Lec 17: Anti-Malware 2

CS492E: Introduction to Software Security

Sang Kil Cha
Recap

• Polymorphism

• Polymorphic encryption
Metamorphic Malware

• No pack/unpack code

• Automatically change the code itself *each time it propagates*
Metamorphic Malware (cont’d)

When propagate

Malicious Code

Morphing Code

Malicious Code’

Morphing Code’
How about ...?

Original Code → Metamorphic Engine

New Code\(_1\) → New Code\(_2\) → \ldots → New Code\(_n\)
Techniques for Metamorphism

- Add some dead code in random places in the code
- Reallocate registers
- Function reordering
- And many more …
Dynamic Analysis

• Behavioral analysis

• Run the program/system and observe behavior

Whether it is polymorphic or metamorphic, it will show the same behavior
Two Categories of Behavioral Detection

• Heuristic-based or Rule-based: detect malicious behavior
  – Remote shell is spawned from a process
  – Malware-specific behavior

• Anomaly-based: detect abnormal behavior
  – Define what normal (benign) behavior is
  – When your system behaves abnormally, raise an alarm

Which one is better? And why?
Heuristic-based Approach: SNORT

- Observe network behaviors
- Consist of a large collection of rules
Anomaly-based Approach

Try to define normal (or expected) behavior in order to identify malicious behavior!

3 Types of Anomalies

- *Point anomalies*: defined with an individual data point
- *Contextual anomalies*: defined in a certain context
- *Collective anomalies*: defined with a collection of related data
Point Anomalies

If an individual data instance can be considered as anomalous with respect to the rest of data, then the instance is termed as a point anomaly.

From Anomaly Detection: A Survey, CSUR 2009
Example: Credit Card Fraud Detection

Customer X typically spends 1,000 won ~ 100,000 won per transaction.

A transaction for which the amount spent is 10,000,000 won is anomalous.
Contextual Anomalies

If a data instance is anomalous in a specific context (but not otherwise), then it is termed as a contextual anomaly.

a.k.a. conditional anomalies

From Anomaly Detection: A Survey, CSUR 2009
Example: Temperature

30 °C in *winter of Daejeon* is abnormal
Example: Credit Card Fraud Detection

Customer X typically spends 100,000 won per week.

Weekly bill of 1,000,000 won during Chuseok holiday is normal.
Collective Anomalies

If a collection of related data instances is anomalous with respect to the entire data set, it is termed as a collective anomaly.

From Anomaly Detection: A Survey, CSUR 2009
Example: Money Transfer

A transfers 100,000 won to X: normal
B transfers 100,000 won to X: normal
C transfers 100,000 won to X: normal
D transfers 100,000 won to X: normal
...
Y transfers 100,000 won to X: normal
Z transfers 100,000 won to X: normal

Abnormal
Behavioral IDS

Collective anomaly detection for HIDS

Natural Immune System

Can we build a malware detection system that is as good as natural immune system?
Definition of Self

• Collect a sequence of system calls for normally operating programs

• Build a profile of normal behavior based on the sequence

• When we observe discrepancies, we treat them as anomalies
Building a Pairwise Profile

- Sliding window of size 4
- Normal execution example:
  
  open-read-mmap-mmap-open-getrlimit-mmap-close

<table>
<thead>
<tr>
<th>call</th>
<th>position 1</th>
<th>position 2</th>
<th>position 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>open</td>
<td>read, getrlimit</td>
<td>mmap</td>
<td>mmap, close</td>
</tr>
<tr>
<td>read</td>
<td>mmap</td>
<td>mmap</td>
<td>open</td>
</tr>
<tr>
<td>mmap</td>
<td>mmap, open, getrlimit</td>
<td>mmap</td>
<td>getrlimit, mmap</td>
</tr>
<tr>
<td>getrlimit</td>
<td>close</td>
<td>close</td>
<td></td>
</tr>
<tr>
<td>close</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Detecting Anomaly

- Sliding window of size 4
- Abnormal execution example:
  

In total 4 mismatch out of 18 (3 x 5 + 2 + 1) possible pairwise mismatches = 22% miss rate

If the miss rate is above a certain threshold, we say the system is abnormal
Obtaining Execution Profile?

• Ptrace

• Attaching debugger to a running process
  – GDB, LLDB, WinDbg, etc.
  – Single stepping: context switching for every single execution

• Instrumentation
  – Pin, DynamoRio, Valgrind, etc.
Defeating Behavior-based Detection

Mimic normal system call sequences!

Mimicry Attacks on Host-based Intrusion Detection Systems, CCS 2002
More Fundamental Question

• How can we trick dynamic analysis?

• How can we hide execution behavior of a program?
Platform-Independent Programs
Common Assumption

A single executable program runs only on a *specific platform*.
Common Assumption (cont’d)

A single executable program runs only on a *specific platform*.
A program can run only on one platform

Automatically generate single binary string that is valid on multiple platforms
A Platform is ...

• ISA (Instruction Set Architecture)
  - ARM, MIPS, Intel

• OS (Operating System)
  - Linux, macOS, Windows
Platform-Independent Program (PIP)
So, Why PIP?

Cool, new paradigm!
Programmer’s Perspective

Advanced Install Options & Other Platforms

- Windows 64-bit
- Windows 32-bit
- macOS
- Linux 64-bit
- Linux 32-bit

Google Play

App Store
Attacker’s Perspective

- Platform-independent exploit (shellcode)
- Platform-independent malware
Execution-based Steganography

*Hide runtime behavior* of the program!
Intuition: False Friends

Γεια
/yα/
Hi

아
/ya/
Hey you!

Greek

Korean
Intuition: False Friends

What’s up?

What are?

English

Korean
Instruction Overlap

\[ 56565656_{16} \]

push esi
push esi
push esi
push esi
bnel $r18,$r22,0x1595c
Basic Construction: Finding Overlaps between Jump Instructions

```
ARM b 0xbb4

x86 jmp 0x4

ARM Logic

x86 Logic

eb0200ea_16 ...
```
Challenges

• Automatically constructing PIPs

• Turing-complete language
  – PIP meta-language for generating PIPs
Single PIP Header

How many headers possible?

Size may differ

Separately compiled binary

PIP

eb0200ea_{16} ... x86 Logic ARM Logic
Over billions of PIP Headers Possible!

• For x86, ARM, and MIPS

• Various jump offsets

But, each binary string should be compiled separately!
Turing-Complete PIP?

• Construct platform-independent instructions
  – A platform-independent gadget is a platform-independent instruction

• Splice platform-independent instructions using jump instructions
Turing-Complete Language with Platform-Independent Gadgets

A platform-independent Instruction

```
add eax, ebx
add r0,r0,r1
```
Finding Gadget Headers

• Headers must be side-effect free

• For all platforms, a gadget header is decoded for each platform as in a form of

\[(\text{nop}^*)(\text{branch})(.*\)]

• **Example:** eb0200ea
  - ARM: \text{b} 0xbb4
  - x86: \text{jmp} 0x4

• \(\gg\) billions of 12-byte overlaps for x86, ARM, MIPS
PIP Allows Different Logic for Each Platform

Desired Behavior A

Mac

Windows

Desired Behavior B
Execution-based Steganography
Classic Steganography

Alice says hello to Bob

Shared-secret: Vocabulary

Hello!

Alice

Secret X

Bob

Secret X

Warden

Alice says hello to Bob
Execution-based Steganography

Secret X (Running on MIPS)

Alice

Shared-secret: Platform

Bob

Secret X (Running on MIPS)

Alice sends a firefox binary

Warden

Dynamic analysis on x86

Alice sends a firefox binary

Secret X (Running on MIPS)
Other Results

• 8 platform-independent shellcode (x86, ARM, and MIPS)
  − Confirmed with 2 real-world exploits

• Platform-independent malware
  − A virus that spreads over NFS

• Platform-independent shellcode for OSes
  − FreeBSD, Linux, and Mac OS X
• Some OS rejects a program if the file format of the program contains wrong architecture information.
  - Some executable file format does not include architecture information (e.g., COFF).

• Architecture checks are important against PIP, even though they were likely not intended as a security measure.
  - Embedded archs, emulators may all be vulnerable to PIP attacks
x86 vs. x86-64

15-byte code: \x31\xc9\x41\xe2\x08\x90 ... 

### x86

xor ecx,ecx  
inc ecx  
loop 0xd  
nop  
...

### x86-64

xor ecx,ecx  
loop 0xd  
nop  
...
Conclusion

• Metamorphism: harder to break than polymorphism

• Dynamic analysis (behavior-based analysis) for the rescue?

• Mimicry attack

• Execution-based steganography (with PIP)
Questions?