Information Security Laboratory

07-Dynamic Analysis
Sang Kil Cha
Evolution of Malware

• Polymorphism

• Metamorphism

Obvious limitation of static analysis
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

You will use SNORT in this course.
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

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 Profile

• Sliding window of size 4
• Normal execution example:

<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, close</td>
<td>getrlimit, mmap</td>
<td>mmap</td>
</tr>
<tr>
<td>getrlimit</td>
<td>close</td>
<td></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 \((3 \times 5 + 2 + 1)\) possible pairwise mismatches = 22% miss rate

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

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

• Instrumentation
  – Pin, DynamoRio, Valgrind, etc.
Instrumentation

```c
void somefn()
{
    char array[42];

    for (int i = 0; i < 42; i++) {
        array[i] = i;
    }
}
```
void somefn()
{
    char array[42];
    printf("before loop\n");
    for (int i = 0; i < 42; i++) {
        printf("inner loop\n");
        array[i] = i;
    }
}
Static

CIL (CC 2002)
LLVM (CGO 2004)

Dynamic

Pin (PLDI 2005)
DynamoRio (CGO 2003)
Valgrind (PLDI 2007)

Source-based

Binary-based

PEBIL (ISPASS 2010)
DynInst (HPCA 2000)
Diablo (ISSPIT 2005)
Dynamic Instrumentation

Code

Code Cache
Dynamic vs. Static Instrumentation

• Dynamic
  − High overhead
  − Easy to instrument external libraries
  − Handles dynamically generated code

• Static
  − Fast
  − Difficult to instrument external libraries (need to be separately instrumented)
  − Cannot handle dynamically generated code
Writing a Debugger!

Next lecture
How to Defeat Dynamic Analysis?

• Anti-forensics

• Anti-debugging
Question?