CS7038 - Malware Analysis - Wk04.1
Malware Taxonomy and Terminology

Coleman Kane
kaneca@mail.uc.edu

January 30, 2018

Why Classification is Important

Malware classification helps us
in multiple ways. Here are a couple key ways:

- Common language
- Risk scoring

In the same way that a birdwatcher may want to communicate conclusions about ecology from analysis of bird species observed, or a geologist needs to precisely describe landforms to communicate expectations, the malware analyst informs their team about cyber threats using a precise language.

Malware Taxonomy
is the science of classification of malware, and is a big part of malware analysis.

Source:

`http://www.redbubble.com/people/misskari`

Types of Classification

There are multiple types of classification, and each serve various end goals.

Functional Classification
This approach tends to use the features implemented by malware to classify the tool. You likely have seen these when malware is described as *backdoor*, *ransomware*, and similar.

Familial, Lineage Classification
This approach tends to focus on the authorship and the lineage of a malware tool. It classifies malware according to *families*, *authorship*, and similar attributes. It focuses on evolution of certain tools, and the expectation that common authorship can help inform incident response.

Behavioral
This approach uses behaviors exhibited by tools to categorize them. Similar to *funtional classification*, but different in that it focuses more on the exhibited behaviors, rather than the features a tool provides to an intruder.

Functional Classification

We will define terminology for the first category now.

The following are common terms classifying malware by function. If you have *Practical Malware Analysis, Sokorski & Honig*, many of these are covered on pages 3-4.

- Trojan Horse
- Backdoor
- Remote Access Tool
- Downloader
- Dropper
- Botnet
- Monitor
- Mailer
- Scareware
- Ransomware
- Information Stealer
- Rootkit
- Worm
- Virus

It is not necessary for every malware tool to fall squarely in one of the above buckets. It is far more common for malware to employ more than one of the above features.

Trojan Horses

Sometimes just abbreviated *Trojan*, this is frequently used to classify malware sample that employ some mechanism to conceal their true identity. In our example, the `evil.pdf` exhibited this behavior by concealing a PDF exploit.

Other examples might be:

- Malware embedded within a bundle that pretends to be a network driver you have downloaded off of the Internet.
- A "demo" music player that contains embedded backdoor code when you run it.

Backdoors & Remote Access Tools

Backdoor functionality is traditionally when malware provides an intruder some level of interactive 1-to-1 access to a compromised system. Though the mechanism differs wildly from tool to tool, it is commonly provided through a network connection between the adversary and the target's host.

A *Remote Access Tool* is frequently used to describe a backdoor that has a high level of interactive functionality, and provides expansive access to the target's host. In some cases, the level of access provided is more powerful (though not necessarily more permissive) than the normal end-user inderface of the targeted system.

The EXE that we extracted from inside `evil.pdf` is a relatively simple *backdoor*, merely providing interactive interaction with `cmd.exe`.

Downloader

An attack targeting a system frequently employs more than a single malware tool throughout the length of the attack. *Downloader* functionality provides the tool with a specific feature to download other tools onto the machine running the malware.

Frequently, this enables an adversary to deliver a relatively lightweight tool via a delivery vector, such as spear-phishing or a malicious Java applet buried in an advertisement. This tool may then be pre-configured to pull down more feature-full malware files from other locations on the Internet - possibly even using some attributes of the compromised host to guide where to retrieve them from.

Breaking attacks up into separate tools can help to reduce detectability of malware, as well as facilitate surgical retooling.

Dropper

In some cases, an attack may need to write malware to disk prior to executing it. This is especially common with Trojan Horse documents (such as our `evil.pdf`) that contain embedded, encoded programs as backdoors. The code typically must be in machine-native form prior to execution, and thus must first be extracted and stored in an OS-compatible container (the EXE file).

When malware is written to disk, and then executed or staged for execution, it is said to implement *dropper* functionality. The document `evil.pdf`, in addition to being considered a *trojan* also would be classified as a *dropper*, as it “drops” an EXE program to disk to execute it, in order to set up the remote channel to our Metasploit instance.

Botnet

Malware employing *botnet* functionality is very similar to *backdoor* functionlity, with one important difference: Frequently this describes malware that is controlled remotely in a 1-to-many relationship. Rather than using the compromise to access individual systems, the attacker would be attempting to build a network of compromised hosts that can be controlled in parallel to carry out bulk actions/objectives. Frequently, the compromised hosts are called the "nodes" of the botnet, which is controlled by an "operator".

A very common and high-visibility of this mechanism is employed in many Distributed Denial of Service (DDoS) tools, where an adversary uses the collective bandwidth and geographic distribution of many controlled nodes to saturate the resources of a target.

Monitor

These days, we frequently have systems that come with various recording devices, such as cameras and microphones. Malware employs *monitor* functionality when it implements some mechanism for recording user activity and/or environment (*monitoring* the target) and delivering the recorded data back to the adversary for review or other use.

Some monitor sources available on a target computer:

- Webcam
- Microphone
- Desktop recording
- Password prompts
- Common data folders (like "My Documents")
- Keyboard (keylogging)
- Web browser traffic
- Network traffic

Mailer

Another functionality that is very frequently implemented is *mailer*, sometimes called *spammer* or *spambot*.

Due to the lucrative nature of delivering bulk advertising and even bulk malware phishing, one common use for compromised systems is to use them as sources of new unsolicited emails.

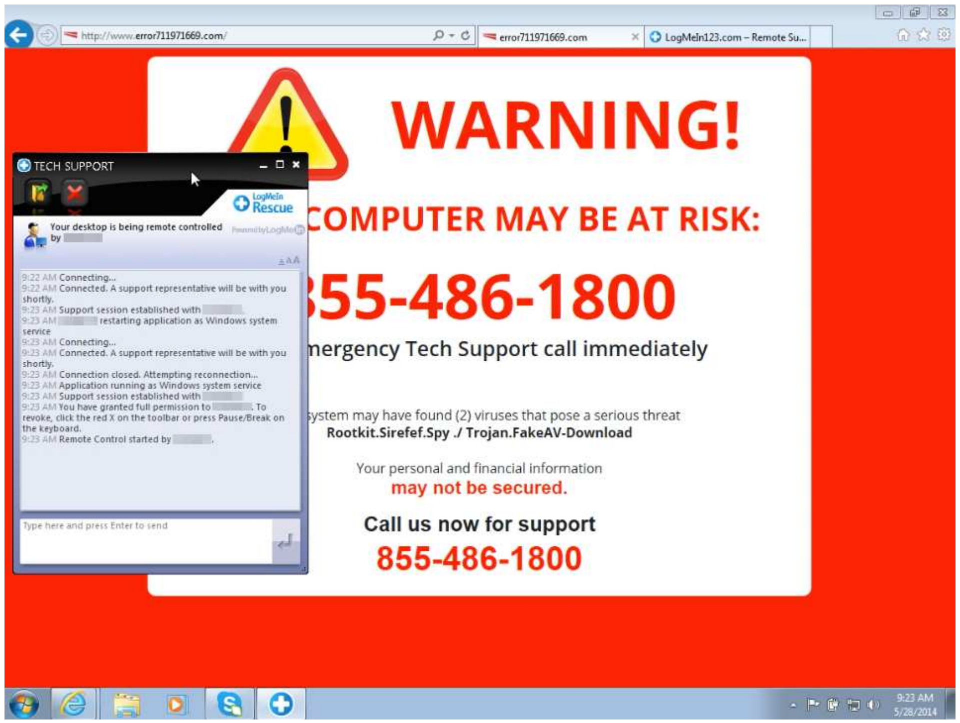
This functionality can range from: simply delivering email directly via SMTP from compromised host(s) to targets, up to mroe complicated mechanisms that enable an attacker to distribute email using the web-based mail account owned by the target (and thus, making it look like the victim is sending emails from their gmail account, for instance).

Scareware/Adware

Frequently a feature employed to achieve a *social engineering* outcome, this type of feature entices or frequently scares the user into taking a specific action to install malware on their system, pay money to remove nuisance pop-ups, or other response that the adversary desires.

A common occurrence of this employs “Fake Anti-Virus”. Once installed onto a host, it will annoy the system owner until a fee is paid to remove it.

Also, a common variant is adware, which typically applies where the software may not be an illegal enterprise, but is annoying and frivolous nonetheless.



WARNING!

COMPUTER MAY BE AT RISK:

855-486-1800

mergency Tech Support call immediately

ystem may have found (2) viruses that pose a serious threat
Rootkit.Sirefef.Spy ./ Trojan.FakeAV-Download

Your personal and financial information
may not be secured.

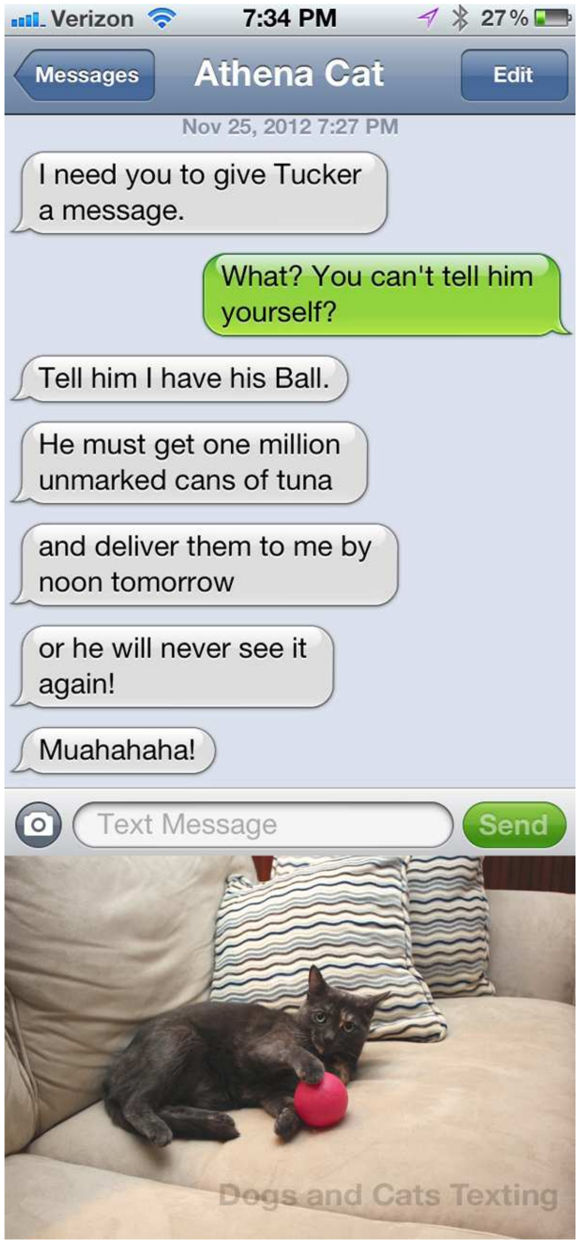
Call us now for support
855-486-1800

Your desktop is being remote controlled by [redacted]

```
9:22 AM Connecting...
9:22 AM Connected. A support representative will be with you shortly.
9:23 AM Support session established with [redacted].
9:23 AM [redacted] restarting application as Windows system service
9:23 AM Connecting...
9:23 AM Connected. A support representative will be with you shortly.
9:23 AM Connection closed. Attempting reconnection...
9:23 AM Application running as Windows system service
9:23 AM Support session established with [redacted]
9:23 AM You have granted full permission to [redacted]. To revoke, click the red X on the toolbar or press Pause/Break on the keyboard.
9:23 AM Remote Control started by [redacted].
```

Type here and press Enter to send

Ransomware


Verizon 7:34 PM 27%

Messages Athena Cat Edit

Nov 25, 2012 7:27 PM

I need you to give Tucker a message.

What? You can't tell him yourself?

Tell him I have his Ball.

He must get one million unmarked cans of tuna

and deliver them to me by noon tomorrow

or he will never see it again!

Muahahaha!

Ransomware functionality typically attempts to achieve similar end-goals as *scareware*. Namely, it intends to social engineer the target into paying a fee or some other activity. In this case, however, *ransomware* typically encrypts files on disk, or transfers them to a remote server and deletes the local copy.

In order for the user to recover their files, they must pay for their safe return, either via decryption or via retransmission.

Many contemporary approaches leverage strong public-key cryptography, where the malware only contains the public key - thus making it practically impossible to recover the files without paying.

Information Stealer

Malware eploying *information stealer* functionality is typically engineered to make the process of stealing personal, priovate, and/or confidential information from a target more efficient. Frequently, this is achieved through automation.

Upon connecting, malware may immediately proceed to copy data matching a hard-coded or network-provided *collection plan* to a remote system.

Common approaches include:

- Contacts list theft
- Browser cookies/history/saved credentials
- Documents theft
- OS passwords and other keystore data

Rootkit

Malware providing *rootkit* functionality utilizes Operating System interfaces to conceal evidence of the malware from someone inspecting the system. Sometimes described as *stealth mode*.

A common approach is to write a driver that gets installed with super-user privileges. This driver then overrides directory traversal, file operations, and process inspection, to ensure that the malware files and processes are not reported to any user programs when they use these facilities.

Virus and Worm

Both *virus* and *worm* functionality describe approaches to self-propagating malware.

Worm
When malware implements *worm* functionality, it means that the malware has the ability to replicate full copies of itself across the system and even over networks.

Virus
When malware implements *virus* functionality, it means that it has the ability to insert its functional code into existing programs and files on your system.