Information Security Laboratory

06-Anti-Malware
Sang Kil Cha
The Arms Race (or Weapons Race)

Attacker vs. Defender
How Do We Implement Security?

• **Prevention**: no attack is possible

• **Detection**: identify something went wrong

• **Response**: respond to the detected threats

• **Recovery**: recover from the damage
Intrusion Prevention System (IPS)

Can you think of any example?
Intrusion Detection System (IDS)

• Host-based IDS (HIDS)

• Network-based IDS (NIDS)
IDS is Important in Security

• If we can detect early and quickly enough, we may be able to respond before any damage is done!

• Effective IDS ≈ IPS
Basic Terms

Truth

False Negatives

True Positives

False Positives

True Negatives

What I say

False
Precision = \( \frac{TP}{TP + FP} \)
Recall = $\frac{TP}{TP + FN}$
Accuracy = \frac{TP+TN}{TP+TN+FP+FN}
Our Goal High Accuracy

• 100% accuracy possible in practice?

• Low false positives and low false negatives

• 1% FP rate is good or bad?
  – Beware of the base rate fallacy
Base-Rate Fallacy

• A cognitive error whereby too little weight is placed on the base (original) rate of possibility (Taken from Investopedia)

• Example: a system with 1,000,000 files
  – 100 malicious files, 999,900 benign files
  – Base rate probability when selecting a file at random:
    ▪ The probability of selecting a benign file = 0.9999
    ▪ The probability of selecting a malicious file = 0.0001
  – My AV has FN-rate of 1% and FP-rate of 1%
  – When AV raises an alarm for a file, what is the probability that the file is indeed malicious?
Malware Authors vs. Defenders
Simplest Malware Detection

• Compute hashes of malware samples
• Compute hashes for target files and find ones that match with one of the malware hashes (a.k.a. signatures)

This is still used in AV (anti-virus) software. Why?
Easy to Bypass

• Add a dummy (dead) code

• Reorder instructions

• Replace instructions with semantically equivalent ones
Pattern Matching (RegExp)

closeDoc{-35}setTimeOut{-30}addAnnot...

A ClamAV signature for CVE-2016-0931 (Adobe Acrobat PDF exploit)
Polymorphism

Change the form of malware when it propagates to *bypass pattern matching*
Changing the Form?

• Malicious use:
  Bypass malicious code detection ($\approx$ Intrusion detection)

• Benign use:
  Software protection (make reverse engineering difficult)
Polymorphism Example

Decryption Routine

Jmp to XYZ

XYZ

Encrypted Code
Polymorphism Example

XYZ is often called **OEP** (Original Entry Point)

We can produce millions of distinct binaries (with the same semantics) by just changing the encryption key.
Self-Modifying Code

• Code that alters its own instructions while it is running

• $W^X$ (Write xor eXecute) policy of modern OS?

Can you write self-modifying code in pure C? (without inline disassembly?)
Polymorphism Example

Decryption Routine

Jmp to XYZ

Encrypted Code 1

Decryption Routine

Jmp to XYZ

Encrypted Code 2

Decryption Routine

Jmp to XYZ

Encrypted Code 3
Checking Decryption Routine

Decryption Routine

Encrypted Code 1

Jmp to XYZ

Decryption Routine

Encrypted Code 2

Jmp to XYZ

Decryption Routine

Encrypted Code 3

Jmp to XYZ
Possible to Create Signatures

Decryption Routine
Decryption Routine
Decryption Routine

Next Question:
Can we also make the decryption routine polymorphic?

See https://github.com/Yara-Rules/rules/tree/master/Packers
Polymorphic Encryption

Make the encryption/decryption routine unique!
Polymorphic Encryption (cont’d)

Original Code → Polymorphic Encryption Engine

Decryption Routine$_1$

Encrypted code$_1$

Decryption Routine$_2$

Encrypted code$_2$

…
Polymorphic Encryption Example

```java
for ( int i = 0; i < codeLen / 4; i++ ) {
    v = in[i]; // for every 4-byte value of the orig code
    key[i] = random_int(); // random 4-byte int
    op[i] = random_op(); // random operation
    switch ( op[i] ) {
        case ADD: v += key[i]; break;
        case SUB: v -= key[i]; break;
        case XOR: v ^= key[i]; break;
        ... // omitted
    }
    out[i] = v; // store the encrypted code
}
```
Polymorphic Decryption Example

```c
for ( int i = 0; i < codeLen / 4; i++ ) {
    v = in[i]; // for every 4-byte of the encrypted code
    k = key[i];
    switch ( op[i] ) {
        case ADD: v -= key; break;
        case SUB: v += key; break;
        case XOR: v ^= key; break;
        ...
    }
    out[i] = v; // store the decrypted code
}
// The encrypted code can be located here (self-modifying)
```
Can We Still Write Signatures?

• Signature database will easily blow up

• Simple static pattern matching does not help anymore

Any issues in polymorphic encryption?
In-Memory Detection

• The same original code will be eventually unpacked to memory at some point

• Memory-based scanning still works! (no more static detection)

• Generic unpacking technique exists
Generic Unpacking

• Renovo: A hidden code extractor for packed executables, *WORM 2007*

• PinDemonium a DBI-based generic unpacker for Windows executables, *BlackHat USA 2016*
Identifying OEP

• The original code will be eventually unpacked to memory at some point, and will be executed.

• Check if the current instruction has been generated at runtime
  − Mark the memory region that is written at runtime as “tainted”
  − When the instruction pointer jumps to a tainted memory region

• Repeat this process to handle multiple hidden layers
  − In each iteration, clean up the taint state
Some Heuristics to Detect OEP

• Check the entropy of code

• Check if known code patterns appear
Entropy before/after Unpacking

Entropy
MessageBox Original

Entropy
MessageBox encrypted with Obsidium

entropy
low medium high random
Metamorphic Malware

• No pack/unpack code

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

Malicious Code

Morphing Code

When propagate

Malicious Code’

Morphing Code’
How about ... ?

Original Code → Metamorphic Engine → New Code

New Code₁ → New Code₂ → ... → New Codeₙ
Simple Techniques for Metamorphism

- Add some dead code in random places in the code
- Reallocate registers
- Function reordering
- Etc.
Metamorphic Code Example

```c
int fun(int n)
{
    int x = 0;
    x += n;
    return x;
}
```

```c
int fun(int n)
{
    int x = 0;
    int y = 42; // XXX
    x += n;
    y *= x;     // XXX
    return x;
}
```
Question?