• Malware Analyst at Bromium (now)
• Threat research in Bromium Labs
• Former incident responder, SOC analyst
• BSides Bristol co-organiser
• Amateur radio operator
• @cryptogramfan

• Malware Analyst at Bromium
• Assistant Professor at Dakota State University
• Teach malware analysis, software exploitation, reverse engineering, and penetration testing
• Director of Training for the Open Information Security Foundation (OISF)
Malware as a Service
The growth of the underground economy has led to increased collaboration and dependencies between criminal actors

- Hacking, carding, cracking forums
- Discord and Telegram channels
- Marketplaces

The model describing the ecosystem of specialised goods and services bought and sold by criminal actors is known as malware as a service (MaaS)

- Bulletproof hosting, exploits, distribution and downloaders

MaaS has enabled actors to purchase these items from third parties without needing to develop the capability internally

Example of this model in action include GozNym (dismantled May 2019 following law enforcement operation), Necurs and Emotet
Malware as a Service – Previous Work

2015

2017

2018

2019
Case Study #1
Emotet Distribution
Emotet is a modular loader that was first identified in the wild in 2014

Originally was a banking Trojan that stole financial information from online banking sessions through man-in-the-browser (MITB) attacks

Observed since 2017 distributing other malware families, such as IcedID, Zeus Panda and TrickBot, instead of its own banking module

Shift in tactics, techniques and procedures (TTPs) suggests a change in Emotet’s business model
  - Sells access to its botnet infrastructure to other threat actors instead of monetising stolen financial information

As of 2019, Emotet is consistently one of the top threats isolated among Bromium customers
Case Study #1 – Emotet Distribution – Possible Business Model

Emotet’s operators

TrickBot’s operators

Organised Crime Group B: Banking Trojan

Exfiltrated Financial Data

Network Admin(s)

Intrusion Specialist(s)

Data Miner(s)

Automatic Vendor Cart websites

Bank Account Details

Mule Herders

Money Mules

Bank Accounts

Consolidation Accounts

Financial Sector

Public

Industry

ISPs
MSSPs

Government

A’s C2

Group A’s C2
(owned and compromised)

Exfiltrated Email Data

Phishing Campaigns Deploying A’s Malware

Phishing Specialist(s)

A’s Malware Deploying B’s Banking Trojan

Exfiltrated Email Data

Group B’s C2

Intrusion Specialist(s)

Data Miner(s)

Automatic Vendor Cart websites

Bank Account Details

Mule Herders

Money Mules

Bank Accounts

Group B’s C2

Consolidation Accounts

Distribution

Profit

Losses and reputational damage for victims

Organised Crime Group A: Malware Loader/Distributor

Money Specialist(s)
• Emotet payloads are hosted on compromised web servers
• Base64 encoded data in a PHP page (index.php)
• Perception that Emotet binaries are mainly hosted on compromised WordPress web sites
  • https://blogs.quickheal.com/beware-your-website-might-be-delivering-emotet-malware/
• What does the data tell us about how Emotet’s actors are compromising web servers?
Identified web servers used to distribute Emotet from September 2018 to October 2019

Data sources included our own automated maldoc analysis (static and dynamic) and public reports from the Cryptolaemus team (who are amazing)
  - Cuckoo with JPCERT/CC’s MalConfScan, CAPE and Bromium sandboxes + Python scripts
  - NetworkX and Graphviz

25,594 URLs used to host Emotet binaries over period (average ~65 fresh URLs per day)

WhatCMS and Shodan (passive) APIs to enumerate technologies present on servers

Successfully enumerated technologies on 9,001 web servers
Case Study #1 – Emotet Distribution – Technologies

Cf. 79% websites run PHP (W3Techs, October 2019)

Sample size = 9001

WordPress dependencies

WordPress plugin

WordPress plugin
Case Study #1 – Emotet Distribution – WordPress

Top WordPress Versions on Emotet Distribution Servers
September 2018 to October 2019

- 5.2.4 (14/10/2019)
- 5.1.3 (19/10/2019)
- 5.0.7 (19/10/2019)
- 4.9.1 (19/10/2019)
- 5.2 (18/05/2019)
- 5 (11/12/2018)
- 4.8.1 (05/09/2019)
- 5.2.7 (20/06/2019)
- Other
168 PHP versions identified

61% of PHP installs were out of support

Data does not point to Elementor or WooCommerce being exploited based on known CVEs and version distribution

WooCommerce and Elementor XSS bugs, but limited scope
  - CVE-2018-18379
  - CVE-2015-2069
  - CVE-2014-6313

PHP represents the largest target for compromise by Emotet’s operators
Case Study #1 – Emotet Distribution – CVE Network Analysis

Cluster of SSH vulnerabilities

Cluster of PHP vulnerabilities
Case Study #1 – Emotet Distribution – Top CVEs

- Top CVEs found apply to SSH, but all none are directly applicable to the aims of Emotet’s actors

<table>
<thead>
<tr>
<th>CVE</th>
<th>Type</th>
<th>Applicable</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2017-15906</td>
<td>SSH</td>
<td>N</td>
<td>6%</td>
</tr>
<tr>
<td>CVE-2016-10708</td>
<td>SSH</td>
<td>N</td>
<td>4%</td>
</tr>
<tr>
<td>CVE-2016-0777</td>
<td>SSH</td>
<td>N</td>
<td>4%</td>
</tr>
<tr>
<td>CVE-2014-1692</td>
<td>SSH</td>
<td>M</td>
<td>4%</td>
</tr>
<tr>
<td>CVE-2011-5000</td>
<td>SSH</td>
<td>N</td>
<td>4%</td>
</tr>
<tr>
<td>CVE-2011-4327</td>
<td>SSH</td>
<td>M</td>
<td>4%</td>
</tr>
<tr>
<td>CVE-2010-5107</td>
<td>SSH</td>
<td>N</td>
<td>4%</td>
</tr>
<tr>
<td>CVE-2010-4755</td>
<td>SSH</td>
<td>N</td>
<td>3%</td>
</tr>
<tr>
<td>CVE-2012-0814</td>
<td>SSH</td>
<td>M</td>
<td>3%</td>
</tr>
</tbody>
</table>
Case Study #1 – Emotet Distribution – PHP & WordPress Plugins

- Exact attack method is difficult to know, but the data suggests:

<table>
<thead>
<tr>
<th>Attack Type</th>
<th>Likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHP exploit</td>
<td>Likely</td>
</tr>
<tr>
<td>Password guessing</td>
<td></td>
</tr>
<tr>
<td>Misconfigurations</td>
<td></td>
</tr>
<tr>
<td>WordPress exploit</td>
<td></td>
</tr>
<tr>
<td>WordPress plugin exploit</td>
<td></td>
</tr>
<tr>
<td>Apache/Nginx/LiteSpeed exploit</td>
<td>Unlikely</td>
</tr>
<tr>
<td>SSH exploit</td>
<td></td>
</tr>
<tr>
<td>Other exploit</td>
<td></td>
</tr>
<tr>
<td>MySQL exploit</td>
<td>Unknown</td>
</tr>
</tbody>
</table>
Case Study #2
Ostap
Ostap is a commodity downloader written in JScript

JScript is an interpreted language and close relative to JavaScript that runs natively on Windows through Windows Script Host (wscript.exe)

After Emotet went offline, TrickBot’s operators used it as a downloader

Notable for:
- Large size at nearly 35,000 lines
- Virtual machine detection and anti-analysis measures using process and hostname blacklists
- Quite low detection rate of 6/55 (11%) on VT
- Apologetic messages to researchers

```plaintext
mallory@mallory-pc:~$ wc ~/Samples/2angola.Jse.beautified
34757 166487 1760029 /home/mallory/Samples/2angola.Jse.beautified
```

![Image of virus detection screen]

Case Study #2 – Ostap – Introduction
Case Study #2 – Ostap – Uptick Summer 2019

Emotet malspam activity pauses late May ‘19

Emotet malspam activity resumes mid Sept ‘19

Threat actors looking for alternative downloaders

↑ Ostap

Emotet Detections, April to October 2019
Case Study #2 – Ostap – Deobfuscation

• Ostap checks that it is running in a sandbox, which can complicate dynamic analysis
• Deobfuscation approach:
  1. Beautify
  2. Identify code structure, key variables and functions
  3. Analyse of character code calculation functions
  4. Automate with Python

<table>
<thead>
<tr>
<th>Process Blacklist</th>
<th>Hostname Blacklist</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgentSimulator.exe</td>
<td>Emily</td>
</tr>
<tr>
<td>anti-virus.EXE</td>
<td>HANSPETER-PC</td>
</tr>
<tr>
<td>BehaviorDumper</td>
<td>HAPUBWS</td>
</tr>
<tr>
<td>BennyDB.exe</td>
<td>Hong Lee</td>
</tr>
<tr>
<td>ctfmon.exe</td>
<td>IT-ADMIN</td>
</tr>
<tr>
<td>fakepos_bin</td>
<td>JOHN-PC</td>
</tr>
<tr>
<td>FrzState2k</td>
<td>Johnson</td>
</tr>
<tr>
<td>gemu-ga.exe</td>
<td>Miller</td>
</tr>
<tr>
<td>ImmunityDebugger.exe</td>
<td>MUELLER-PC</td>
</tr>
<tr>
<td>KMS Server Service.exe</td>
<td>Peter Wilson</td>
</tr>
<tr>
<td>ProcessHacker</td>
<td>SystemIT</td>
</tr>
<tr>
<td>proexp</td>
<td>Timmy</td>
</tr>
<tr>
<td>Proxifier.exe</td>
<td>WIN7-TRAPS</td>
</tr>
<tr>
<td>python</td>
<td></td>
</tr>
<tr>
<td>tcpdump</td>
<td></td>
</tr>
<tr>
<td>VBoxService</td>
<td></td>
</tr>
<tr>
<td>VBoxTray.exe</td>
<td></td>
</tr>
<tr>
<td>VmRemoteGuest</td>
<td></td>
</tr>
<tr>
<td>vmtoolsd</td>
<td></td>
</tr>
<tr>
<td>VMware2B.exe</td>
<td></td>
</tr>
<tr>
<td>VzService.exe</td>
<td></td>
</tr>
<tr>
<td>winace</td>
<td></td>
</tr>
<tr>
<td>Wireshark</td>
<td></td>
</tr>
</tbody>
</table>
• Visual Basic for Applications (VBA) macro writes JScript to disk on a single line

• Reformat and add indentations back to the script using Einar Lielmanis’s JS Beautifier tool, which also works for JScript because they share a similar syntax

$ js-beautify 2angola.Jse > 2angola.Jse.beautified
Case Study #2 – Ostap – Deobfuscation – Identify Structure

- Identify the functions responsible for deobfuscating the downloader.

- The script includes many junk variables that aren’t used anywhere and can be removed.

- Often it’s possible to distinguish variables that have been automatically generated by an obfuscator from meaningful ones because their naming convention will differ.

```javascript
AmUzEmyself86ko = 'undefined';
AmUzEdoom2ko = '5RleMVU11SM17C6DAzaDFewDkhEft3khScSo8NyuYf9hb6igkDVHEWTNyTvqPNqrcXZ8fsZs1Rpf_57DRk4mY05bFB0E5WO5AOlouXy2ogaDznFmWPZQmgSmkluVY_Ou9C9d1YdPXMEmAnS3K57d15p0w09eMLD8Kcfua7zAG049kqgvUDVB8xEp1_CiCn6dr64TMpuNFM_A22NFACRuiAmb17zeoi7WRunIQgUaWHTwExaE27zXy7pL4yfid01d4yxkx8P36YREO0Z91uybAvDI5Ps67tvv3QC4Bpy_kccvosBMfUh9MwohVZUyBz1AmUzEconduced53ko = 'undefined';
AmUzEhatred64ko = 'undefined';
AmUzEthat71ko = 'undefined';
AmUzEwell57ko = 'undefined';
AmUzEagain15ko = 'undefined';
AmUzEfilled81ko = 'undefined';
AmUzElessen33ko = 'undefined';
AmUzEdiscussion57ko = 'undefined';
AmUzEfailiances20ko = 'undefined';
AmUzEunderstand93ko = 'undefined';
gunsder = ['fr' + 'om'] + ''; 
```

All these variable assignments are junk code, except for “gunsder”. It’s promising because it contains the string “from” and is referenced 2,515 times.
“Gunsder” is concatenated with other strings to reference the fromCharCode() method that converts a Unicode character code into a character.

The function below uses arithmetic operators to calculate a Unicode character code from the values stored in an array called pkkwrit4.

The Unicode character code and the character $h$ are then supplied to the fromCharCode function, which returns a Unicode character ($f$).

Each character in the downloader has its own function to calculate its character code (this sample had 7,540 functions).

```javascript
// fromCharCode
function fromCharCode(input, charh) {
  try {
    jiucome_8(input, charh);
  } catch (e) {
    if (true & & charh !== 'h') {
      return 1;
    } else {
      return String(['fromCharCode'](input));
    }
  return false;
}

var AmUzEvictory16 = String((function() {
  var pkkwrit4 = []; pkkwrit4[0] = 1; pkkwrit4[1] = 103;
  return xxqneol(pkkwrit4[1] - pkkwrit4[0], 'h');
});
```
• In the functions that calculate the Unicode character codes, the final character code value is always calculated using the elements at the same two indices of an array, e.g. 0 and 1
• Some arithmetic is performed on these elements before they are supplied to the fromCharCode function
• Python’s re module to write regular expressions that match the elements in each array at index 0 and 1 and store them in lists
• Next, we’ll clean up the matches using the re.sub() function and then convert them into integers
• We can then use Python’s zip() function to perform the arithmetic on the values in the index 0 and 1 lists
• Finally, the script converts the character codes into Unicode characters, removes line breaks and prints the result
Case Study #2 – Ostap – deobfuscate_ostap.py

- deobfuscate_ostap.py is available from my GitHub page:
  - https://github.com/cryptogramfan/Malware-Analysis-Scripts/blob/master/deobfuscate_ostap.py
Case Study #2 – Ostap – Campaign Delivering TrickBot

Ostap sample

Web server hosting TrickBot binary

AS15476
Netname: King-Servers

July to August 2019
Conclusion
• As the underground economy grows, malware as a service will continue to create opportunities for criminals to make money from selling their services and wares to other criminals

• Malware as a service has reduced the barrier to entry to cybercrime because it’s now easier to outsource missing capabilities, such as malware distribution

• However, increased inter-dependencies between malicious actors have also made criminal networks more vulnerable to disruption by law enforcement

• It’s an exciting area of research. We hope you found the talk interesting!
Thank you!

@cryptogramfan
@jstrosch