The Fool, The BeEF and The Butcher

Pasquale Stirparo
SANS DFIR Summit, Prague 2015
$whoami

- Pasquale Stirparo
  - M.Sc. in Computer Engineering
  - Ph.D. in Information Security
- GCFA, GREM, OPST, OWSE, ECCE
- Senior Information Security Incident Response Engineer @ Fortune 500
- “Learning iOS Forensics” book co-author
- SANS Aficionado :)
- Maintainer of mac4n6 Artifacts Location project (https://github.com/pstirparo/mac4n6)
- Contacts: pstirparo@gmail.com, @pstirparo

Disclaimer: views and opinions expressed are my own and do not reflect those of my employer
Motivation

• According to SANS Institute “95% of enterprise data breaches start with a spear phishing attack”¹

• In 2014, The Telegraph conducted a “real life experiment” with one of their journalist, were professional ethical hackers were able to take over the computer of the journalist and collect personal information²

• In this case hackers used BeEF, social engineering and phishing emails to achieve their goal

² http://www.telegraph.co.uk/technology/internet-security/11153381/How-hackers-took-over-my-computer.html
THE FOOL
THE BeEF
AND THE BUTCHER

Pasquale Stirparo - SANS DFIR Summit, Prague 2015
BeEF: Browser Exploitation Framework

• Penetration testing tool focusing on web browsers.

• It hooks the browser via a javascript inside a normal HTML page; it exposes RESTful API that allow BeEF to be scripted through HTTP/JSON requests.

• All running inside the browser, hence leaving the chances of finding file system artifacts very low.
Attack Scenario

1. User receives an email with a link to a “legit” but infected website
2. Once clicked, the browser is hooked and BeEF extract a complete fingerprint of it
3. A fake Facebook logout alert is prompted to steal credentials via the “Pretty Theft” module
4. An invisible iframe is created, overlapping the current page, which opens a connection to a server where Metasploit is listening
5. Once connected, metasploit delivers a meterpreter by exploiting a recent flash vulnerability
You should be hooked into BeEF.

Have fun while your browser is working against you.

These links are for demonstrating the "Get Page HREFs" command module:
- The Browser Exploitation Framework
- hackers.org homepage
- Slashdot

Have a go at the event logger. Insert your secret here:

You can also load up a more advanced...

**Facebook Session Timed Out**
Your session has timed out due to inactivity.
Please re-enter your username and password to login.

Email:
Password:

Log in
Analysis Methodology

Four states to be checked
1. Clean state with Firefox Browser ON
2. After browser has been successfully hooked
3. After BeEF modules in action: credentials stolen
4. After exploitation with BeEF+Metasploit

For all of them, a memory dump and a disk dump have been taken, plus registry and file system changes monitored (regshot + poweshell script)
Results so far...

- BeEF is stealthy:
  - No files dropped
  - No new processes spawned (only metasploit flash exploit did it)

- Initial quick analysis on the disk dump found artifacts only on the browser cache
  - C:\Users\p4c0\AppData\Local\Mozilla\Firefox\Profiles\w3nf5opu.default\cache2\entries
  - More comprehensive analysis still to be done

- BeEF code is retrievable from the memory

$ python vol.py -f beefed.dmp --profile=Win7SP0x86 pslist
$ python vol.py -f beefed.dmp --profile=Win7SP0x86 memdump -p 4040 -D
$ strings dest_dir/4040.dmp > strings_ff_beefed.txt
Results so far...

Code in memory, so I decided to go with Yara:

- Yara rules to spot hooked browser
  - 1 Yara rule “hand written”
  - 2 Yara rules generated with yarGen (awesome tool)
  - All of them worked

- A Yara rule specific for the “pretty theft” module
  - Now you may also know which attack has been carried on

- I will release them after this talk on my GitHub page
  - Consider them still in “beta”
  - https://github.com/pstirparo/yara_rules
Results so far...

- Lesson learned: when writing yara rules for memory hunting, put always both "wide ascii" to your strings

```
$ python vol.py -f beefed.dmp --profile=Win7SP0x86 yarасan -p 4040 --yarafile=beef_pretty_theft.yar
[... snip...]
Rule: src_ptheft_command
Owner: Process firefox.exe Pid 4040
0x0e6c6d72 73 00 6e 00 65 00 61 00 6b 00 79 00 64 00 69 00 s.n.e.a.k.y.d.i.
0x0e6c6d82 76 00 2e 00 69 00 6e 00 65 00 72 00 48 00 v...i.n.n.e.r.H.
0x0e6c6d92 54 00 4d 00 4c 00 3d 00 20 00 27 00 64 00 T.M.L.=...'.<.d.
0x0e6c6da2 69 00 76 00 20 00 69 00 64 00 3d 00 22 00 i.v...i.d.=."w.
0x0e6c6db2 69 00 6e 00 64 00 6f 00 77 00 5f 00 63 00 6f 00 i.n.d.o.w._.c.o.
0x0e6c6dc2 6e 00 74 00 61 00 69 00 6e 00 65 00 72 00 22 00 n.t.a.i.n.e.r.".
0x0e6c6dd2 20 00 27 00 2b 00 77 00 69 00 6e 00 64 00 6f 00 ..'+.w.i.n.d.o.
0x0e6c6de2 77 00 62 00 6f 00 72 00 64 00 65 00 72 00 2b 00 w.b.o.r.d.e.r.+.
[... snip...]
```
Project Roadmap

• Full Windows coverage (currently ongoing)
  o Hiberfil.sys and pagefile
  o Registry analysis
  o Timeline analysis with Plaso

• Yara
  o Writing rules for most/all BeEF modules
  o Verify robustness against heavy obfuscation
  o Heavy testing to better verify false positives' rate

• Replicate same analysis on Mac OS X

• Replicate on Mobile (iOS, Android, WP)
  o Yes, BeEF can hook your phone browser too

• Observe for possible patterns on network traffic
Thank You