SANS FORENSIC CHALLENGE: “ANN’S AURORA”

Sherri Davidoff, GCFA, GPEN
Eric Fulton, GWAPT
Jonathan Ham, CISSP, GCIA, GCIH

http://LakeMissoulaGroup.com

Lake Missoula Group, LLC
 Previously:
- Contest #1: “Ann’s Bad AIM” (8/12/09)
- Contest #2: “Ann Skips Bail” (10/10/09)
- Contest #3: “Ann’s AppleTV” (12/28/09)
- Contest #4: “The Curious Mr. X” (2/3/10)
- Contest #5: “Ms. Moneymany’s Mysterious Malware (4/1/10)

 Why do we do this?

 What do we do with the submissions?

 Where do we hope this goes?

 Answer: Building a Stronger Forensics Community
CONTEST #6: “ANN’S AURORA”

AKA the SANS Forensics Challenge!

• This time we teamed up with SANS specifically to do something for this Summit.
• Specifically, dealing with aspects of the APT in the wild!
• Over 60 submissions for one of our toughest challenges yet.
• Fabulous prizes, thanks to Rob, SANS, and some generous vendors!
CASE BACKGROUND

- Ann Dercover is after SaucyCorp's Secret Sauce recipe
- Tracks the lead developer, Vick Timmes
- Wants to remotely access SaucyCorp's servers
- She sees him log into his laptop (10.10.10.70) and VPN to SaucyCorp HQ
- Ann obtains a 0-day exploit for IE
- Launches a client-side spear phishing attack
- He clicks on the link....
  - She strikes!
THE CONTEST

• You are the investigator!

• YOUR MISSION:
  • Track down Ann's activities
  • Identify the original web request
  • Analyze the malware
  • Build a timeline of the attack
  • Carve out suspicious files
PRIZES

- Lenovo Ideapad Netbooks (2 Netbooks - 1 netbook per winner)
- Apple iPad - Sponsored by NetWitness Corporation
- Flip Video Recorder - Sponsored by MANDIANT Inc.
- F-Response TACTICAL (1 licensed copy) - Sponsored by F-Response
- Forensic Toolkit 3 (1 licensed copy) - Sponsored by AccessData Corp.
- Digital Forensics Magazine Subscriptions: Free print subscription for 12 months for the winner, and 2 digital online subscriptions for finalists. - Sponsored by Digital Forensics Magazine
- 2011 Digital Forensics/IR Summit Passes (3 passes - 1 pass per top three winners)
Q: What was the full URI of Vick Timmes' original web request?
A: http://10.10.10.10:8080/index.php
Q: 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?
A: vEI
Vick's computer made a second HTTP request for an object.
Q: What was the filename of the object that was requested?
A: index.phpmfKSxSANkeTeNrah.gif
Q: What is the MD5sum of the object that was returned?
A: df3e567d6f16d040326c7a0ea29a4f41
Q: When was the TCP session on port 4444 opened?
A: 1.3 s
Q: When was the TCP session on port 4444 closed?
A: 87.6s
Q: In packet 17, the malicious server sent a file to the client. What type of file was it?
A: Windows executable.
In packet 17, the malicious server sent a file to the client. What was the MD5 sum?
SOLVING THE CRIME

• Now we have the half-duplex reassembled stream payload
• Let's use foremost to carve out the data

```
$ foremost packet17.raw
Processing: packet17.raw
|*|
$ ls output/
  audit.txt  dll
$ file output/dll/00000000.dll
output/dll/00000000.dll: PE32 executable for MS Windows (DLL) (GUI) Intel 80386 32-bit
$ md5sum output/dll/00000000.dll
b062cb8344cd3e296d8868fbef289c7c  output/dll/00000000.dll
```

Q: In packet 17, the malicious server sent a file to the client. What was the MD5 sum?
A: b062cb8344cd3e296d8868fbef289c7c
SOLVING THE CRIME

• There is some odd stuff going on in here though. Let’s look at the port 4445 traffic...

• How often does the TCP initial sequence number (ISN) change?

cheetah:contests jonathan$ tcpdump -nnr evidence06.pcap 'port 4445'
reading from file evidence06.pcap, link-type EN10MB (Ethernet)
17:40:35.258314 IP 10.10.10.70.1037 > 10.10.10.4445: S 553522758:553522758(0) win 65535 <mss 1460,nop,nop,sackOK>
17:40:35.258396 IP 10.10.10.4445 > 10.10.10.70.1037: R 0:0(0) ack 553522759 win 0
17:40:35.594943 IP 10.10.10.70.1037 > 10.10.10.4445: S 553522758:553522758(0) win 65535 <mss 1460,nop,nop,sackOK>
17:40:35.594980 IP 10.10.10.4445 > 10.10.10.70.1037: R 0:0(0) ack 1 win 0
17:40:36.141827 IP 10.10.10.70.1037 > 10.10.10.4445: S 553522758:553522758(0) win 65535 <mss 1460,nop,nop,sackOK>
17:40:36.141872 IP 10.10.10.4445 > 10.10.10.70.1037: R 0:0(0) ack 1 win 0
17:40:36.142471 IP 10.10.10.70.1037 > 10.10.10.4445: S 553800369:553800369(0) win 65535 <mss 1460,nop,nop,sackOK>
17:40:36.142531 IP 10.10.10.4445 > 10.10.10.70.1037: R 0:0(0) ack 277612 win 0
17:40:36.688700 IP 10.10.10.70.1037 > 10.10.10.4445: S 553800369:553800369(0) win 65535 <mss 1460,nop,nop,sackOK>
17:40:36.688741 IP 10.10.10.4445 > 10.10.10.70.1037: R 0:0(0) ack 277612 win 0
17:40:37.235554 IP 10.10.10.70.1037 > 10.10.10.4445: S 553800369:553800369(0) win 65535 <mss 1460,nop,nop,sackOK>
17:40:37.235656 IP 10.10.10.4445 > 10.10.10.70.1037: R 0:0(0) ack 277612 win 0
17:40:37.236520 IP 10.10.10.70.1037 > 10.10.10.4445: S 554100968:554100968(0) win 65535 <mss 1460,nop,nop,sackOK>
17:40:37.236546 IP 10.10.10.4445 > 10.10.10.70.1037: R 0:0(0) ack 578211 win 0
17:40:37.782456 IP 10.10.10.70.1037 > 10.10.10.4445: S 554100968:554100968(0) win 65535 <mss 1460,nop,nop,sackOK>
17:40:37.782512 IP 10.10.10.4445 > 10.10.10.70.1037: R 0:0(0) ack 578211 win 0
17:40:38.329315 IP 10.10.10.70.1037 > 10.10.10.4445: S 554100968:554100968(0) win 65535 <mss 1460,nop,nop,sackOK>
17:40:38.329388 IP 10.10.10.4445 > 10.10.10.70.1037: R 0:0(0) ack 578211 win 0
SOLVING THE CRIME

• There is some odd stuff going on in here though. Let’s look at the port 4445 traffic...

• How about the IP IDs?

cheetah:contest4 $ tcpdump -nnv evidence06.pcap 'port 4445'
reading from file evidence06.pcap, link-type EN10MB (Ethernet)
17:40:35.258314 IP (tos 0x0, ttl 128, id 359, offset 0, flags [DF], proto TCP (6), length 48) 10.10.10.70.10.10.37 > 10.10.10.4445: S, cksum 0x000e (correct), 553522758:553522758(0) win 65535 <msg 1460,nop,nop,sackOK>
17:40:35.258398 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 48) 10.10.10.4445 > 10.10.10.70.10.10.37: R, cksum 0x3a8b (correct), 0x0(0) ack 553522759 win 0
17:40:35.594943 IP (tos 0x0, ttl 128, id 368, offset 0, flags [DF], proto TCP (6), length 48) 10.10.10.70.10.10.37 > 10.10.10.4445: S, cksum 0x000e (correct), 553522758:553522758(0) win 65535 <msg 1460,nop,nop,sackOK>
17:40:35.594986 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 48) 10.10.10.4445 > 10.10.10.70.10.10.37: R, cksum 0x3a8b (correct), 0x0(0) ack 553522759 win 0
17:40:36.141027 IP (tos 0x0, ttl 128, id 361, offset 0, flags [DF], proto TCP (6), length 48) 10.10.10.70.10.10.37 > 10.10.10.4445: S, cksum 0x000e (correct), 553522758:553522758(0) win 65535 <msg 1460,nop,nop,sackOK>
17:40:36.141072 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 48) 10.10.10.4445 > 10.10.10.70.10.10.37: R, cksum 0x3a8b (correct), 0x0(0) ack 1 win 0
17:40:36.142471 IP (tos 0x0, ttl 128, id 362, offset 0, flags [DF], proto TCP (6), length 48) 10.10.10.70.10.10.37 > 10.10.10.4445: S, cksum 0x0d19e (correct), 553500369:553500369(0) win 65535 <msg 1460,nop,nop,sackOK>
17:40:36.142531 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 48) 10.10.10.4445 > 10.10.10.70.10.10.37: R, cksum 0x3a8b (correct), 0x0(0) ack 277612 win 0
17:40:36.688708 IP (tos 0x0, ttl 128, id 363, offset 0, flags [DF], proto TCP (6), length 48) 10.10.10.70.10.10.37 > 10.10.10.4445: S, cksum 0x0d19e (correct), 553500369:553500369(0) win 65535 <msg 1460,nop,nop,sackOK>
17:40:36.688741 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 48) 10.10.10.4445 > 10.10.10.70.10.10.37: R, cksum 0x3a8b (correct), 0x0(0) ack 277612 win 0
17:40:37.235555 IP (tos 0x0, ttl 128, id 364, offset 0, flags [DF], proto TCP (6), length 48) 10.10.10.70.10.10.37 > 10.10.10.4445: S, cksum 0x0d19e (correct), 553500369:553500369(0) win 65535 <msg 1460,nop,nop,sackOK>
17:40:37.235656 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 48) 10.10.10.4445 > 10.10.10.70.10.10.37: R, cksum 0x3a8b (correct), 0x0(0) ack 277612 win 0
There is some odd stuff going on in here though. Let’s look at the port 4445 traffic...

How about the source ports?
There is some odd stuff going on in here though. Let’s look at the port 4445 traffic...

So did anything ever answer?

```
cheetah:contests jonathan$ tcpdump -nnr evidence06.pcap 'port 4445 and tcp[13] & 0x12 == 0x12'
reading from file evidence06.pcap, link-type EN10MB (Ethernet)
17:42:02.985580 IP 10.10.10.10.4445 > 10.10.10.70.1044: S 1436350344:1436350344(0) ack 197937316
5 win 5840 <mss 1460,nop,nop,sackOK>
```

Ouch!!!
Subsequently, the malicious server sent an executable file to the client on port 4445. What was the MD5 sum?
SOLVING THE CRIME

• Again, now we have the half-duplex reassembled stream payload
• Let's use foremost to carve out the data

```
$ foremost 4445.raw
Processing: 4445.raw
|*|
$ ls output/
audit.txt  dll
$ file output/dll/00000000.dll
output/dll/00000000.dll: PE32 executable for MS Windows (DLL) (GUI) Intel 80386 32-bit
$ md5sum output/dll/00000000.dll
b062cb8344cd3e296d8868fbef289c7c  output/dll/00000000.dll
```

Subsequently, the malicious server sent an executable file to the client on port 4445. What was the MD5 sum?
A: b062cb8344cd3e296d8868fbef289c7c
Q: When was the TCP session on port 4445 opened?
A: 123.7 s
Q: When was the TCP session on port 4445 closed?  
A: 198.4s
BEHIND THE CURTAIN:
EXPLOIT DESCRIPTION

• Backtrack vs. XP SP2
• Metasploit hosting Aurora Exploit
• XP connects to Metasploit Web Server
• Metasploit exploits browser, loads stager
• Stager downloads reverse TCP meterpreter
• Meterpreter makes SSL connection back to metasploit listener (pwnage dance!)
THE ATTACK

• The attacker immediately loads another meterpreter shell which attempts to connect back every 15 seconds

• Then the attacker goes to work downloading files and a screenshot from the victim computer.
THE ATTACK (2)

• The attacker loses the first shell (user closed browser) but reconnected via 2nd shell

• Many Encrypted Commands (Print screen, download file, etc.)

• Attacker has full control via meterpreter payload
BEHIND THE ENCRYPTION

• What isn’t visible to the forensic analyst
  – Lists computer information (to ensure correct target)
  – Takes a screenshot of the desktop to see what the user is doing
  – Attacker prints working directory, and moves to the directory of the Secret Sauce!
  – Attacker downloads file
ISSUES WITH THE ATTACK

• Victim is required to connect to server hosting Aurora

• Packing and obfuscation required for Meterpreter

• Connect back time (15s)

• Requires old OS (kinda)
IMPACTS

• The usual APT issues
• The internet is a war field (as always)
• Computer forensics is important
• Network forensics is just as important
• For APT threats, a team of skilled individuals is necessary to cover the breadth of knowledge required to unravel an attack.
WINNER & FINALISTS

• Winner!
  – Wesley McGrew

• Finalists:
  – Erik Hjelmvik (SILVER)
  – Leendert Pieter van Drimmelen (BRONZE)
  – Eric Kollmann
  – Jeff Wichman
  – Ruben Recabarren
  – Julian Anton
  – Candice Quates
WESLEY MCGREW:
PCAPLINE

- Automatically parses a packet capture and creates a multilayered HTML report
- Features (as described by Wesley):
  - HTML reports that allow for easy navigation/importing into a larger report
  - Generates a summary of flows between hosts on the network
  - Flows are broken up by segments representing parts of the conversation
  - Segments are dissected, carved, hashed. Currently, Pcapline supports:
    - HTTP GET Requests
    - HTTP Responses - Extracts and hashes contents
    - Forensics Challenge malware file transfers
    - Other data segments are presented as text, hex dumps, or raw content
- Pcapline is very easy to use! Runs on MacOS X, Linux or UNIX. May also work in Windows (try Cygwin)
- This script is extremely easy to run
  $ ./pcapline.py ..../evidence06.pcap
### Pcapline Output for evidence06.pcap

**Notes**

- Times measured in seconds elapsed since first packet captured
- The time of first packet capture may not be the time that the capture was started
- Packet numbers refer to the input pcap file, and are indexed starting at '1', to match Wireshark's display
- Flows of data are sorted by start time in ascending order
- Non-data packets that arrive in the middle of data segments (normally just ACKs) are omitted from flow reports for brevity

<table>
<thead>
<tr>
<th>Flow #</th>
<th>Protocol</th>
<th>Host 1</th>
<th>Host 2</th>
<th>Start Time</th>
<th>Last Packet Time</th>
<th>Bytes of Data</th>
<th>Packet #s</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>TCP</td>
<td>10.10.10.70:1035</td>
<td>10.10.10.8080</td>
<td>0.000000</td>
<td>65.564956</td>
<td>6677</td>
<td>1 - 12, 1343</td>
</tr>
<tr>
<td>0002</td>
<td>TCP</td>
<td>10.10.10.70:1036</td>
<td>10.10.10.4444</td>
<td>1.265851</td>
<td>87.587480</td>
<td>1335961</td>
<td>13 - 1152, 1169 - 1173, 1188 - 1197, 1202 - 1206, 1219 - 1228, 1244 - 1312, 1344 - 1502, 1561 - 1565</td>
</tr>
<tr>
<td>0003</td>
<td>TCP</td>
<td>10.10.10.70:1037</td>
<td>10.10.10.4445</td>
<td>35.947030</td>
<td>40.986836</td>
<td>0</td>
<td>1153 - 1168, 1174 - 1187</td>
</tr>
<tr>
<td>0004</td>
<td>TCP</td>
<td>10.10.10.70:1038</td>
<td>10.10.10.4445</td>
<td>47.732517</td>
<td>52.689938</td>
<td>0</td>
<td>1198 - 1201, 1207 - 1218, 1229 - 1242</td>
</tr>
<tr>
<td>0005</td>
<td>UDP</td>
<td>10.10.10.70:138</td>
<td>10.10.10.255:138</td>
<td>55.409175</td>
<td>55.409175</td>
<td>119</td>
<td>1243</td>
</tr>
<tr>
<td>0006</td>
<td>TCP</td>
<td>10.10.10.70:1039</td>
<td>10.10.10.4445</td>
<td>59.462957</td>
<td>64.502478</td>
<td>0</td>
<td>1313 - 1342</td>
</tr>
<tr>
<td>0007</td>
<td>TCP</td>
<td>10.10.10.70:1040</td>
<td>10.10.10.4445</td>
<td>71.257992</td>
<td>76.205550</td>
<td>0</td>
<td>1503 - 1532</td>
</tr>
<tr>
<td>0008</td>
<td>TCP</td>
<td>10.10.10.70:1041</td>
<td>10.10.10.4445</td>
<td>82.993986</td>
<td>88.018132</td>
<td>0</td>
<td>1533 - 1560, 1566 - 1567</td>
</tr>
<tr>
<td>0009</td>
<td>TCP</td>
<td>10.10.10.70:1042</td>
<td>10.10.10.4445</td>
<td>94.878166</td>
<td>100.049324</td>
<td>0</td>
<td>1568 - 1597</td>
</tr>
<tr>
<td>0010</td>
<td>TCP</td>
<td>10.10.10.70:1043</td>
<td>10.10.10.4445</td>
<td>106.838688</td>
<td>111.971168</td>
<td>0</td>
<td>1598 - 1627</td>
</tr>
<tr>
<td>0011</td>
<td>TCP</td>
<td>10.10.10.70:1044</td>
<td>10.10.10.4445</td>
<td>118.746261</td>
<td>198.441738</td>
<td>844469</td>
<td>1628 - 2554</td>
</tr>
</tbody>
</table>

Example report: http://mcgrewsecurity.com/codedump/evidence06.pcap_output/
WESLEY MCGREW: PCAPLINE

- Automatically exports each flow
- Displays:
  - Hosts
  - Ports
  - Protocol
  - # Bytes
  - Packet #s
  - Timing info

Flow 1

Summary

- Hosts
  - 10.10.10.70:1035
  - 10.10.10.10:8080
- Start time: 0.000000 seconds
- Last packet sniffed time: 65.564956 seconds
- Protocol: TCP
- Bytes of data: 6677 bytes
- Packet #s: 1 - 12, 1343
- .pcap of this flow: packets.pcap
WESLEY MCGREW:  
PCAPLINE

- Drill down  
- View data from packets  
- Automatically carves files and displays MD5 – even the PE32 executables  
- Displays TCP flags
ERIK HJELMVIK:  
NETWORK MINER .92

- Updated for Contest #6
- Now properly extracts HTTP file transfers
- Better guesses server/client when 3-way handshake is missing
ERIK HJELMVIK: NETWORK MINER .92

Here you can see how Network Miner automatically identified the HTTP file transfers.

- There is an HTML page and a GIF.
- Lots of handy details (filename, size, etc).
- Right click to open the file.
ERIK HJELMVIK: NETWORK MINER .92

- Network Miner automatically identified the TCP sessions in the packet capture
- Again, lots of details
  - Frame #, Client, hostname, source & dest IP & port, date and time, etc
ERIK HJELMVIK: NETWORK MINER .92

- Drill down into details of each frame
- Network Miner: http://networkminer.sourceforge.net/
LEENDERT PIETER VAN DRIMMELSEN: STREAM_TS.PY

- Automatically displays TCP connection established/closed times
- Accepts tshark filters

```python
$ python stream_ts.py ..//evidence06.pcap 'tcp.port==4444'

Connection: 10.10.10.70:1036 -> 10.10.10.10:4444
Connection established at: Apr 28, 2010 17:40:00.577135000. Relative timestamp: 1.3 (1.265851000)
Connection closed at: Apr 28, 2010 17:41:26.898764000. Relative timestamp: 87.6 (87.587480000)

$ python stream_ts.py ..//evidence06.pcap 'tcp.port==4445'

Connection: 10.10.10.70:1044 -> 10.10.10.10:4445
Connection established at: Apr 28, 2010 17:42:02.985483000. Relative timestamp: 123.7 (123.674199000)
```
LEENDERT PIETER VAN DRIMMELLEN: PEXTRACT.C

- Extracts PE files from packet captures or incoming traffic
- Accepts BPF filters
- Also tries to find executables that are XOR obfuscated

```
leendert-pieter@lenovo:/contest6$ ./pextract -f evidence06.pcap "port 4445"
peextract: reading from evidence06.pcap [port 4445]
Connection: 10.10.10.70:1044-10.10.10.10:4445
Saving MS executable as 10.10.10.70:1044-10.10.10.10:4445.0._exe (0 used as xor key)

leendert-pieter@lenovo:/contest6$ md5sum 10.10.10.70\:1044-10.10.10.10\:4445.0._exe
b062cb8344cd3e296d8868fbeb289c7c  10.10.10.70:1044-10.10.10.10:4445.0._exe
```
LEENDERT PIETER VAN DRIMMELLEN:  
ANALYSE_SYN_PACKETS.PY

- Calculates how often an IP or TCP field changes
- Specify address, port and tshark field

```bash
$ python analyse_syn_packets.py ..../evidence06.pcap 10.10.10.10 4445 tcp.seq
Found 120 packet(s) that match the supplied parameters
tcp.seq changes every 3 packet(s).
$
$ python analyse_syn_packets.py ..../evidence06.pcap 10.10.10.10 4445 ip.id
Found 120 packet(s) that match the supplied parameters
ip.id changes every 1 packet(s).
$
$ python analyse_syn_packets.py ..../evidence06.pcap 10.10.10.10 4445 tcp.srcport
Found 120 packet(s) that match the supplied parameters
tcp.srcport changes every 15 packet(s).
tcp.srcport changes every 12 second(s).
```
ERIC KOLLMANN: CONTEST6.PL

- Get info about conversations:
- Eric's tools are at: http://myweb.cableone.net/xnih/download
ERIC KOLLMANN: CONTEST6.EXE

- List info about packets
- Limit by flags (ie. "S" for SYN, "FA" for FIN/ACK)
- Supports BPF filters
- Also see "mzcarver" for carving out executable

```
contest6.exe -r evidence06.pcap -l S
13 17:40:00,577135 S 10.10.10.70:1036 -> 10.10.10.10:4444 53 72acc97a 00000000
1153 17:40:35,258314 S 10.10.10.70:1037 -> 10.10.10.10:4445 359 20fe1646 00000000
1155 17:40:35,594943 S 10.10.10.70:1037 -> 10.10.10.10:4445 360 20fe1646 00000000
1157 17:40:36,141827 S 10.10.10.70:1037 -> 10.10.10.10:4445 361 20fe1646 00000000
```
CONGRATULATIONS, AND THANKS!

• Once again, we were blown away by the submissions.
• Some nifty new tools.
• Better even, some **amazingly** thorough write-ups
  - Excellent reasoning
  - Excellent methodologies
  - Excellent analysis!
• Stay tuned, and we’ll keep ’em coming!
  - ForensicsContest.com
SANS FORENSIC CHALLENGE: “ANN’S AURORA”

Q?

Sherri Davidoff, GCFA, GPEN
Eric Fulton, GWAPT
Jonathan Ham, CISSP, GCIA, GCIH

http://LakeMissoulaGroup.com

ForensicsContest.com Contest #6: “Ann’s Aurora” © 2010 Lake Missoula Group, LLC