"[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution."

Behavioral Matrix and Tool Analysis of Energetic Bear and GreyEnergy Actor

Kyle O'Meara
SEI CERT Coordination Center
June 2019

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Copyright 2019 Carnegie Mellon University. All Rights Reserved.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

CERT Coordination Center® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM19-0585

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Agenda

- **whoami**
- **Motivation**
- **Energy Sector Threats**
- **Behavioral Matrix Explanation**
- **Technical Tool Deep Dive**

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

whoami

- Software Engineering Institute (SEI) CERT Coordination Center (CERT/CC)
  - Threat Analysis Directive -> Malware Analysis Team -> Reverse Engineering (RE) Team
- Adjunct Faculty & Faculty Advisor at Carnegie Mellon University and Adjunct Faculty at Duquesne University
- Develop and run the BSidesPGH Capture the Flag (CTF)
- Past work includes:
  - FireEye - Sr. Security Consultant dedicated to the SharkSeer Program
  - Private Consulting Firm - Digital Forensics and Incident Response
  - National Security Agency (NSA) - Various roles as a Technical Analyst
- Larger speaking engagements: Blackhat USA & Europe Arsenal, DEF CON, ShmooCon, NatCSIRT, Countermeasure, BSidesPGH, FIRST TC, Senate Cyber Threat Intelligence Summit
- Publications can be found on the SEI Digital Library and CERT/CC GitHub

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

RE Team

The CERT/CC Reverse Engineering (RE) Team is committed being a world leader in advancing the art of malicious code analysis.

These include:

- Studying and analyzing the long or short term evolution of malware families
- Binary RE: Analysis and when appropriate YARA rule generation and/or the extraction of configuration data to include C2s, domains, username & password combinations, unique keys
- RE Tools: Create and maintain a suite of utilities to assist in and speed up the reverse engineering process
- RE Experience Beyond x86/x64: Experience in RE of other architectures and platforms such ARM, MIPS, PowerPC, and Android
- Hardware RE: firmware extraction and analysis

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Motivation

- Improve our corpus of knowledge on the RE Team
  - YARA rule and/or config dumper
- Identify gap areas of specific malware tools used by adversaries.
- How to do that?
  - Create a matrix that maps to internal and open source
  - Pick a piece of malware that has gaps in the matrix
  - YARA rule and/or config dumper

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Behavioral Matrix

- Fields
  - Malware Common Names
  - CERT/CC APIAnalyzer Behavior (if appropriate)
  - MITRE ATT&CK Techniques
  - CERT/CC Known YARA Rules
  - CERT/CC Config Dumper
  - OSINT YARA rules
- *This matrix will not only highlight internal gaps but also gaps that exist in the open source community*

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

CERT/CC APIAnalyzer

- APIAnalyzer is a tool for finding sequences of API calls with the specified data and control relationships.
  - This capability is intended to be used to detect common operating system interaction paradigms like opening a file, writing to it, and the closing it.
- Part of Pharos static binary analysis framework
  - https://github.com/cmu-sei/pharos

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Energy Sector

- This time the focus is on the energy sector
- How many threat actors have been known to target the energy sector?
    - 20
- How many known tools?
    - Too many to count

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Caveat

- The following names and tools might NOT be exhaustive
- “Best know for” context when appropriate

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Russia

Energetic Bear

- *Other known names:* Dragonfly, Crouching Yeti, Group 24, Iron Liberty
- *Tools:* Havex/Oldrea, Sysmain, Trojan Heriplor/API Hashing Tool, Karagany, LightsOut Exploit Kit

Sandworm

- *Other known names*: Quedagh Group, Black Energy, Iridium
- *Tools:* W32/Industroyer, CrashOverride
- *Best known for:* NotPetya

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

China

UPS

- *Other knows names*: Gothic Panda, APT 3, Group 6
- *Tools*: Shotput, Pirpi, PlugX/Sogu, Kaba, Cookie Cutter

Beijing Group

- *Other known names*: Sneaky Panda
- *Tools*: Hydraq, Elderwood Project
- *Best Known for*: Aurora Operation

C0d0so

- *Other known names*: APT 19, Sunshop Group
- *Tools*: Bergard Trojan, Derusbi, TXER

Hurricane Panda

- *Other known names*: APT 31, Black Vine
- *Tools*: China Chopper Webshell, PlugX, Mimikatz, Sakula

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

China (continue)

Poisonous Panda

- *Other known names & Tools*: N/A

Violin Panda

- *Other known names*: APT 8, APT 20
- *Tools*: Poison Ivy, CAKELOG, CANDYCLOG, COOKIECLOG, CETTRA
- *Best known for*: Nitro Attacks

Wet Panda

- *Other known names*: N/A
- *Tools*: PlugX

Emissary Panda

- *Other known names*: LuckyMouse, TG-3390, APT 27
- *Tools*: PlugX, China Chopper Webshell, HttpBrowser

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Iran

Cutting Kitten

- *Other known names*: TG-2889
- *Tools*: TinyZBot, PupyRAT
- *Best known for*: Operation Cleaver

Shamoon

- *Other known names*: Volatile Kitten
- *Tools*: Shamoon/Disttrack

APT 33

- *Other known names*:
- Tools: SHAPESHIFT, DROPSHOT, TURNEDUP

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Iran (continue)

Magic Hound

- *Other known names*: Timberworm, MAGNALLIUM, Elfin
- *Tools*: Shamoon, POWERTON, PUPYRAT, POSHC2, TURNEDUP, Quasar RAT
- *Best known for*: Stronedrill/Shamoon 2.0

Rocket Kitten

- *Other known names*: Flying Kitten, Saffron Rose, Ajax Security Team, Group 26
- *Tools*: GHOLE/Core Impact, CWoolger, .NETWoolger, Puppy RAT, MagicHound.Leash

OilRig

- *Other known names*: Cobalt Gypsy, Twisted Kitten, Crambus, HELIX KITTEN, APT 34
- *Tools*: Helminth, ISMDoor, Clayslide, QUADAGENT, customized Mimikatz, POWBAT, POWRUNER

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Hacktivists

- Ghost Jackal
  - *Other known names*: N/A
  - *Tools*: web application exploitation, website defacement
- Corsair Jackal
  - *Other known names*: N/A
  - *Tools*: web application exploitation, website defacement

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Non Country Specific

Sea Turtle

- *Other known names*: N/A
- *Tools*: Target known vuls, spear phishing
- *Best known for*: DNS hijacking

GreyEnergy Group

- *Other known names*: N/A
- *Tools*: Maldoc, FELIXROOT/GreyEnergy Mini (Dropper), GreyEnergy Backdoor
- *Best known for*: “A successor to BlackEnergy”, critical infrastructure attacks

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Behavioral Matrix Exemplars

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Non Country Specific – GreyEnergy Group

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

GreyEnergy Group

- Tool: FELIXROOT/GreyEnergy Mini (Dropper)

Malware Common Names	Actor Common Name	CERT APIAnalyzer Behavior	MITRE ATT&CK Techniques of Malware*	CERT Known YARA Rules	CERT Config Dumper	OSINT YARA Rules
FELIXROOT/GreyEnergy Mini (Dropper)	GreyEnergy	Informational: TerminateSelf Process Manipulation: CreateService, SpawnProcess	Code Signing, Command-Line Interface, Credential Dumping, File Deletion , Input Capture, Modify Existing Service, Modify Registry, Multi-hop Proxy, Obfuscated Files or Information, Process Injection, Remote File Copy, Rundll32, Software Packing, Standard Application Layer Protocol, Standard Cryptographic Protocol, System Service Discovery	No	Yes	Yes

* MITRE sees GreyEnergy as software and not a group

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Motivation

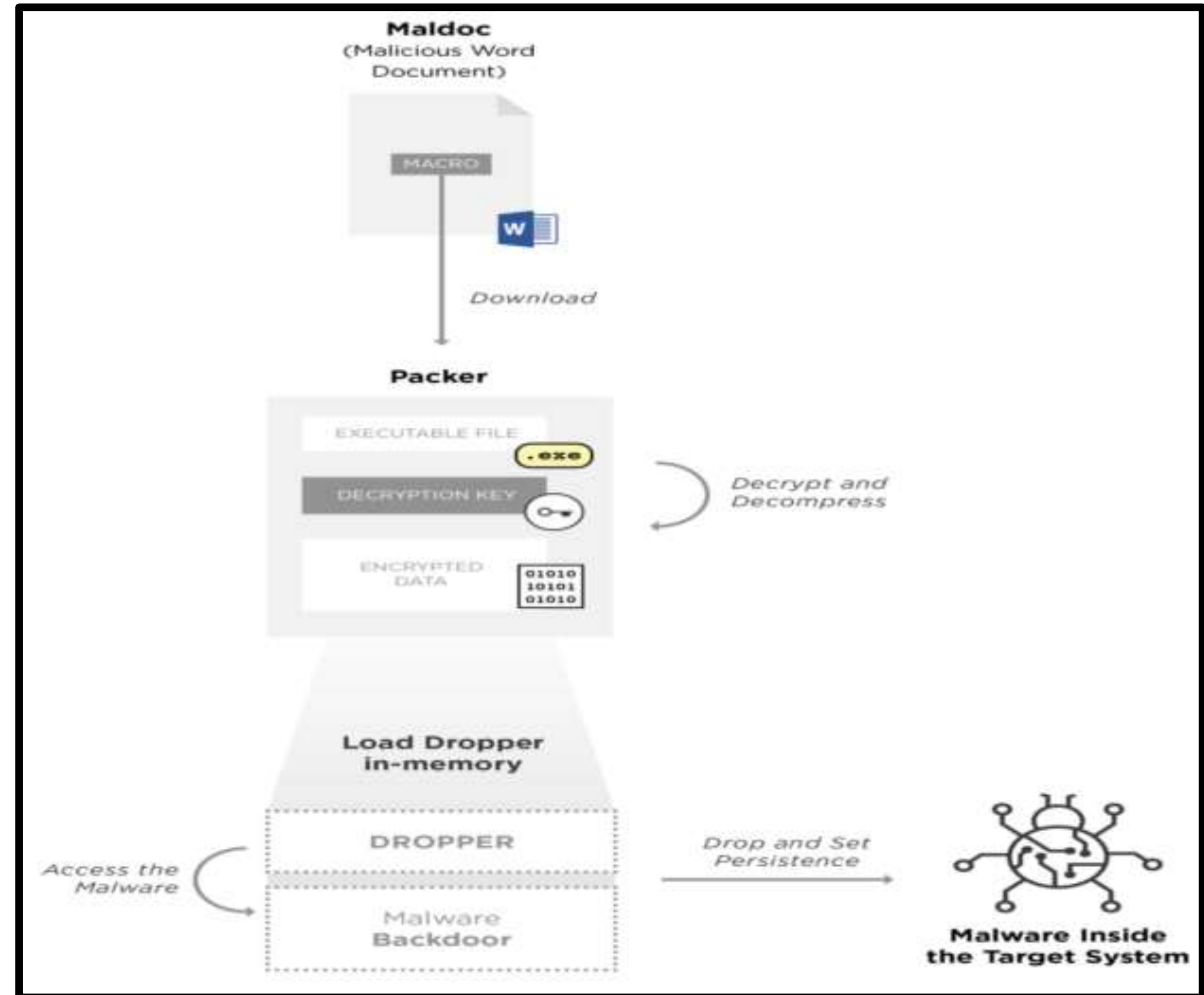
- Take a look at a group that specifically targeted the energy sector
- After reading a couple reports (ESET and Nozomi Networks) saw a challenge
- Challenge:
  - The reports were similar but didn't match – Nozomi Networks contained a specific workflow
    - Leverage the ESET report
  - Config dumper
    - Would only have results produced by config dumper if you execute the malware in a sandbox environment

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Attack Flow

1. Infection Vector: Maldoc
2. Stage 1: Packer
3. **Stage 2: Dropper**
4. Stage 3: Backdoor



Nozomi Networks Repot - GreyEnergy: Dissecting the Malware from Maldoc to Backdoor

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Dropper

- Leveraged the Nozomi Networks report (authored by Alessandro Di Pinto) analysis
- They wrote a YARA module and an unpacker tool
- The unpacker tool gives you the capability to get the dropper and the backdoor

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Dropper

- Focusing on the dropper
  - **Single execution of system**
  - **String obfuscation**
    - All the strings are obfuscated and stored in the .rdata section of the PE
      - This section is for read only data
      - This slows down analysis
    - The file obfuscates the strings with 8 byte XOR key
    - However, instead of utilizes only 4 of the 8 bytes of the key
    - *This made a nice situation for a config dumper*
  - **Memory wipe**
  - **Drop backdoor**

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Dropper

- **String obfuscation**
  - Strings are associated with what will be done with the dropped backdoor
    - File name of backdoor indicators
      - .db file extension which is a DLL
    - Persistence indicators
      - blank .lnk file pointing to rundll32.exe command
    - Execute backdoor
      - Executes the DLL with rundll32.exe
      - Cleans up traces of infection via shell: deletes and pings localhost
- Config dumper
  - Not everything in .rdata is a encoded string
  - Found 15 strings in the exemplars

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Analysis Process

- Analysis process – can be full automated
  1. Nozomi Networks GreyEnergy YARA module
  2. Nozomi Networks GreyEnergy Unpacker
     1. Dump both dropper and backdoor
  3. CERT/CC GreyEnergy Dropper Config Dumper
     1. Compare against previous TTPs

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Results

- Expanded corpus of analysis with the config dumper
- As a community urge for more detailed threat reporting
  - Meaning more clarification when defining malware types
  - Group vs. malware

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Russia – Energetic Bear

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Energetic Bear

- *Tool*: Trojan Heriplor/API Hashing Tool

Malware Common Names	Actor Common Name	CERT APIAnalyzer Behavior	MITRE ATT&CK Techniques of Malware	CERT Known YARA Rules	CERT Config Dumper	OSINT YARA Rules
Trojan Heriplor/API Hashing Tool	Energetic Bear, Dragonfly, Crouching Yeti, Group 24, IronLiberty	No results	Obfuscated Files or Information (Malware Attribute Enumeration and Characterization (MAEC) Project under: Anti Static Analysis (M9002)/Executable Code Obfuscation (E1027))	Yes	Yes	Yes

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Energetic Bear API Hashing Tool

- More of a story than the GreyEnergy Group analysis
- In the fall of 2018, the CERT/CC RE Team received a tip from a trusted source about a YARA rule (see below) they had been monitoring that alerted in VirusTotal
- YARA rule came from Department of Homeland Security (DHS) Alert TA17-293A
  - This document is associated with Russian activity.
- Current activity on year old signature == investigation

```
rule APT_malware_2
{
meta:
    description = "rule detects malware"
    author = "other"
strings:
    $api_hash = { 8A 08 84 C9 74 0D 80 C9 60 01 CB C1 E3 01 03 45 10 EB ED }
    $http_push = "X-mode: push" nocase
    $http_pop = "X-mode: pop" nocase
condition:
    any of them
}
```


© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Energetic Bear API Hashing Tool

- The YARA rule (from the DHS report) was allegedly associated with the Energetic Bear group
- Energetic Bear, named by CrowdStrike, conducts global intelligence operations primarily against the energy sector.
- They have been in operation since 2012
- This intrusion set is also named Dragonfly (Symantec), Crouching Yeti (Kaspersky), Group 24 (Cisco), and Iron Liberty (SecureWorks), among others

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Analysis Methodology

- Analyzed the YARA rule and initial exemplar
- Created a tightly scoped YARA rule to discover new exemplars
- Discovered API hashes found in new exemplars
- Questioned attribution
- Attribution
- Results

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Analyzed the YARA rule and Initial Exemplar

- Understand the YARA
  - What do I think the variables represent?
- Specifically, interested in the *$api_hash* variable
- The variables *$http_post* and *$http_push* appeared to be associated with HTTP header fields
- Focused my analysis on the *$api_hash* variable

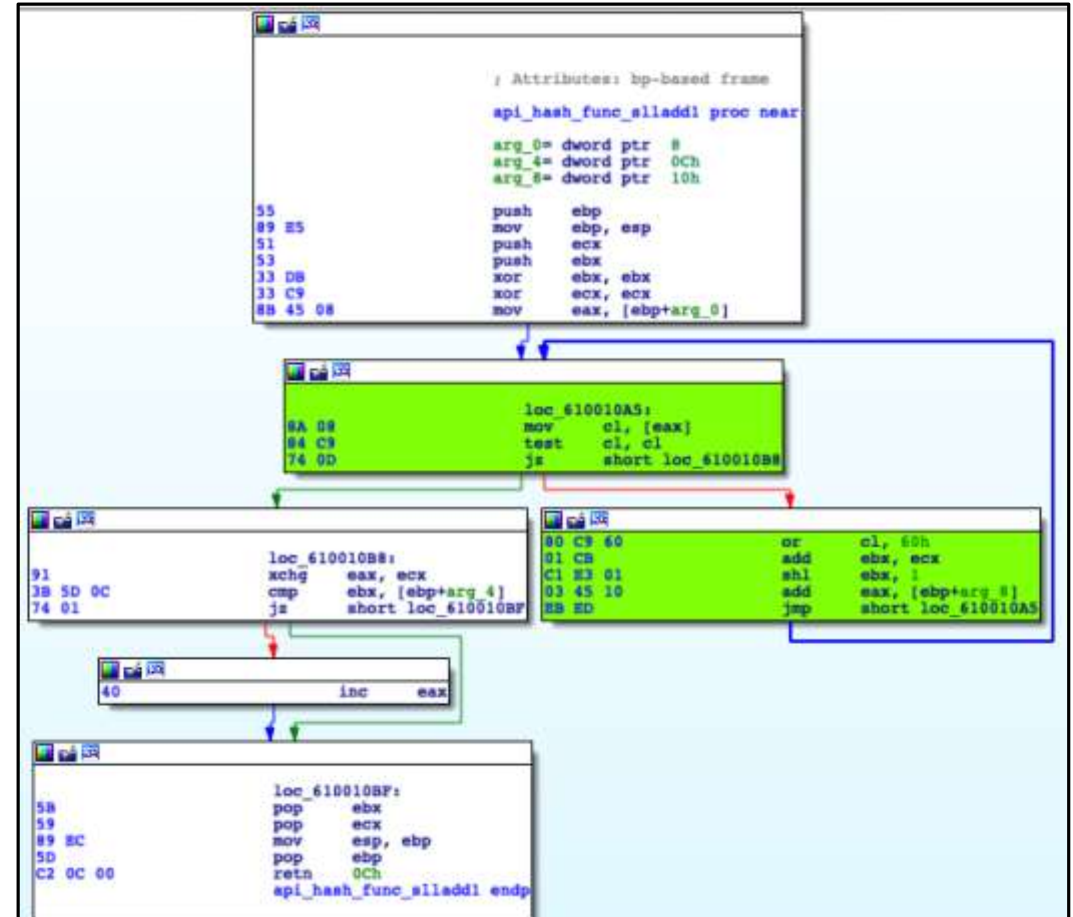
```
rule APT_malware_2
{
meta:
    description = "rule detects malware"
    author = "other"
strings:
    $api_hash = { 8A 08 84 C9 74 0D 80 C9 60 01 CB C1 E3 01 03 45 10 EB ED }
    $http_push = "X-mode: push" nocase
    $http_pop = "X-mode: pop" nocase
condition:
    any of them
}
```


© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Analyzed the YARA rule and Initial Exemplar

- The *$api_hash* variable was alerting on the routine, highlighted in green
- The key points to highlight are the looping routine that contained an
  - *or* of 0x60,
  - *shift logical left (shl) by 1*,
  - followed by an *add*
- Hashing routine…Windows API hashing…?
  - YARA variable name helped



© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Analyzed the YARA rule and Initial Exemplar

- Used the technical details of the previous API hashing routine and OSINT gathering, I found two items:
  - FireEye Flare IDA Plugins GitHub page that contained a plugin called *Shellcode Hashes*
  - FireEye blog post from 2012 titled “Using Precalculated String Hashes when Reverse Engineering Shellcode” which aided my understanding of API hashing
- In the FireEye Flare IDA plugin, *Shellcode Hashes* script, there was a similar routine that was mentioned on the last slide
  - *for loop* which contains an *or* of 0x60, followed by *add,* and a *shift left by 1*

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Things Known to this Point:

1. Exemplar uses API hashing technique that has been publicly seen before

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Analyzed the YARA rule and Initial Exemplar

- CERT/CC API hashing tool that creates a set of YARA signatures of API hashes for a given set of dynamic link library (DLL) files
  - This API hashing tool also contains a very similar function that matched the routine mentioned from the exemplar and the FireEye IDA plugin
- *Now I have 2 sources to back my findings based off of original YARA rule*
- Ran the CERT/CC API hashing tool on the exemplar for the specific *sll1Add* routine
  - Received an alert for kernel32.dll API hashes

Function	Byte Value (big endian)
LoadLibraryA	86 57 0D 00
VirtualAlloc	42 31 0E 00
VirtualProtect	3C D1 38 00

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Things Known to this Point:

1. Exemplar uses API hashing technique that has been publicly seen before
2. CERT has a tool that does API Hashing that includes this API hashing technique

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Analyzed the YARA rule and Initial Exemplar

- CERT/CC UberFLIRT ("kind of like FLIRT but Uber")
  - UberFLIRT calculates and stores PIC hashes of arbitrary functions, easily share information via a central database and has less false positives
  - Used with IDA
- Labeled the API hashing function shown in previous slides in IDA as *api_hash_func_slladd1* and saved it to the UberFLIRT database
  - This helps with analyzing potentially similar exemplars
- Examining the exemplar further:
  - Discovered 2 values that are passed as parameters to a function near entry point
  - The 2 values are 0x0038D13C and 0x000D4E88
    - The value 0x0038D13C is the hash of *VirtualProtect* which matches the values discovered by the CERT/CC API hashing tool on the previous slide
    - The other value, 0x000D4E88 is discussed next

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Analyzed the YARA rule and Initial Exemplar

- Examining this function further that took these 2 parameters:
  - Exemplar uses manual symbol loading techniques
    - Find symbols itself vs. using the Windows loader
  - Very similar to that of shellcode to find and load APIs
- This is where the API hash values come into play.
  - It's a way for the exemplar to obfuscate it's process
  - The 2 hashes we see first are to check the correct kernel32.dll (0x000D4E88) is being accessed and the second is for the first function, Virtual Protect (0x0038D13C), to utilize
  - Then the other functions, found as hashes in the exemplar, are checked and used
    - Will mention all the other functions later
- Labeled the function *manual_symbol_resolution* and saved it to the UberFLIRT database
  - This helps with analyzing potentially similar exemplars

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Things Known to this Point:

1. Exemplar uses API hashing technique that has been publicly seen before
2. CERT has a tool that does API Hashing that includes this technique
3. Labeled 2 functions using Uberflirt to help with future analysis
4. Exemplar using a technique typically used by shellcode
5. 2 initial hash values were identified - one for Virtual Protect, one for kernel32.dll name

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Created a Tightly Scoped YARA Rule to Discover New Exemplars

- Find new, but similar exemplars
- The following was my process to find additional exemplars:
  - Created API hash YARA rule to discover potential new exemplars
    - Used the CERT/CC MASS
  - Analyzed new exemplars for the presence of identified functions
    - *api_hash_func_slladd1* and *manual_symbol_resolution*
  - Created a tightly scoped YARA rule

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Created a Tightly Scoped YARA Rule to Discover New Exemplars

- CERT/CC Massive Analysis and Storage System (MASS)
  - In order to conduct research into executable code, we maintain a large archive of potentially malicious software artifacts
  - The archive consists of more than a billion artifacts collected from early 2005 until present day
  - The corpus of artifacts allows us, the analysts, to conduct research on how different malware families and threat actors have evolved over time

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Created a Tightly Scoped YARA Rule to Discover New Exemplars

- Using the MASS and this YARA rule to discover 36 new exemplars for a total of 37
- This YARA rule represents the *push* of the API hash value (0x0038D13C), *push* of the DLL hash value (0x000D4E88), and the *call* to *manual_symbol_resolution*
- I made an assumption that these 2 values were the 2 values always sent to the first function at the entry point of potential exemplars

```
rule api_hashes_2_call
{
  strings:
    (2019-02-22)
    $api_hashes_2_call = { 68 3C D1 38 00 68 88 4E 0D 00 E8 ?? ?? ?? ?? }
  condition:
    uint16(0) == 0x5a4d and $api_hashes_2_call
}
```


© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Things Known to this Point:

1. Exemplar uses API hashing technique that has been publicly seen before
2. CERT has a tool that does API Hashing that includes this technique
3. Labeled 2 functions using Uberflirt to help with future analysis
4. Exemplar using a technique typically used by shellcode
5. 2 initial hash values were identified - one for Virtual Protect, one for kernel32.dll name
6. Created specific YARA rule and identified 36 more exemplars

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Created a Tightly Scoped YARA Rule to Discover New Exemplars

- Wanted to make sure I really had new exemplars
- Refined the original YARA rule (from DHS report)
- Realized that some of the new exemplars did not alert with this refined YARA rule
- Analyzed this subset of new exemplars that did not hit
  - discovered two slight variations in the API hashing function
- The first was an addition of 1 extra byte
- The second dealt with 64-bit files

```
rule energetic_bear_api_hashing_tool {
meta:

    description = "Energetic Bear - API Hashing"
    assoc_report = "DHS Report TA17-293A"
    author = "CERT RE Team"
    version = “1”

strings:
    $api_hash_func = { 8A 08 84 C9 74 0D 80 C9 60 01 CB C1 E3 01 03 45 10 EB ED }
    $http_push = "X-mode: push" nocase
    $http_pop = "X-mode: pop" nocase

condition:
  $api_hash_func and (uint16(0) == 0x5a4d or $http_push or $http_pop)
```


© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Created a Tightly Scoped YARA Rule to Discover New Exemplars

- Refined the YARA rule further to incorporate these two variations, for a total of 3 variations

```
rule energetic_bear_api_hashing_tool {
meta:
    description = "Energetic Bear API Hashing Tool"
    assoc_report = "DHS Report TA17-293A"
    author = "CERT RE Team"
    version = “2”

strings:
    $api_hash_func_v1 = { 8A 08 84 C9 74 ?? 80 C9 60 01 CB C1 E3 01 03 45 10 EB ED }
    $api_hash_func_v2 = { 8A 08 84 C9 74 ?? 80 C9 60 01 CB C1 E3 01 03 44 24 14 EB EC }
    $api_hash_func_x64 = { 8A 08 84 C9 74 ?? 80 C9 60 48 01 CB 48 C1 E3 01 48 03 45 20 EB EA }

    $http_push = "X-mode: push" nocase
    $http_pop = "X-mode: pop" nocase

condition:
    $api_hash_func_v1 or $api_hash_func_v2 or $api_hash_func_x64 and (uint16(0) == 0x5a4d or $http_push or $http_pop)
}
```


© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Things Known to this Point:

1. Exemplar uses API hashing technique that has been publicly seen before
2. CERT has a tool that does API Hashing that includes this technique
3. Labeled 2 functions using Uberflirt to help with future analysis
4. Exemplar using a technique typically used by shellcode
5. 2 initial hash values were identified - one for Virtual Protect, one for kernel32.dll name
6. Created specific YARA rule and identified 36 more exemplars
7. Refined original YARA rule to cover 3 variations

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Discovered API Hashes Found in New Exemplars

- Identified *sll1Add* routine API hash values found in all of the 37 exemplars
- All exemplars had API hash values for functions from kernel32.dll
- Most of the exemplars had *sll1Add* routine API hash values for functions from ws2_32.dll
- There were a few outliers that had *sll1Add* routine API hash values for functions form wininet.dll

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Discovered API Hashes Found in New Exemplars

- Use of functions from ws2_32.dll and wininet.dll demonstrates that these exemplars have potential network communications
  - The use of two different DLLs for network communications draws conclusions to the existence of at least 2 different versions of the API hashing tool
- Further analyzed the exemplars to identify any network-based IOCs
- 33 of 37 exemplars, identified 29 unique IP address and port combinations

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Discovered API Hashes Found in New Exemplars

- 4 of 37 the exemplars had structure outbound POST request.
- For 2 of these 4, I captured the requests in a pcap using FakeNet

```
POST / HTTP/1.1
X-mode: pop
X-id: 0x00000000,0x5547a48a
User-Agent: Mozilla
Host: 187.234.55.76:8080
Content-Length: 0
Connection: Keep-Alive
Cache-Control: no-cache
```

- Inferred the outbound POST request structure from strings for the remaining 2

```
X-mode: push\r\nX-type: more\r\nX-id:
0x00000000,0x523fe61c\r\n
X-mode: push\r\nX-type: last\r\nX-id:
0x00000000,0x523fe61c\r\n
X-mode: pop\r\n\r\nX-id: 0x00000000,0x523fe61c\r\n
Mozilla
POST
```

- You can build Snort signatures for this traffic

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Things Known to this Point:

1. Exemplar uses API hashing technique that has been publicly seen before
2. CERT has a tool that does API Hashing that includes this technique
3. Labeled 2 functions using Uberflirt to help with future analysis
4. Exemplar using a technique typically used by shellcode
5. 2 initial hash values were identified - one for Virtual Protect, one for kernel32.dll name
6. Created specific YARA rule and identified 36 more exemplars
7. Refined original YARA rule to cover 3 variations
8. Identified all API hash values present in all of the exemplars
9. Identified 2 additional versions based on network comms
10. Identified additional IOCs

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Questioned Attribution

- Tried to identify other public reporting or research related to this Energetic Bear API hashing tool
- I did not find any reporting or research

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Questioned Attribution

- Since there was a link to Energetic Bear, I thought it could be a remote access trojan (RAT) like Havex, which is also attributed to this particular group
- Decided to explore the potential RAT association
- Discovered research by Veronica Valeros on "A Study of RATs: Third Timeline Iteration"
  - Contacted her directly and ask if she recalled any of the RATs use an API hashing technique
  - She could not recall, but mentioned that it could have been missed because she was not explicitly looking for this technique.
- Used her research to attempt to identify more RATs that use an API hashing technique
- <u>I have not found any that use this technique</u>

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Attribution

- The power of open source sharing has been positive
- It was brought to my attention (thanks to Matt Brooks from Citizen Lab) that this API hashing tool is related to Trojan.Heriplor from Symantec’s Dragonfly report
- The hash in Symantec's report is, in fact, one of the exemplars
  - Symantec provided this hash in the form of a picture, and I must have fat-fingered the hash when transcribing it
  - However, this specific API hashing technique isn't mentioned in their report
- Symantec's Trojan.Heriplor analysis attributes my analysis of this API hashing tool to Energetic Bear
- More importantly, this linkage also shows that this tool is still actively used
  - Most recent hit on my YARA in VirusTotal was June 3rd

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Results

- I hope by publicly discussing that analysis that I can encourage information sharing and allow us, as a community, to urge for more detailed threat reporting
  - i.e. no pictures of hashes ¯_(ツ)_/¯
- Have spoken to the MITRE ATT&CK team and the Malware Attribute Enumeration and Characterization (MAEC™) team
  - API Hashing has been added to their Malware Behavior Catalog Matrix
  - You can find the API Hashing listed as an Executable Code Obfuscation method under the Anti-Static Analysis behavior
- Complete analysis on blog which includes YARA rule and IOCs
- YARA rule can also been found in Florian Roth's Signature Base on GitHub
- My analysis and attribution alignment makes this report (blog) the **first** publicly documented report of API hashing technique being used by a nation state actor

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Contact Information

Presenter

Kyle O’Meara

Sr. Member of the Technical Staff

Email: komeara@cert.org

Twitter: @cool_breeze26

Contact me directly for GreyEnergy Dropper Config Dumper

Blog post on SEI website titled “*API Hashing Tool, Imagine That*”
https://insights.sei.cmu.edu/cert/2019/03/api-hashing-tool-imagine-that.html

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

References

- Nozomi Networks - GreyEnergy: Dissecting the Malware from Maldoc to Backdoor
- Nozomi Networks GitHub – GreyEnergy Unpacker
- ESET – GreyEnergy: A successor to BlackEnergy
- FireEye - Microsoft Office Vulnerabilities Used to Distribute FELIXROOT Backdoor in Recent Campaign
- Department of Homeland Security (DHS) Alert (TA17-293A)
- CrowdStrike Global Threat Report: 2013 Year in Review
- APT Groups and Operations
- FireEye Flare IDA Plugins Github
- Using Precalculated String Hashes when Reverse Engineering Shellcode
- Sikorski, Michael and Honig, Andrew; Practical Malware Analysis; 2012
- FakeNet
- A Study of RATs: Third Timeline Iteration by Veronica Valeros
- Malware Attribute Enumeration and Characterization (MAEC™)
- MITRE's Malware Behavior Catalog
- Execution Code Obfuscation Method from the Malware Behavior Catalog
- Signature Base by Florian Roth

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.