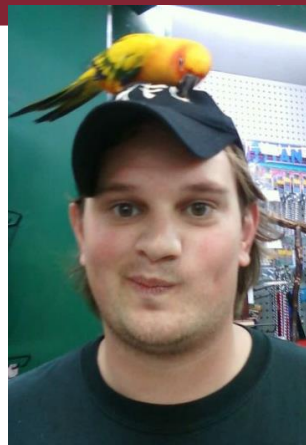


All Your Metadatas Are Belong To Me: Reverse Engineering Emails on an Enterprise Level

Ronnie Tokazowski
Senior Researcher
PhishMe, Inc.

About Me

```
~ # cat intro.py
#!/usr/bin/env python
a = ['x41x41x90', '@iHeartMalware']
h = ['RE', 'crypto', 'hacking stuff']
c = ['OSCP', 'GREM']
p = ['APT / cybercrime trolling']
d = ['Project Dyre']
```



What We Will Cover

- Extract metadata / reverse engineer an email
- Saving all the things (NSA style)
- Yara all the things
- Demos!
- Code releases
- Closing

Reverse Engineering

- Take things apart
- Break things
- Gather findings
- Report findings



Information Gathered

- Spices used for a particular sauce
- Domains
- IP addresses
- File metadata (author, date created)
- Encoding algorithms
- PE timestamps



Use Cases for Reported Data

- Searching for traffic to C2 servers
- Blocking domains found in malware
- Further understanding of a cryptographic algorithm
- Network intelligence (APT1)

Enough Theoretical...Let's Get to It!



Reverse Engineering Goals

- Obtain metadata about a given email chunk
- Hash everything! (you will want it later...)
- Deep-dive and find bad stuff
- Alerting or other event correlation?
- Good / actionable intelligence





Reverse Engineering Toolkit: Yara

Tool used for malware classification. Has a HUGE potential for other uses.

```
rule PM_Paypal_Spam {  
  strings:  
    $a1 = "46.165.252.13"  
    $a2 = "@paypal.com" nocase  
  condition:  
    any of them }
```

Yara can be downloaded from: <http://plusvic.github.io/yara/>

Reverse Engineering Toolkit: Yara (cont.)

- You can see the sig trip

```
ERROR scanning -s: could not open file
Ronnie@Ronnies-MacBook-Pro ~/Desktop $ yara -s yara_sig.yar my_email.eml
PM_Paypal_Spam my_email.eml
0x126:$a2: @peypal.com
0x17a:$a2: @peypal.com
0x18f:$a2: @peypal.com
```

Cat Pipe ./stdin

- Scanning an email with yara.exe is cool, but there are limitations.
 - Nothing decoded
 - One layer
 - Quotable printable text breaks...EVERYTHING
 - style=3D'margin:=0in;margin-bottom:.0001pt'><span style=3D'

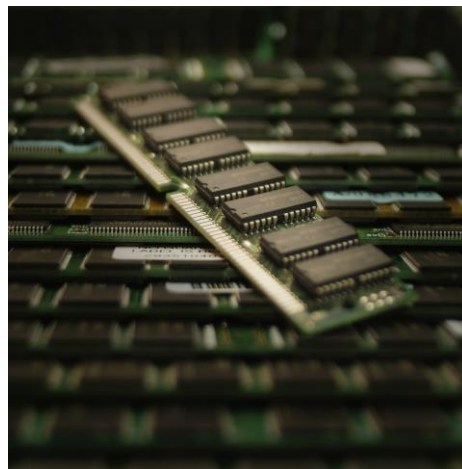


Python – Find all the Things

```
def parse_the_email(my_email):  
    msg = email.message_from_string(my_email)  
    for part in msg.walk():  
        awesome_sauce.append(part.get_payload(decode=True))
```

Python – Find all the Things

- While it's still in memory, let's grab metadata, too



Python – Metadata all the Things

- Since you can't find out who someone is from metadata...
- Shredder.py pulls different fields from the decoded text.
To, from, subject, etc

```
if part["to"] != None: my_to.append(part["to"])
if part["bcc"] != None: my_to.append(part["bcc"])
if part["cc"] != None: my_to.append(part["cc"])
if part["from"] != None: my_from.append(part["from"])
...
```

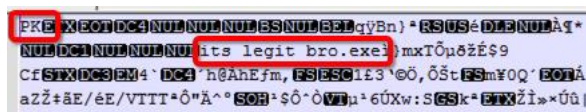
Python – Yara all the Things

- While we're at it...just Yara all the things

```
rules = yara.compile("yara_headers.yar", includes=True)
```

```
for each in awesome_sauce:
```

```
    matches = rules.match(data=each, callback=mycallback)
```



```
PKE XEOTDC4NUJNUJNUJ6SNUJEEqyBn)*BSUSéDIE NUJÄI*
NUJDC1NUJNUJNUJ its legit bro.exe)mxTÖu8žÊ$9
CfSTXDC3EM4`DC4`h@AhEfm,ESIESCif3`eÖ,ÖStESm¥OQ`EOTÄ
aZŽ+äE/éE/VITIT*Ö"Ä^°SOH`$Ö`ÖVIn`6ÜXw:S[ESk*ETXŽİ»»ÜÜ|
```


Hash all the Things

- Still looping awesome_sauce here

```
tMd5 = hashlib.md5(each).hexdigest()
tSha256 =
hashlib.sha256(each).hexdigest()
tSsdeep = ssdeep.hash(each)
```



Regex All The Things

- Why not iterate through awesome_sauce to pull out links?

```
def extract_my_links(data):
    links = re.findall('http[s]?://(?:[a-zA-Z]|[0-9]|$-_@.&+]|!*\\(|,)|(?
    %[0-9a-fA-F][0-9a-fA-F]))+', data)
```

Research all the Things

- <http://www.dshield.org/ipinfo.html?ip=<parsed domain/IP>>
- <https://www.robtex.com/dns/<parsed domain>.html>
- <http://network-tools.com/default.asp?prog=express&host=<parsed domain/IP>>
- Use VirusTotal's API

The Database

- Unique ID generated at runtime. Schema:
- ID + hashes
- ID + URL
- ID + Metadata (to, from, subject, etc)
- ID + Yara hits

Sqlite3 Schema

- `sqlite> .schema`
- `CREATE TABLE email_meta(key ..., date ..., e_to ..., e_from ..., e_subject ..., attachment ..., reporting_mechanism ..., email ...);`
- `CREATE TABLE hash_db(key ..., md5 ..., sha256 ..., ssdeep ...);`
- `CREATE TABLE urls(key ..., url ..., cleaned_url ...);`
- `CREATE TABLE yara_hits(key ..., hit ...);`

Why This Schema?

- All pieces go back to the unique key. Use cases:
- “Have I ever seen this MD5 sent via email”
- “Have I ever seen this URL in an email”
- “*Have I ever seen this TLD of a URL in my email*”
- “What email tripped Yara rule X”
- “Who else tripped that email”
- ...And the list goes on

Throughput

```
$ time cat order\ #247-0944829-3045913.eml | ./shredder.py no
```

```
real    0m0.096s
```

```
user    0m0.044s
```

```
sys      0m0.048s
```

Translation: less than a .1 second to read the email, scan with Yara, hash everything, metadata everything, be able to research everything, correlate everything...



Source: www.desdehollywood.com

Other Use Cases

- Scan all email from an enterprise (Postfix + catchall)
- Throw metadata into a database
- Pivot the data
- Ask questions about the data
- Love the data
- ...Pretty much an email IDS

Demo



To Recap, What do we Have?

- Parser for email
- Scanner of emails for custom signatures and content
- Hashing of everything (sha256, SSDeep, MD5)
- Regex of all email links
- All metadata about an email (sender, subject, etc)
- Sanitized links
- Ability to research the links

Got Code

 **DOWNLOAD**

[*https://github.com/x41x41x90/pm_shredder*](https://github.com/x41x41x90/pm_shredder)

Questions?

PHISHME



phishme.com/blog



phishme.com/resources/videos



phishme.com



Thank You!

Twitter: @iHeartMalware

Work: ronnie.tokazowski@phishme.com

Gmail: x41x41x90@gmail.com



MIRcon.
2014

29