Malware Analysis on a Budget

Coleman Kane
Coleman.Kane@ge.com
kaneca@mail.uc.edu
@colemankane
`https://github.com/ckane`

May 21 2016

/me

- Security Operations, Cyber Intelligenge for GE Aviation
- Ph. D. candidate, Computer Science Engineering
  Cyber Operations track, University of Cincinnati
- B.S. Computer Engineering, University of Cincinnati
- CRITs project contributor
- FreeBSD community member & contributor since 2001
- 6 years experience in IT Security, at least 10 years in CS and IT
- Cincinnati native
- Running (because I like cake)
- 6 year old daughter
- Fancies cats

Go find evil

Go find evil

What is malware?

A very good place to begin is to define what is considered to be "malware". In my experience, in attacks, there are two classes of malware:

1. Artifacts designed to be used for malicious purposes (attack tools, exploits, etc.)
2. Artifacts designed for legitimate use, that are also used for malicious attacks

The second case will involve tools that may or may not be classifiable in every context as "malware" within your environment.

Why Do Malware Analysis?

There is a constant arms race between the cybersecurity industry and malware authors

IT PUTS THE MALWARE ON ITS HOST,
OR ELSE IT GETS THE PHISH AGAIN
imgflip.com

Why Do Malware Analysis?

A very valid question to ask yourselves:

- *What are you trying to get out of malware analysis?*

When building a security program, a core goal of malware analysis should be to document unique characteristics of malicious artifacts, in order to detect (and even prevent) them when they are introduced in your environment.

Embrace open-source security tools and the public community that supports and nurtures them.

Build a chart, mapping analysis outputs to program inputs.

As your security program matures, similarly mature your analysis program.

Two Faces of Malware Analysis

There are primarily two sides to malware analysis. Gathering data on both of them is important:

- **Dynamic**: When the malware is executing, what effect does it have on the host system and your networked environment?
- **Static**: When the malware is not executing, what characteristics distiguish its data from the rest of the artifacts in your environment (including other malware).

We will briefly go over a number of analysis tools that our team and colleagues have found to be helpful. Most of them fall into one of the two above categories, however some of them can assist with both.

Let’s Go Shopping

Analysis Hardware

I have found it helpful to use a separate system for analysis, frequently a Linux server that I can log into. When a Windows environment is needed, I can utilize Virtual Machines via RDP on this server. I recently went out and purchased the following hardware, for approximately **$400**.

- Asus M5A97 Motherboard (o/b NIC, HDMI): $60
- Case w/ Power: $30
- AMD FX 8320E CPU (8 core, 3.2GHz): $90
- 3TB SATA HD 7200rpm: $80
- 32GB RAM (4x8GB): $140

I put Ubuntu 16.04 onto it, in my case, and it installed pretty easily.

Dynamic: Cuckoo Sandbox and Virtualbox VM

Dynamic analysis requires a safe and isolated environment to execute malicious software, such that it doesn't expose you to undesired risk.

- Oracle Virtualbox (`http://virtualbox.org`) is a free and (mostly) open-source platform to provide virtual machine environments
- Cuckoo Sandbox (`http://cuckoosandbox.org`) is also free, and provides monitoring and data management for VM's with a focus on our use case of malware analysis
- VMCloak (`http://vmcloak.org`) is also free, and provides a helpful framework for automating the maintenance work involved in creating sandbox VM images

Oracle Virtualbox offers RDP and VNC options for its VM's so that you may interact with a system while it is executing malware. You can also install Virtualbox on your desktop/laptop, and use the same VMs there.

Dynamic: Sandboxes

In many cases, I have found that it is more valuable to build a few "least secure" images of varying configurations, than attempting to recreate your environment's "default build".

Why?: Because chances are better than not that configuration variance across your environment is high. Your systems are secured when they leave IT. Then you let your users have them.

You want to collect and document **what would happen if the attack is successful**, rather than use the tools to "validate your newly-issued hardware is secure". The latter is a valuable data point to know, but save that for when you have ample funding for it as a project of its own.

Static Analysis Overview

There are a large amount of static analyzers out there. The below families describe some types of static analyzers that should be helpful in quickly getting your system off the ground:

- **Meta-analyzers**: Analysis engines that are comprised of multiple analysis stages and document deep inspection findings in a structured output
- **Disassemblers**: Tools developed to identify machine code and convert it into human readable instruction mnemonics
- **Parsers**: Tools used to parse structured, typically binary, file formats into human readable reports, and even extracted artifacts they contain
- **Decompilers**: Kind of hybrid between parser & disassembler - can convert bytecode into source code
- **Scanners**: Tools used to scan contents, activity logs, and other artifacts for signatures of malicious activity

Scanners

Scanners

Some “scanners” you may be familiar with include most “signature based” Anti-Virus systems, such as Sophos, Symantec, McAfee. In many cases, these maintain a database of virus definitions that amount to patterns that recognize unique content within malicious files. Typically, you cannot add new information to this database yourself.

Victor Alvarez’s Yara Project: `http://plusvic.github.io/yara/`

Probably the most widely used tool for building your own signature database. Lots of open-source places to start:

- `https://github.com/Yara-Rules/rules/`
- `https://github.com/Xen0ph0n/YaraGenerator`
- `https://malwareconfig.com/yara/`

Scanners (yara example)

```
rule Tinba2 {
 meta:
  author = "n3sfox <n3sfox\@gmail.com>"
  date = "2015/11/07"
  description = "Tinba 2 (DGA) banking trojan"
  reference = "https://securityintelligence.com/tinba-malware-
               reloaded-and-attacking-banks-around-the-world"
  filetype = "memory"
  hash1 = "c7f662594f07776ab047b322150f6ed0"
  hash2 = "dc71ef1e55f1ddb36b3c41b1b95ae586"
  hash3 = "b788155cb82a7600f2ed1965cffc1e88"

 strings:
  $str1 = "MapViewOfFile"
  $str2 = "OpenFileMapping"
  $str3 = "NtCreateUserProcess"
  $str4 = "NtQueryDirectoryFile"
  $str5 = "RtlCreateUserThread"
  $str6 = "DeleteUrlCacheEntry"
  $str7 = "PR_Read"
  $str8 = "PR_Write"
  $pubkey = "BEGIN PUBLIC KEY"
  $code1 = {50 87 44 24 04 6A ?? E8}

 condition:
  all of ($str*) and $pubkey and $code1
}
```

Scanners (yara regex example)

```
rule jRat
{
 meta:
  author = " Kevin Breen <kevin@techanarchy.net>"
  date = "2014/04"
  ref = "http://malwareconfig.com/stats/jRat"
  maltype = "Remote Access Trojan"
  filetype = "Java"

 strings:
  $meta = "META-INF"
  $key = "key.dat"
  $conf = "config.dat"
  $jra1 = "enc.dat"
  $jra2 = "a.class"
  $jra3 = "b.class"
  $jra4 = "c.class"
  $reClass1 = /[a-z]\.class/
  $reClass2 = /[a-z][a-f]\.class/

 condition:
  ($meta and $key and $conf and #reClass1 > 10 and #reClass2 > 10)
  or ($meta and $key and all of ($jra*))
}
```

Parsers

WHAT AIN'T NO COUNTRY I EVER HEARD OF.
DO THEY SPEAK ENGLISH IN WHAT?
quickmeme.com

Parsers

Your users work with many structured file types in their daily work. Structures within these files are leveraged by adversaries to target vulnerabilities in specific parts of the applications used to open them.

When analyzing these attacks, it is important to use a parser that doesn't happen to be composed of the same software code targeted by the exploit. The community has built a number of parsers for common file formats to meet this need, available for free.

Parsers (open source)

Some parsers, and the file types they operate on:

- `python-oletools / olevba`, `http://www.decalage.info/python/oletools`, works on MSOffice "OLE" file formats which include old (pre-DOCX) files, as well as many components inside newer office documents. Can parse VB macros from documents.
- `pdf-parser, pdf-tools`, `https://blog.didierstevens.com/programs/pdf-tools/`, works on PDF documents
- `officeparser.py`, `https://github.com/unixfreak0037/officeparser`, similar to oletools above, which was based upon this utility
- `p7zip`, `http://p7zip.sourceforge.net/`, multi-archive parser. Remember how newer MSOffice and OpenOffice documents are just ZIP archives? Also extracts EXE sections.
- `exiftool`, `http://search.cpan.org/~exiftool/`, originally for extracting image header data, at some point was extended to basically parse every file header under the sun

Decompilers

Decompilers

Decompilers parse executable bytecode, and attempt to reverse-engineer it back into source code that could be human readable and more expressive in helping identify the nature of malware.

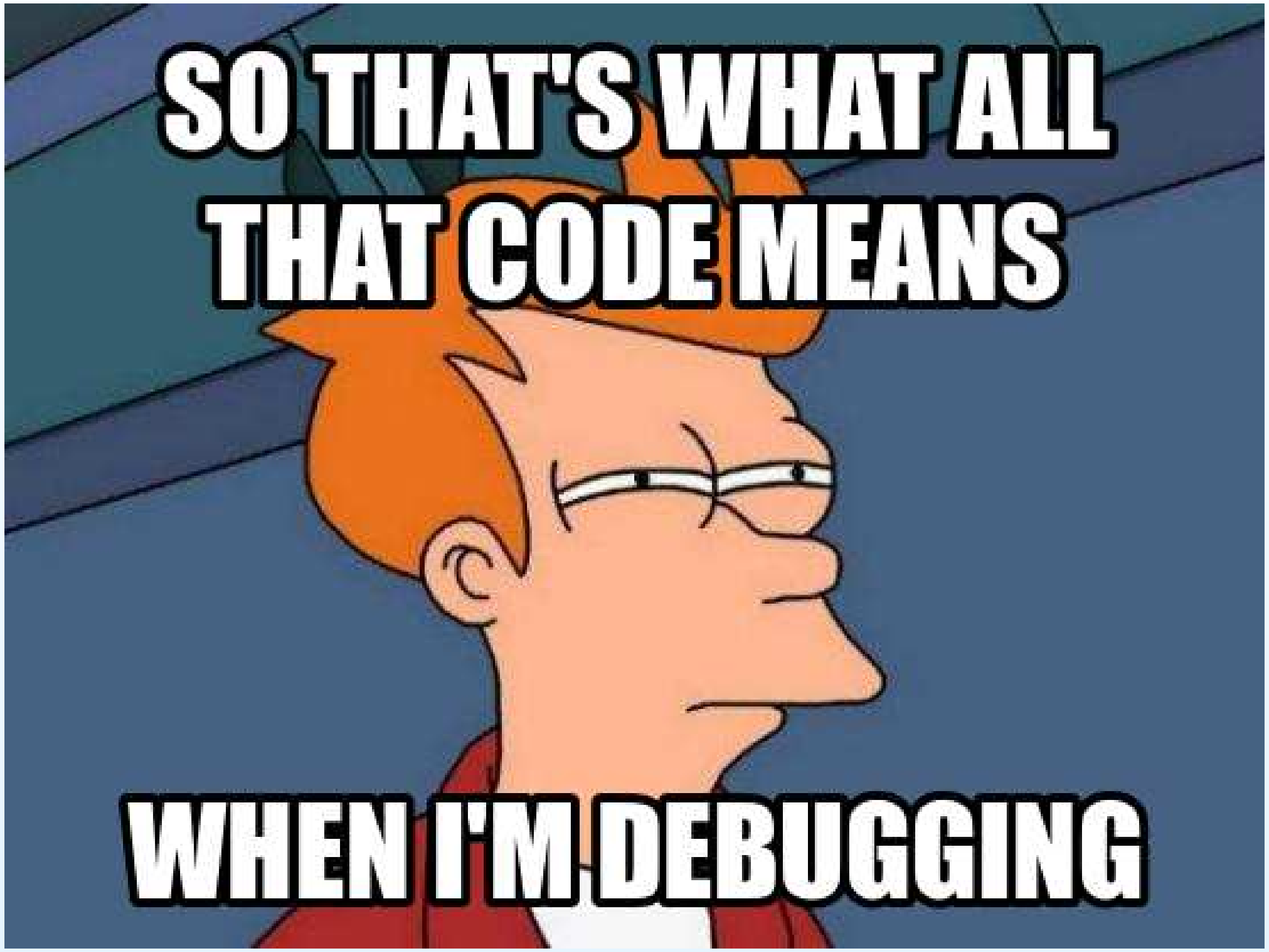
We have observed many samples of malware written in C# .NET, Java, and Flash (compiled ActionScript) and have found some utilities that are useful in recovering source code to help analysis.

In some ways, you can look at a decompiler as a special class of file parser.

Decompilers (cont.)

- **C#**: dotPeek & ILSpy
  - `http://ilspy.net/`: ILSpy
  - `https://www.jetbrains.com/decompiler/`: dotPeek
- **Java**: Jad (command-line), JD (library / GUI)
  - `http://jd.benow.ca`: JD (Java Decompiler project)
  - `http://www.javadecompilers.com/jad`: Jad (now defunct, but still very popular)
- **Flash ActionScript**: jpexs-decompiler
  - `https://www.free-decompiler.com/flash/` - formerly “asdec”

Disassemblers



Disassemblers

Disassembly involves reading binary machine code or bytecode, and then providing (*semi*-)human readable listing of the instruction sequences.
Some that I've used and are helpful:

- **ndisasm**: `http://www.nasm.us`, Disassembler from the Netwide Assembler distribution. Primarly works on raw data, x86 16, 32, and 64 bit.
- **ROSE Compiler**: `http://www.rosecompiler.org`, Disassembler that also can do block and path identification
- **IDA Free** / **Pro**: `https://www.hex-rays.com/products/ida/`, De-facto standard for malware analysis at the assembly level. Free version is quite functional and featureful.

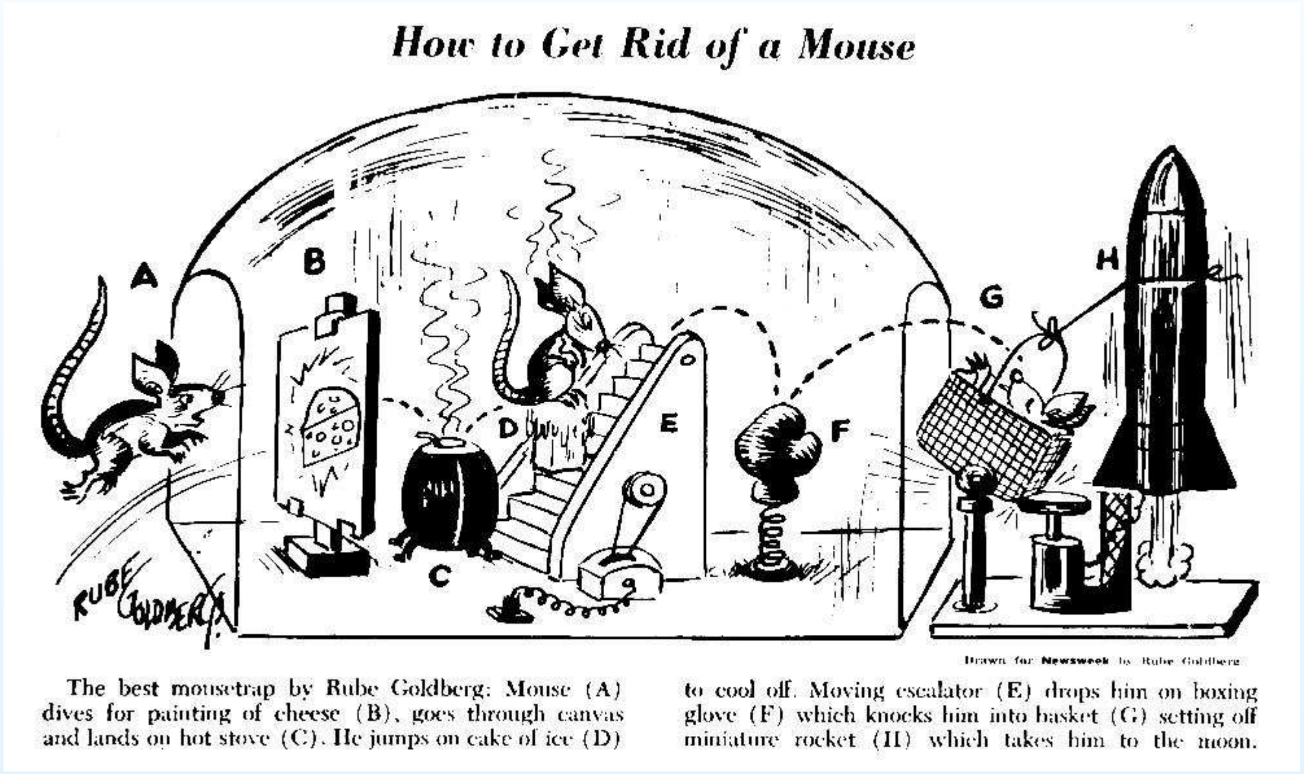
Debuggers

Frequently, you encounter malware that is obfuscated or packaged in a way that introduces hurdles to static analysis. In these cases, it is desirable to have a view of the disassembly during program execution, and to be able to control program execution.
Some popular popular debuggers are below, and some static analysis tools (like IDA) offer some online debugger features as well:

- **OllyDbg**: `http://www.ollydbg.de`, a really old and really popular debugger for windows
- **Immunity Debugger**: `http://debugger.immunityinc.com`, a newer debugger that's similar to OllyDbg with some newer features and more recent updates
- **GNU Debugger (GDB)**: `https://www.gnu.org/software/gdb/`, ubiquitous debugger for today's UNIX-like environments (including Mac!)

Meta-Analyzers

Meta-analysis frameworks are commonly pluggable systems that join multiple analysis tools together into an analysis suite, complete with consolidated reporting, and sometimes archival features.


How to Get Rid of a Mouse
A
B
C
D
E
F
G
H
RUBE GOLDBERG
Drawn for Newsweek by Rube Goldberg

The best mousetrap by Rube Goldberg: Mouse (A) dives for painting of cheese (B), goes through canvas and lands on hot stove (C). He jumps on cake of ice (D) to cool off. Moving escalator (E) drops him on boxing glove (F) which knocks him into basket (G) setting off miniature rocket (H) which takes him to the moon.

Tie things together: Meta-Analyzers

Meta-analysis frameworks are commonly pluggable systems that join multiple analysis tools together into an analysis suite, complete with consolidated reporting, and sometimes archival features.

These are frequently targeting automated, bulk analysis use-cases, and are **my #1 recommended place to jump in to get involved in contributing**.

In most cases, these provide an analysis pipeline as well as a plugin framework for analysts to bring their own tools to the party with minimal effort.

Meta-Analyzers

Two of my favorites:

- **LaikaBOSS**: Analysis framework developed and released recently by Lockheed Martin. Lots of community activity and capability.
  - `https://github.com/lmco/laikaboss`
  - `http://lockheedmartin.com/us/what-we-do/information-technology/cybersecurity/laika-boss.html`
- **MASTIFF**: Analysis framework and storage system developed by Tyler Hudak (*@SecShoggoth*) of KoreLogic. Offers free online service.
  - `https://mastiff-online.korelogic.com/`
  - `https://git.korelogic.com/mastiff.git/`

Organizing Things

Organizing Things

A common organizational strategy I've used is to manage each collected artifact as an "investigation", creating a folder to permanently store analysis tool outputs.

Build a script that can execute all command-line analysis tools in one go.

Something similar to below is not uncommon, and very helpful for searching:

```
mwzoo/
  |
  +--- artifact1.exe/
             |
             +--- artifact1.exe
             |
             +--- malware -> artifact1.exe
             |
             +--- mastiff-output/
             |
             +--- idapro-artifacts/
             |
             +--- cuckoo-runs/
```

Questions?