Binary analysis:
Malware Detection & Analysis

Yue Duan
Outline

● Malware basics
● Research papers:
  ○ Semantics-Aware Malware Detection
  ○ Panorama: Capturing System-wide Information Flow for Malware Detection and Analysis
  ○ BareCloud: Bare-metal Analysis-based Evasive Malware Detection
Malware Detection Basics

- **Malware**
  - software intentionally designed to cause damage

- **Malware Detection techniques**
  - Static analysis
  - Dynamic analysis
Malware Detection Basics

- **Static analysis**
  - testing and evaluation of an application by examining the code without executing the application
  - **Pros:**
    - Good code coverage
    - Time efficiency
  - **Cons:**
    - False positives
    - Code obfuscation
    - Encryption
Malware Detection Basics

- Dynamic analysis
  - testing and evaluation of an application during runtime
  - Pros:
    - Capture behaviors accurately
  - Cons:
    - Poor code coverage
    - High runtime overhead
Semantics-Aware Malware Detection

Mihai Christodorescu, Somesh Jha, Sanjit A. Seshia, Dawn Song, Randal E. Bryant

IEEE S&P 2005
Semantics-Aware Malware Detection

- State-of-the-art techniques - pattern matching
  - susceptible to obfuscations
  - purely syntactic
  - ignore the semantics of instructions

- Attacker’s goal - preserve behaviors
  - Transformation of code and data
  - Addition of new code and data
Semantics-Aware Malware Detection

No Resilience to Obfuscations

*False Negative Rate for Obfuscated Worms*

Source: “Testing Malware Detectors” (ISSTA 2004)
Semantics-Aware Malware Detection

● Major contributions
  ○ Introduce semantic signatures
    ■ Combine syntactic and semantic information
  ○ Develop a prototype based on the signatures
  ○ Empirical study shows that one semantic signature can detection a malware family
Semantics-Aware Malware Detection

- Example: detect mass-mailing virus
  - Detect email capability
  - Detect self-propagation

```c
s = socket (...);
connect (s);
...
sprintf (buf, “EHLO %S”, dnsname);
send (s, buf);
```
Semantics-Aware Malware Detection

- Variant 1: string manipulation
  - Hide known constants
  - Syntactic signature does not match

```c
s = socket (...);
connect (s);

...
char str [80];
strcpy (str, “EH”);
strcat (str, “LO %S”);
sprintf (buf, str, dnsname);
send (s, buf);
```
Semantics-Aware Malware Detection

- Variant 2: string obfuscation
  - Hide known constants using simple encryption techniques

```c
s = socket (...);
connect (s);
...
char* str = decrypt (encrypted string);
sprintf (buf, str, dnsname);
send (s, buf);
```
Semantics-Aware Malware Detection

- **Attackers**
  - Same behavior in different forms
  - Contain same semantics
- **Semantic signatures**
  - Combine syntactic info and semantic info
  - Detect any variant
Semantics-Aware Malware Detection

- Evaluation

McAfee uses individual signatures for each worm.

**Semantic signatures provide forward detection.**
Semantics-Aware Malware Detection

- Evaluation: Obfuscation resilient

<table>
<thead>
<tr>
<th>Obfuscation Type</th>
<th>Semantics-Aware Detection</th>
<th>McAfee</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average Time</td>
<td>Detection Rate</td>
</tr>
<tr>
<td>Nop insertion</td>
<td>74.81 s</td>
<td>100%</td>
</tr>
<tr>
<td>Stack op. insertion</td>
<td>159.10 s</td>
<td>100%</td>
</tr>
<tr>
<td>Math op. insertion</td>
<td>186.50 s</td>
<td>95%</td>
</tr>
</tbody>
</table>
Panorama: Capturing System-wide Information Flow for Malware Detection and Analysis

Heng Yin, Dawn Song, Manuel Egele, Christopher Kruegel, Engin Kirda

ACM CCS 2007
Panorama

- Malware detection
  - Signature based detection
    - Cannot detect new malware and variants
    - Semantic-aware signature can detect some
  - Behavior based
    - Heuristics: high false positives and false negatives
    - Hooking-based

- Malware analysis
  - Manual process
Panorama

● Observation
  ○ Information access and processing (IAP) behavior
    ■ malicious/suspicious IAP behaviors as traces for malware detection and analysis
    ■ Steal, tamper, or leak sensitive information
Panorama

- Approach: Whole-system dynamic taint analysis
  - Run the system in an emulator
  - Selectively mark data as tainted
  - Monitor taint propagation
  - Extract OS-level knowledge
  - Generate taint graphs
  - Graph based detection and analysis
TAINT ANALYSIS

```java
String a = request.getParameter("foo");
String b = a + "bar";
String c = b.replaceAll("foo", "bar");
byte[] d = c.getBytes();
String e = new String(d, "UTF-8");
response.getWriter().println(e);
```

Vulnerability!
Panorama

- **Taint Graph**
  - Instruction and hardware level raw events
  - OS-level knowledge

A password stealer catches the password and saves it into a log.
Panorama

● Graph-based Detection and Analysis
  ○ Anomalous information access
    ■ Keyloggers:
      ● Text: when sending keystrokes to a text editor
      ● Password: when sending password to a web form
    ■ Backdoors:
      ● ICMP: when pinging a remote host
      ● FTP: when logging into a server
      ● etc
  ○ Anomalous information leakage
    ■ URL: the keystrokes sent to the address bar
    ■ HTTP: the incoming HTTP traffic
Panorama

- Evaluation: effectiveness

- Evaluation: performance
  - Curl, scp, gzip: 20x slowdown on average
  - Test cases: 10-15 mins
Panorama

- Evaluation: Case study

Google Desktop obtains the incoming HTTP traffic, saves it into two index files, and then sends it out through an HTTPS connection, to a remote Google Server.
BareCloud: Bare-metal Analysis-based Evasive Malware Detection

Dhilung Kirat, Giovanni Vigna, and Christopher Kruegel

Usenix Security 2014
VM environment is different from real machine.
BareCloud

- VM detection and evasion:
  - CPU instruction semantics
  - Timing attacks
  - VM bugs
  - Etc

- Motivation:
  - Can we automatically identify evasive malware while preserving transparency?

- Key idea:
  - Collect behavioral information on multiple platforms and compare the behaviors
BareCloud

- System overview

prescreening

Execution

Behavior extraction

Behavior comparison
BareCloud

- Prescreening
  - Select interesting samples
    - Likely to have environment-sensitive behaviors
  - Use Anubis platform
BareCloud

- **Execution**
  - Run malware sample on 4 platforms simultaneously
    - Bare-metal
    - Anubis (emulator)
    - Ether (Intel VT)
    - Virtualbox (Type2 hypervisor)
BareCloud

- Behavior extraction
  - Two common ways
    - VMI based approach
    - In-guest monitoring
  - Problem:
    - Not transparent enough
BareCloud

- **Behavior extraction**
  - BareCloud extracts file system behaviors and network behaviors
    - File system: compare the disk contents from before and after the malware execution
    - Network: use an external traffic capture component
BareCloud

- Behavior comparison
  - Compare the behavior profiles in a hierarchical way
BareCloud

- Evaluation
  - 110,000+ malware samples
  - 5835 evasive malware samples are found

<table>
<thead>
<tr>
<th>Environment</th>
<th>Detection count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anubis</td>
<td>4,947</td>
<td>84.78</td>
</tr>
<tr>
<td>Ether</td>
<td>4,562</td>
<td>78.18</td>
</tr>
<tr>
<td>VirtualBox</td>
<td>3,576</td>
<td>61.28</td>
</tr>
<tr>
<td>All</td>
<td>2,530</td>
<td>43.35</td>
</tr>
<tr>
<td>Total</td>
<td>5,835</td>
<td></td>
</tr>
</tbody>
</table>
Summary

● Malware detection and analysis
  ○ Static analysis
  ○ Dynamic analysis


● Panorama: Capturing System-wide Information Flow for Malware Detection and Analysis, *ACM CCS 2007*

● BareCloud: Bare-metal Analysis-based Evasive Malware Detection, *Usenix Security 2015*
Thank you!

Question?