP ID change? (Choose one.)**
      - *****Every packet**
      - **Every third packet**
      - **Every 10-15 seconds**
      - **Every 30-35 seconds**
      - **Every 60 seconds**

   c. **How often does the source port change? (Choose one.)**
      - **Every packet**
      - **Every third packet**
      - *****Every 10-15 seconds**
      - **Every 30-35 seconds**
      - **Every 60 second**

tcp.dstport == 4445

	delta time	Time	Source	Destination	Protocol	s. port	d. port
1323	0.546826	61.3305…	10.10.10.70	10.10.10.10	TCP	1039	4445
1325	0.000614	61.3311…	10.10.10.70	10.10.10.10	TCP	1039	4445
1327	0.546146	61.8773…	10.10.10.70	10.10.10.10	TCP	1039	4445
1329	0.546832	62.4242…	10.10.10.70	10.10.10.10	TCP	1039	4445
1331	0.000592	62.4249…	10.10.10.70	10.10.10.10	TCP	1039	4445
1333	0.546188	62.9711…	10.10.10.70	10.10.10.10	TCP	1039	4445
1335	0.546809	63.5180…	10.10.10.70	10.10.10.10	TCP	1039	4445
1337	0.000603	63.5186…	10.10.10.70	10.10.10.10	TCP	1039	4445
1339	0.546176	64.0649…	10.10.10.70	10.10.10.10	TCP	1039	4445
1341	0.437434	64.5023…	10.10.10.70	10.10.10.10	TCP	1039	4445
1503	2.745565	71.2579…	10.10.10.70	10.10.10.10	TCP	1040	4445
1505	0.463069	71.7211…	10.10.10.70	10.10.10.10	TCP	1040	4445
1507	0.546827	72.2680…	10.10.10.70	10.10.10.10	TCP	1040	4445
1509	0.001146	72.2692…	10.10.10.70	10.10.10.10	TCP	1040	4445
1511	0.436284	72.7055…	10.10.10.70	10.10.10.10	TCP	1040	4445

```
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 48
    Identification: 0x020e (526)
  ▶ Flags: 0x4000, Don't fragment
```

Theory: using the tcp.dstport == 4445 filter, I was able to see all packets attempting to connect on port 4445.from here, I just had to check a few parameters. For the ISN, I watched the steam index, which would only change with each new tcp stream, which would also change the ISN, as it would be a new tcp sequence. For the IP ID, the IP identification field in the packets was changing each time. Lastly, for the source port change, I simply had to count the seconds between each srcport change, and it came out to 10-13 seconds for each.

8. **Eventually, the malicious server responded and opened a new connection. When was the TCP connection on port 4445 first successfully completed? (Provide the number of seconds since the beginning of the packet capture, rounded to tenths of a second. ie, 49.5 seconds)**

A: With this, I simply looked for the first non-retransmission packet going to the server, an ACK packet. 123.7s

9. **Subsequently, the malicious server sent an executable file to the client on port 4445. What was the MD5 sum of this executable file?**

A: this was almost the same commant/theory as the earlier question I had trouble with, 6b. Knowing this and the question, it was easy to put nearly the same command, foremost -t exe 010.010.010.010.04445-010.010.010.070.01044.
With this, Audit.txt showed another dll file, and when MD5'd, showed to be the same file as earlier:

```
Foremost started at Sat Oct  3 12:42:36 2020
Invocation: foremost -t exe 010.010.010.010.04445-010.010.010.070.01044
Output directory: /home/sansforensics/Documents/output
Configuration file: /etc/foremost.conf
------------------------------------------------------------------
File: 010.010.010.010.04445-010.010.010.070.01044
Start: Sat Oct  3 12:42:36 2020
Length: 813 KB (833091 bytes)

Num      Name (bs=512)         Size      File Offset     Comment

0:      00000000.dll         730 KB               4      04/03/2010 04:07:31
Finish: Sat Oct  3 12:42:36 2020

1 FILES EXTRACTED

exe:= 1
------------------------------------------------------------------

Foremost finished at Sat Oct  3 12:42:36 2020
sansforensics@siftworkstation: ~/Documents
$ md5sum output/dll/00000000.dll
b062cb8344cd3e296d8868fbef289c7c  output/dll/00000000.dll
```

10. **When was the TCP connection on port 4445 closed? (Provide the number of seconds since the beginning of the packet capture, rounded to tenths of a second. ie, 49.5 seconds)**

A: Here, I simply found the last ACK packet after the FINACK on the above used filter, which was sent at 198.4, indicating the connection closed.