Flowers for Automated Malware Analysis

Chengyu Song and Paul Royal
College of Computing
Georgia Institute of Technology
Agenda

- Modern Malware
- History of Malware Analysis
  - Technologies, Detections, Transparency Requirements
- Inverting Environment Detection
  - Flashback
- Defeating Automated Malware Analysis
  - Host Identity-based Encryption (HIE)
  - Instruction Set Localization (ISL)
- Discussion
  - Potential Countermeasures
- Conclusion
Modern Malware
Modern Malware

- The centerpiece of current threats on the Internet
  - Botnets (Spamming, DDOS, etc.)
  - Information Theft
  - Financial Fraud

- Used by real criminals
  - Criminal Infrastructure
  - Domain of Organized Crime
Malware Cont’d

- There is a pronounced need to understand malware behavior
  - Threat Discovery and Analysis
  - Compromise Detection
  - Forensics and Asset Remediation

- Malware authors make analysis challenging
  - Direct financial motivation
Malware Obfuscations

● Pictorial Overview

Program A

Push EBP
MOV EBP, ESP
SUB ESP, 8
CALL 00401170
...

Obfuscation Tool

Encrypt/Compress/Transform

Transformed Machine Code
(Appears as Data)

<Unpack Code>

Program A’

● Project ZeroPack

ZeroPack

File bifrose.exe received on 02.25.2009 18:41:57 (CET)
Current status: finished
Result: 38/39 (97.44%)

File bifrose_zp.exe received on 02.25.2009 18:42:11 (CET)
Current status: finished
Result: 0/39 (0%)
Obfuscations Cont’d

- **Server-side Polymorphism**
  - Automate mutations

- **When done professionally: Waledac**

---

File **postcard.exe** received on 02.25.2009 22:03:16 (CET)
Current status: **finished**
Result: 35/39 (89.75%)

---

File **disc.exe** received on 02.25.2009 21:53:13 (CET)
Current status: **finished**
Result: 11/39 (28.21%)
Obfuscations Cont’d

- ISA Virtualized Malware
  - VMProtect, Code Virtualizer

![Diagram]

- Original Program $P_{x86}$
- ISA Virtualized Program
  - Bytecode $P_L$
  - Emulator $EM_{x86}^L$
- L Interface
- Translation
- x86 Interface
History of Malware Analysis Technologies
In-guest Tools

- Reside in the analysis environment
- Vulnerable to detection of monitoring instrumentation

```c
HMODULE kernel32 = NULL;
void *createfile_function_pointer = NULL;
unsigned char opcodes[2];

kernel32 = LoadLibrary("kernel32");
createfile_function_pointer =
    (void*)GetProcAddress(kernel32, "CreateFileA");
memcpy(opcodes, createfile_function_pointer, sizeof (opcodes));

if(opcodes[0] == 0xFF && opcodes[1] == 0x25){
    puts("Instrumentation detected.");
}
```
Reduced-privilege VMMs

- Operate through sensitive data structure relocation, binary software translation
- Vulnerable to detection of side effects
- In older versions of VMWare, SYSRET treated as NOP when executed in ring 3
Whole-system Emulators

- Operate by emulating processor ISA (e.g., x86)
- Vulnerable to detection of unfaithful CPU emulation

```c
#include <stdlib.h>
#include <stdio.h>
#include <windows.h>

int seh_handler(struct _EXCEPTION_RECORD *exception_record, void *established_frame, struct _CONTEXT *context_record, void *dispatcher_context)
{
    printf("Malicious code here.\n");
    exit(0);
}

int main(int argc, char *argv[])
{
    unsigned int handler = (unsigned int) seh_handler;
    printf("Attempting detection.\n");
    __asm("movl %0, %%eax\n	pushl %%eax\n" ::
        "r" (handler): "%eax");
    __asm("movl %esp, %fs:0\n	z\n    movl %esp, %fs:0\n	"):
    __asm(".byte 0x26, 0xcf\n
    return EXIT_SUCCESS;
}
```
Hardware Accelerated VMs

- Operate through use of hardware virtualization extensions (e.g., Intel VT-x or AMD SVM)
  - Extensions to x86 ISA (new instructions)
- Certain instructions cause VMExits
  - Must be handled correctly
- Older versions of KVM terminate with unhandled exit on guest execution of VMREAD
Transparency Requirements

- Higher Privilege
- No Non-privileged Side Effects
- Same Instruction Execution Semantics
- Identical Exception Handling
- Identical Notion of Time
Requirements Cont’d

- **In-guest Tools**
  - No higher privilege
  - Non-privileged side effects
  - Exception handling issues

- **Reduced Privilege Guests (VMware, etc)**
  - Non-privileged side effects

- **Emulation (QEMU, Simics)**
  - No identical instruction execution semantics
State of Detection

- Analysis tool/environment detection is a standard, inexpensive option
State of Detection Cont’d

- **Detections by Popular Malware**
  - **Conficker**
    - Checks for relocated LDT
  - **TDL4**
    - Checks for device emulation via WQL
  - **Bredolab**
    - Checks for device emulation via DeviceIoControl()
Inverting Analysis Detection
Nature of the Arms Race

- Until recently, malware was “analysis environment aware”
  - Detect analysis environments
  - Execute successfully otherwise
- Malware could be “analysis environment oblivious”
  - Exploit observation that malware is overwhelmingly collected in one environment and analyzed in another
  - Bind to and successfully execute only on originally infected host
Flashback

- Propagated in part by drive-by downloads
- Payload is only intermediate agent
  - Agent gathers hardware UUID, submits request to C&C for full version
  - Hardware UUID hashed (MD5), hash used as decryption key to RC4 stream cipher
  - Full version will only run on host with same hardware UUID
Defeating Automated Malware Analysis
Malware DRM

● Goal
  – Make automated malware analysis ineffective and unscalable

● Approach
  – Cryptographically bind a malware instance to the originally infected host

● Techniques
  – Host Identity-based Encryption (HIE)
  – Instruction Set Localization (ISL)
Host Identity-based Encryption

- Replace random encryption key with a key derived from host identity

- Host ID: Information that can uniquely identify a host
What to encrypt

- Full binary?
  - May not be a good idea
  - Leaves hint for brute-force cracking
- Instead, only encrypt critical mechanisms
  - For example, encrypt C&C domain names or portions of domain name generation algorithm (DGA)
Requirements for Host ID

- Unique
- Invariant (to avoid false positives)
  - Can be as short as lifecycle of the malware campaign (e.g., days or weeks)
- Can be gathered without privileges
- No special hardware support
HIE Cont’d

- **Prototype Host ID (Windows)**
  - Subset of Process Environment Block
    - Username, Computer Name, CPU Identifier
  - MAC Address
  - GPU Information
    - GetAdapterIdentifier
  - User Security Identifier (SID)
    - Randomly generated by the OS
    - Unique across a Windows domain
HIE Cont’d

- Key Derivation Function (KDF)
  - Key = KDF(ID, Salt, Iteration)
  - ID = Concatenation of all information
  - Salt = Random number >= 64 bits
  - Work Factor/Iteration = 10+/100+
  - KDF = Bcrypt or SHA family
Deployment Logistics

- Host ID must be determined before malware instance is installed
  - Use intermediate downloader agent
- Intermediate agent could be used by researchers to obtain instance bound to analysis environment
  - Use short-lived, one-time URLs similar to password reset procedures
Advantages

- Protections of Modern Cryptography
  - Knowledge of how key is derived does not affect the integrity of the protection

- Sample Independence
  - Intelligence collected from one malware instance provides no advantage in analyzing another
Instruction Set Localization

- **Why ISL?**
  - Pure host-based protection is not sufficiently resistant to forgery

- **Goal of ISL**
  - Use C&C server to “authenticate” malware client based on both host and network identity
  - Decouple malicious functionality to prevent offline analysis
Malware as Platform-as-a-Service

- HIE-protected binary contains no malicious functionality
- Binary acts as interpreter of bytecode for malicious tasks served by C&C
- Task Bytecode
  - Can be unique to each executable
    - A different bytecode ISA for each host
  - Alternatively, can be protected by key derived from both host and network-level identifiers
ISL Cont’d

- Replace random instruction set with instruction set bound to the host

Diagram:
- Host-ID Generation Module
- Malware
- Emulator (EM\textsubscript{x86})
- Client
- C&C Server
- Translation
- Malicious Functionality (P\textsubscript{X86})
- Bytecode (P\textsubscript{L})
- Host-ID
- Network ID
- P\textsubscript{L}
ISL Cont’d

- **Prototype Network ID**
  - Geo-location
    - Granularity of state/province level (IP address is not stable)
      - Permits certain level of mobility
    - Autonomous System Number (ASN)
      - Geo-location may be outdated or incorrect
  - Collected at C&C
    - Considered intractably difficult to forge
Alternative to Unique Instruction Sets

- Instruction set derivation is not trivial
- Use *task decryption key*
  - Assigned when the malware instance is delivered to the host
  - Encrypt bytecode tasks using the unique ID (the key derived from host ID and network ID)
    - KDF = HMAC(unique ID), or keyed hash, with the secret key kept at C&C server
ISL Cont’d

- Advantages
  - HIE-protected binary is only an interpreter (contains no malicious functionality)
    - Instance cannot be analyzed offline
  - Complementary to HIE for tasks served to the interpreter
    - Unless the analyst can correctly mimic the host and network environment, tasks will not decrypt/execute
Discussion
Operational Security

- Both HIE and ISL use modern cryptography
  - Same environment must be provided for successful analysis
  - Without access to original environment, entire key space must be searched
    - Key space can be of arbitrary size
  - Some configurations may be impossible to duplicate
Operational Security Cont’d

- HIE and ISL are insensitive to analysis techniques
  - General knowledge of these techniques does not compromise protections offered
  - Granularity of analysis used does not affect protections
  - Protections can be broken only if the configuration parameters of the original execution environment are matched
Potential Countermeasures

- Analyze malware on the original infected host
  - Approach would require allowing otherwise blocked suspicious/known malware to execute on a legitimate system
    - Could impact business operations and continuity
    - Would have complex legal and privacy implications

- Use high-interaction honeypot
  - Bind malware to analysis environment by replicating compromise circumstances
    - Inefficient
    - Bound samples will comprise only a small portion of all collected samples
Countermeasures Cont’d

- Collect and duplicate host and network environment information
  - Depending on the information, may have privacy and policy problems
  - Duplicating network identifier requires analysis system deployment on an unprecedented and globally cooperative scale
Countermeasures Cont’d

- Collect and duplicate only host identifier, record and replay the network interaction in separate environment
  - Without small additional protection, could bypass ISL
  - Mitigated by using SSL/TLS to encrypt the C&C channel
Employ allergy attack

- Make the information used by HIE and ISL unstable
  - For example, change MAC address, username, SID for every program invocation
  - Malware would not execute correctly successfully on the infected host
- Would affect a variety of legitimate software
- Success would depend on the willingness of users to accept security over usability
Conclusion

- Historically, malware has been “analysis environment aware”
- Recent developments (e.g., Flashback) show that malware can be “analysis environment oblivious”
  - Primitive DRM-like technologies can be matured (e.g., HIE and ISL)
- Future work must mitigate these protections or examine alternatives to threat detection and analysis
Please fill out your feedback forms.
Questions?