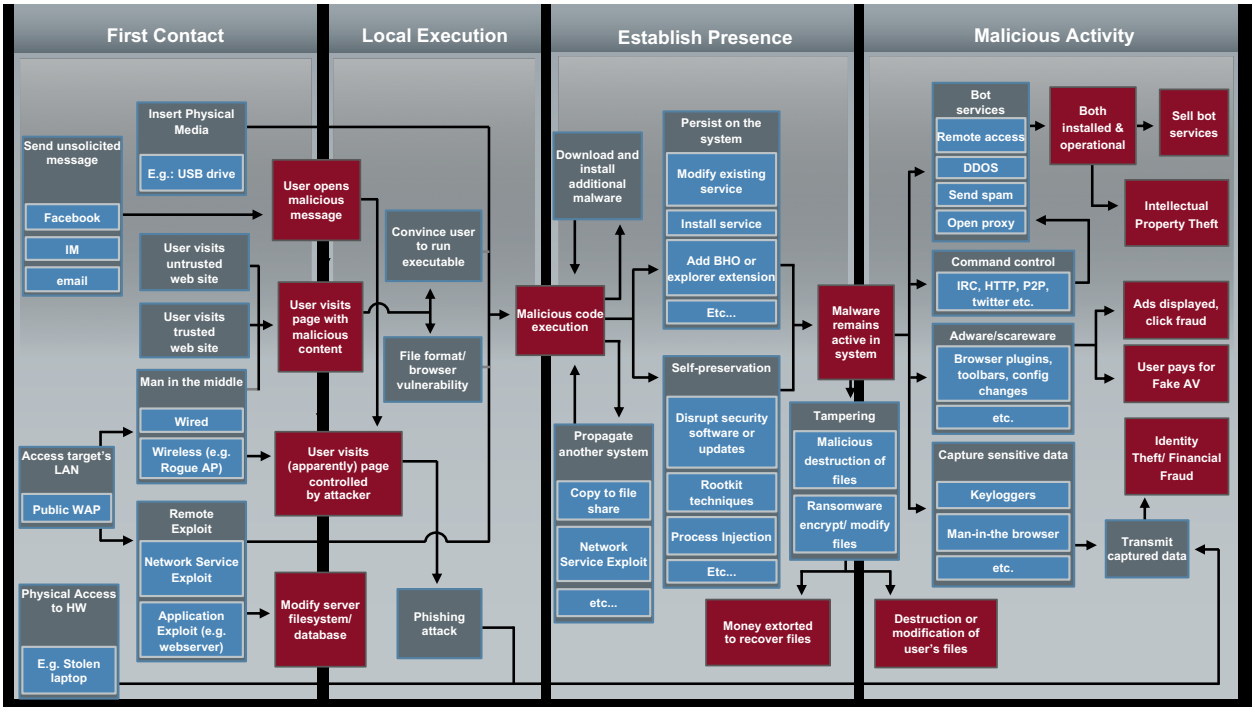
Johnny Zhong
July 14, 2019
CS373 – Defense Against The Dark Arts
Week 3
Malware Defense

Process

Attack Vector Graph – how to approach problems followed by transition on how to approach issue from defensive perspective



4 key tenets:

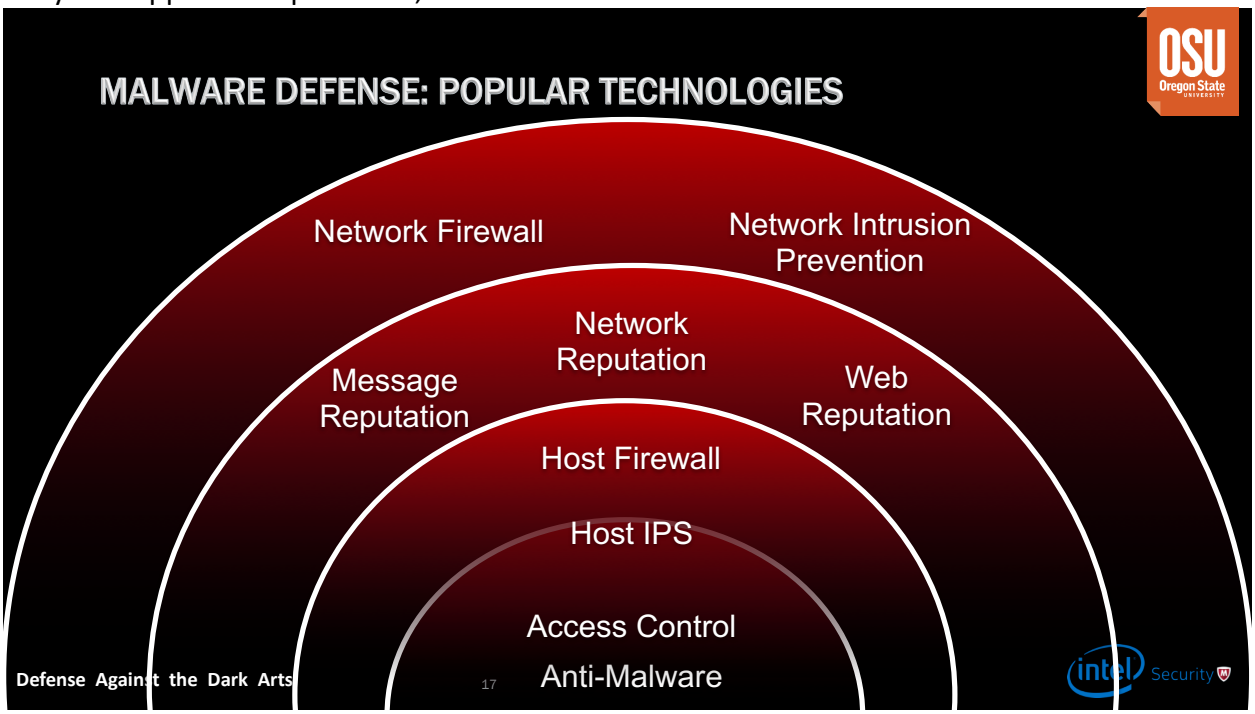
- First contact
    - Method to reach victims
    - Performed after attack has been developed
    - Some examples:
        - Email
        - Social Engineering
        - Curiosity
        - Banner Ads/Malvertising
        - Poisoned Search Results
        - Watering Hole
- Local Execution
    - Autorun (exploiting a feature)
    - Exploitation (ex: via PDFs)

    - Social Engineering
- Establish Presence:
    - Blend in or Hide –
        - Appear legitimate
            - use os-like filenames
            - change timestamp of modified
            - signed
        - hide
            - bootkit – install file and pass permissions
            - rootkit – change the responses the OS provides
    - Persist – be back after a reboot
        - System start
        - Windows Start
        - Application Start
    - Get around AV
        - Hide in registry
- Malicious Activity
    - Methods:
        - Get docs, passwords, email, processes
        - Hook into browser, keyloggers, screen scrape
        - Parse hidden info
        - Log activity
    - Example: memory scrape in PoS devices (Home Depot, Target)

How do we block these threats?

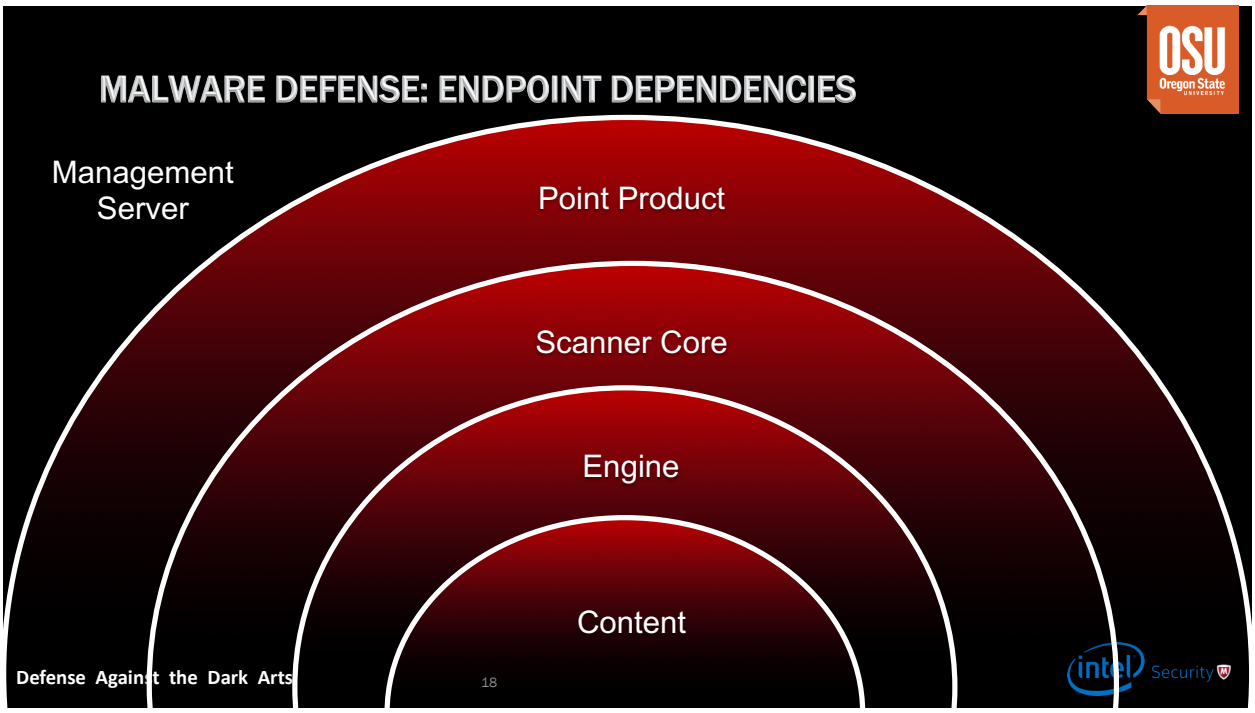
- First Contact:
    - Education
    - Blacklist (web reputation)
    - Script blockers
    - Physically block ports
- Local Execution
    - 2FA, RSA tokens
    - Quarantine?
    - UAC
    - Host Intrusion Prevention
- Establish Presence
    - Disable detection updates
    - Disable security updates
    - Whitelist self
- Malicious Activity
    - Determine what types of information are going out
    - Ransomware

A layered approach is preferred, which makes it more difficult for attackers.



Notes:

- Policies may be organization based
- Known threats are blocked, new threats may come from an update to policy
- Products must deliver enough functionality that people can work securely



Notes:

- ePO – ePolicy Administrator – can define groups and change policies based on groups
- Works next to point product (virus scan enterprise, host intrusion prevention)
- Scanner Core – a product specialized product – ex: a product that is used to scan rootkits
- Engine: consumes content scanner core creates
- Cloud:
    - Decomposition: how do you break open an archive to look at its content?
    - heuristics

YARA – open source language for scanning files and memory

- uses pattern matching
- can be strings, bytes patterns
    - strings expressed in quotes
    - byte patterns expressed in {}
- can use conditionals

YARA code snippet (Lab1):

```
rule Sytro
{
    strings:
     $str40="Jenna Jam"
     $str27="AikaQ"
    condition:
     all of them
```

```
}
```

The above code indicates that YARA is performing a search for the strings specified in the samples provided. The goal is to determine commonality among the samples that are specific enough to the samples such that we minimize false positives while being encompassing enough to include derivations of the samples, should the samples change.

YARA Code Snippet (Lab2):

```
rule CVE-2008-2551
{
 strings:
     $str1="DownloaderActiveX"
     $str2={63 31 62 37 65 35 33 32 [1-3] 33 65 63 62 [1-3] 34
65 39 65 [1-3] 62 62 33 61 [1-3] 32 39 35 31 66 66 65 36 37 63
36 31}
    condition:
     all of them
}
```

In this code snippet, we use the curly brackets to specify that the sequence we're looking for is a byte sequence, in this case, a hexadecimal number. We're also using a wildcard here to indicate that any sequence of characters of length 1 to 3 are acceptable at the given points on the hexadecimal byte sequence.

YARA Code Snippet (Lab3):

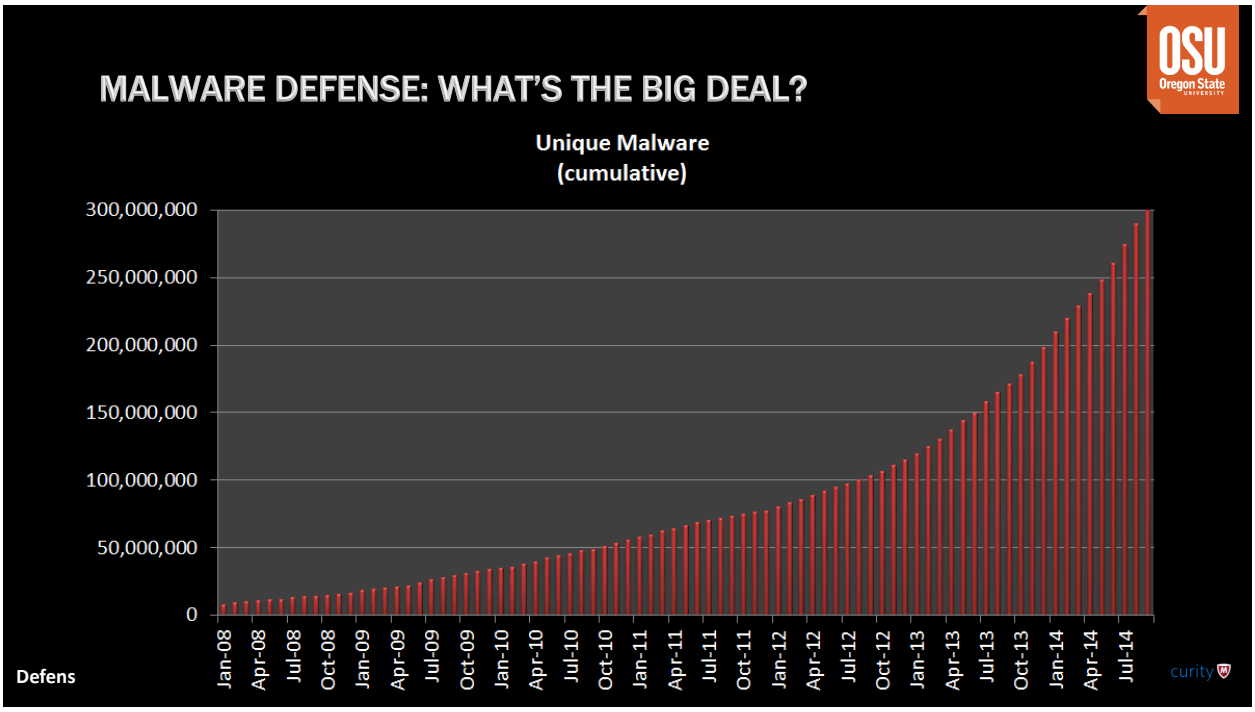
```
rule Tugu
{
 strings:
     $str1="existeClavePropiaAVG"
     $str2={15 E4 96 38 3F 5A 03 96 A7 AD 86 D8 58 50 D5 BB}
     $str3="TuguuAdw"
    condition:
     $str1 or $str2 or $str3
}
```

In this code snippet, we see the use of the "or" condition. This YARA code will return files that match any or all of the strings/byte sequences specified. The hexadecimal code here refers to a certificate.

Signkeys – meant to establish trust, but can be purchased. It seems the strict definition of a signkey is to simply indicate that the file was not tampered with between the distributor and the receipt of the file.

YARA notes:

- detection ratio: number of lines of code to how many files were detected. Sort of a measure of effort vs reward. Ideally, minimal effort nets a high amount of files found.
- String meaning: Reading the files may also provide a string with significance, perhaps a vendor or a company or a distributor has their name in the file. This may lead to the ability to a sort of signature in itself if files by the adware vendor contain a name.
- Largely performed by automation, but the best signatures are created by humans
- Works with memory as well as files



Automation of binary detection due to the huge number of unique binaries out in the world.
Causes of the increase:

- Detection of malware is working
- Polymorphism
- Binary possibly dependent on IP

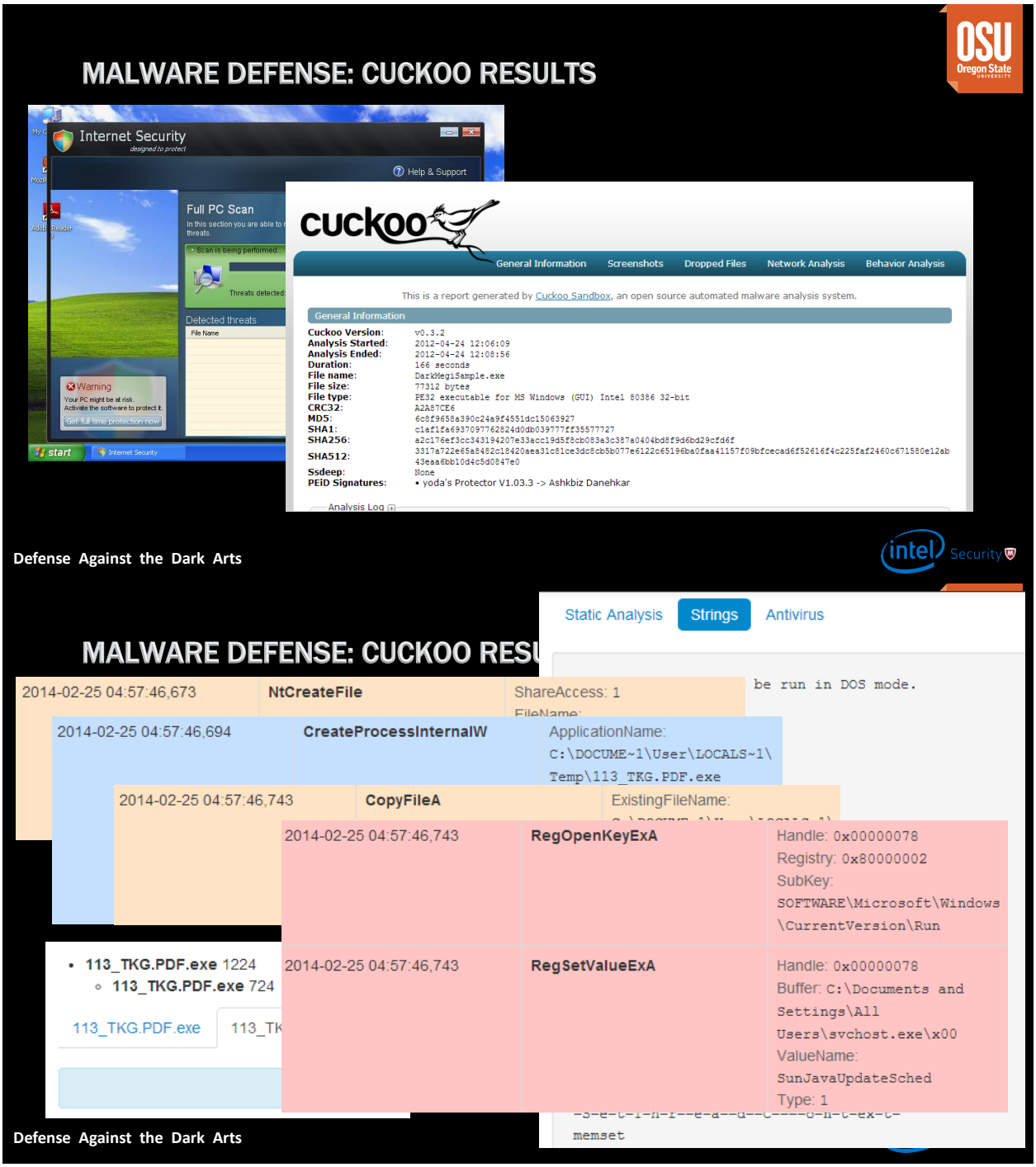
Automation/Machine Learning Notes:

- Fast, can work at scale
- Working time
- Initial methods included decision trees/forests
- Consistency
- Maybe noise

Cuckoo Automated Analysis

- Used by researchers
- Runs on a host – acts as a controller
- Allows user to understand what malware does in an isolated environment

- Sample behavior is provided back to the host
- It'll take screenshots, log network traffic, take memory dumps. Lots of automated features to allow for later analysis by researchers
- Randomly named dll to prevent malware from detecting it's in a VM
- Likely used by Malwr.com



Notes:

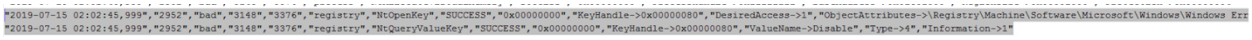
- Sequence of events provided to indicate malware behavior.
- Cuckoo handles many different file formats
- May require VMs to be loaded with a specific version of the application in question (adobe, office)
- Makes mouse jump around to fool malware

Blog Post:

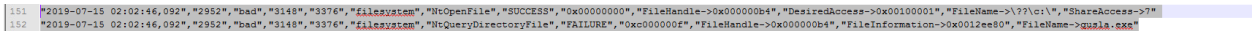
Johnny Zhong
Analysis Completed on July 14 2019 at 17:08 (-07)

A malware sample was provided and its behavior was analyzed. The MD5 Hash for the sample is 00670F2B9631D0F97C7CFC6C764DD9D9. Cuckoo was used to perform automated analysis of its behavior and the following details were found.

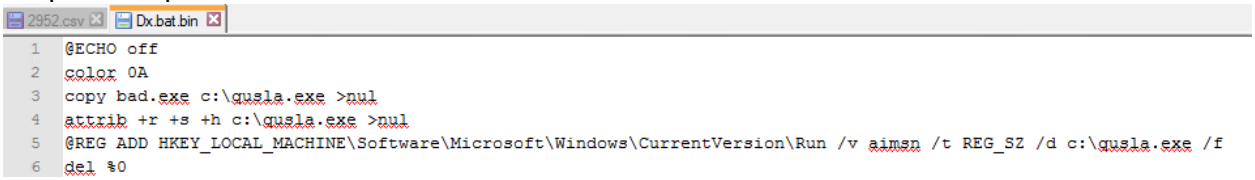
When run, the sample first disables the Windows Error Reporting, so that the user is not alerted to changes made or its presence.



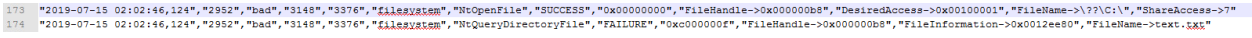
Afterwards, a file is created called “qusla.exe”, which is to reside in the C:\ directory.



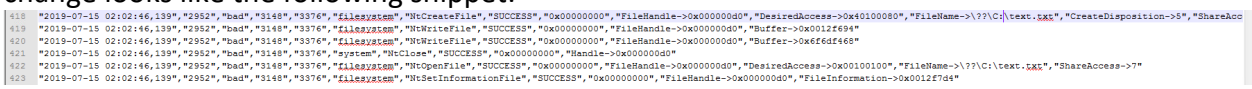
This is performed by a script created and deleted by the malware called “Dx.bat”. “Dx.bat” also creates an entry in the registry to run “qusla.exe” on boot of the machine. The contents of the script are in plaintext and are as follows:



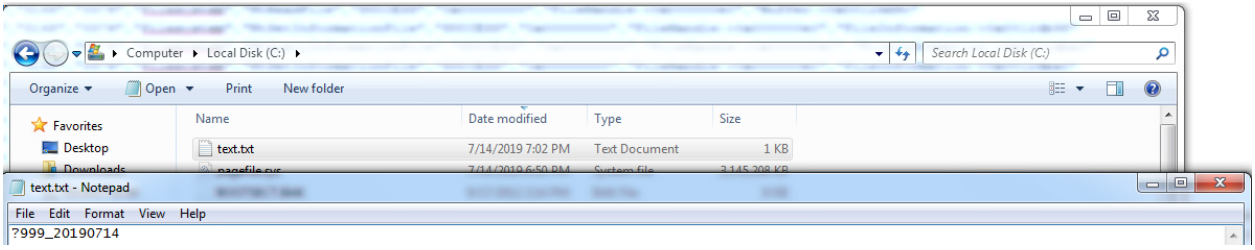
A file, “C:\text.txt”, is also created by the sample. As seen in this log file:



The contents appear to be a recording of the date that the malware was run. The log for this change looks like the following snippet:



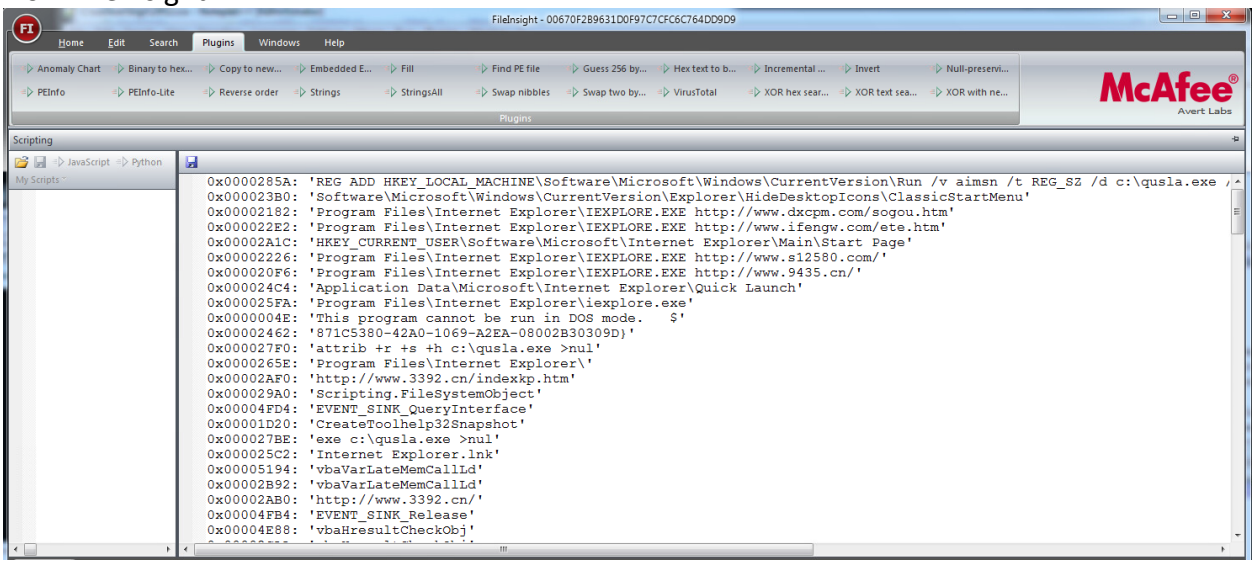
For those curious if they have been affected by 00670F2B9631D0F97C7CFC6C764DD9D9, a simple method to determine this is to look for this file in the C:\ directory. It should look like this:



Also, an affected user’s Internet Explorer homepage would be set to __. This was observed in the log file below, with an example from Internet Explorer:

The date when the user was affected is provided as an endpoint for the new homepage.

The behavior was corroborated by contents of the sample, extracted using the “strings” plugin from FileInsight.



These strings served as inspiration for the following yara search script that can be used for detecting the malware.

```
rule 00670F2B9631D0F97C7CFC6C764DD9D9{
     strings:
          $www9435cn={68 00 74 00 74 00 70 00 3A 00 2F 00 2F 00
77 00 77 00 77 00 2E 00 39 00 34 00 33 00 35 00 2E 00 63 00 6E
00}
          $www3392cn={68 00 74 00 74 00 70 00 3A 00 2F 00 2F 00
77 00 77 00 77 00 2E 00 33 00 33 00 39 00 32 00 2E 00 63 00 6E
00}
          $quslaexe={71 00 75 00 73 00 6C 00 61 00 2E 00 65 00
78 00 65 00}

     condition:
          ($www9435cn and $www3392cn) or $quslaexe
}
```