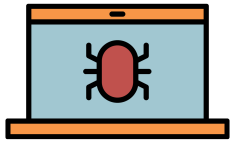
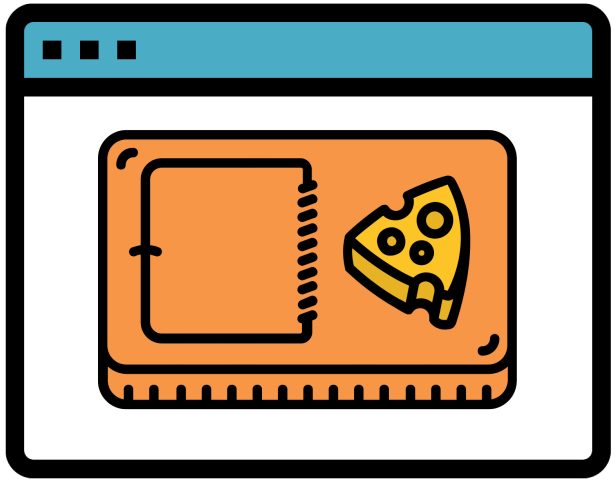


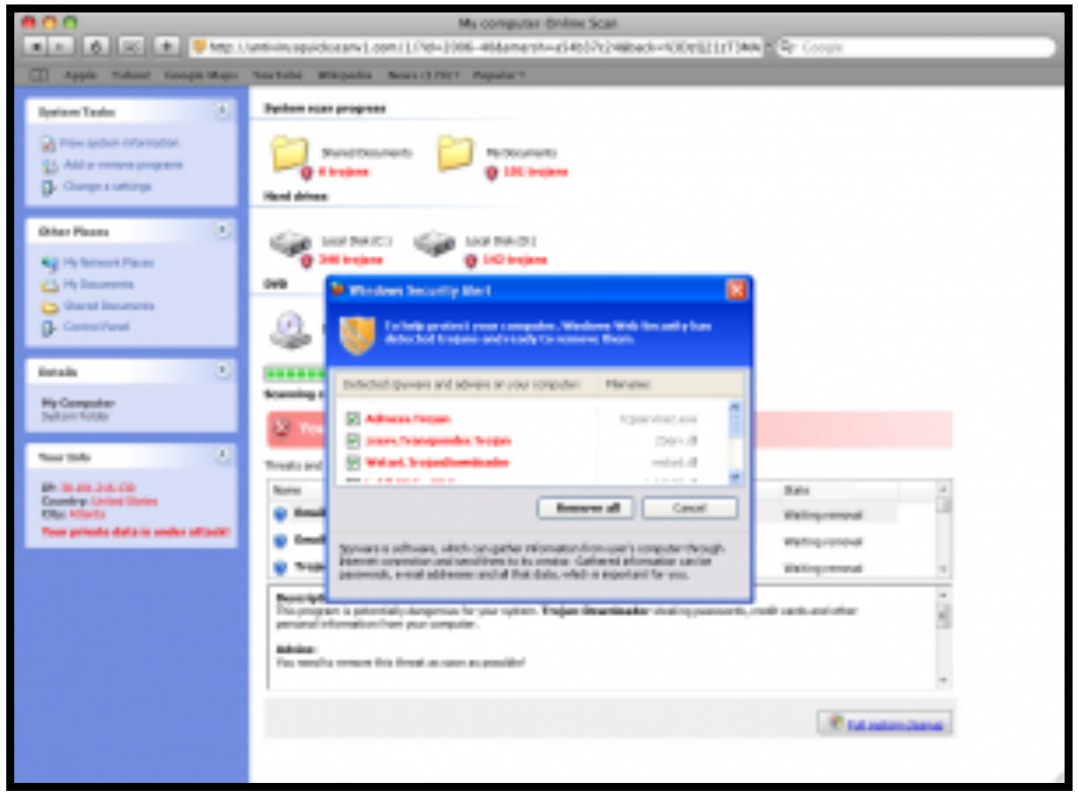
Malware Prevalence



"Safe" websites not safe to visit

- Reading USAToday.com results in malware on your computer

Malware Prevalence

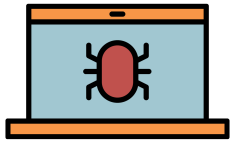


USAToday.com Malware

- What happened?

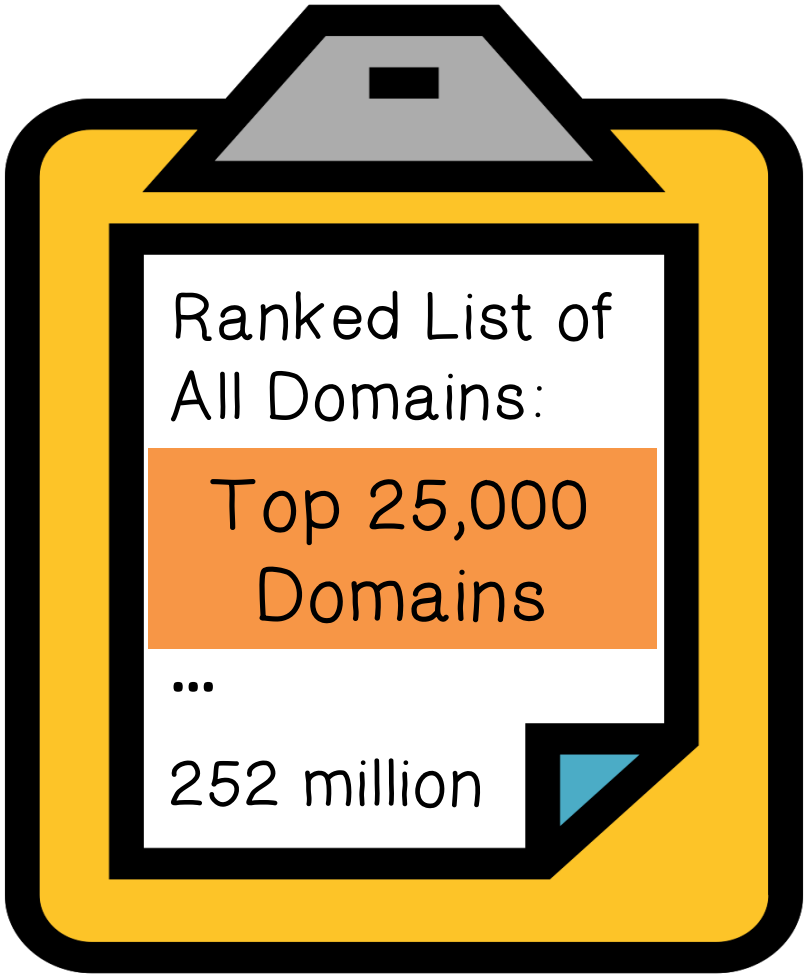
VIRUS TOTAL

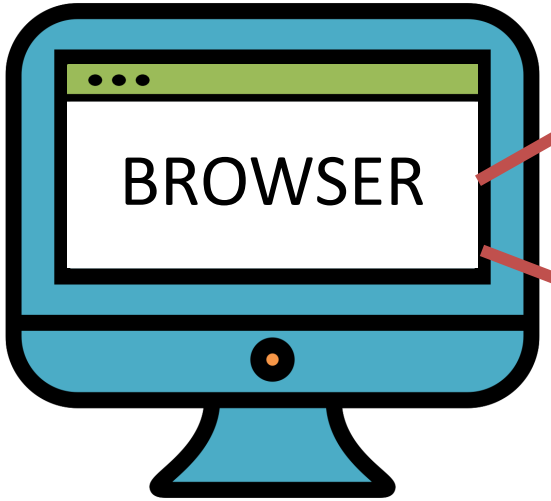
File **Install_2006-40.exe** received on **2009.05.07 18:04:01 (UTC)**
Current status: **finished**
Result: **1/40 (2.50%)**



Malware Prevalence

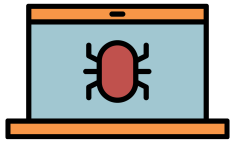
Case Study: Alexa



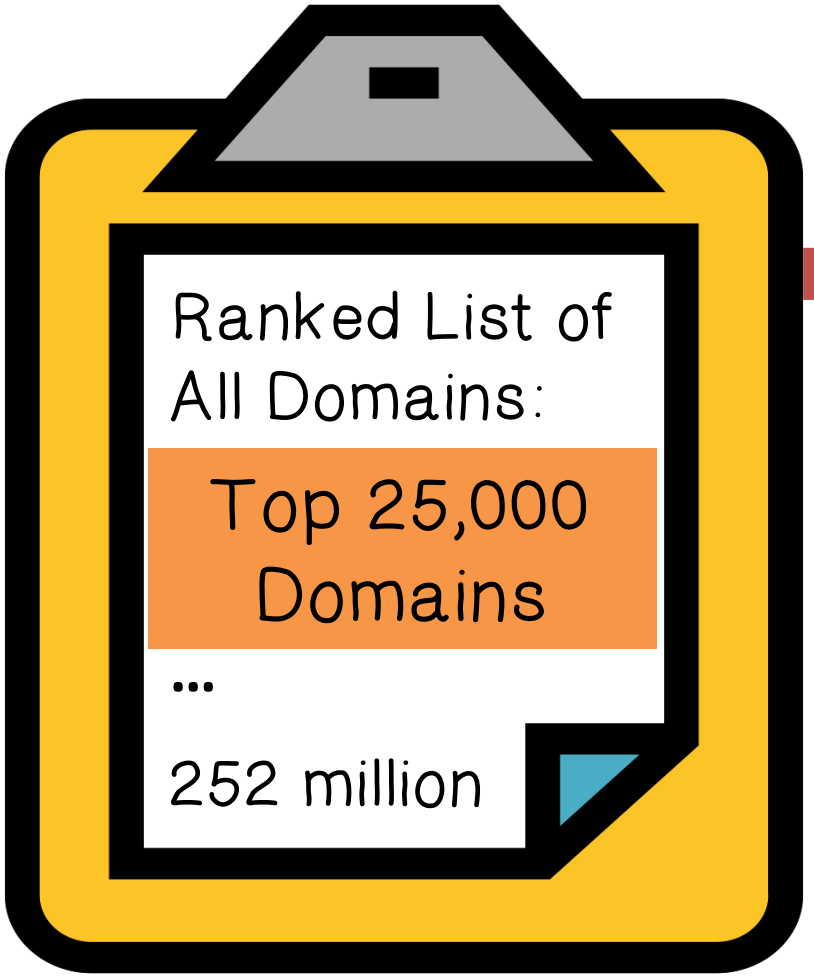


25,000 Domains Visited

Network Traffic Monitored



Malware Prevalence



Case Study: Alexa Results

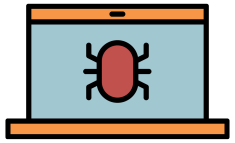
39 Domains resulted in Drive-by Downloads

87% of sites involved exploits for Java

46% of sites served exploits via ad networks

7.8M users served malicious content

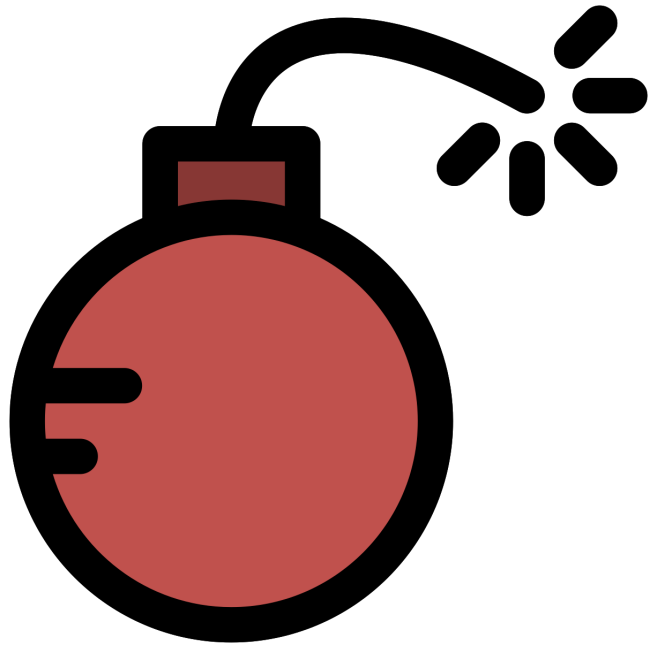
1.2M likely compromised

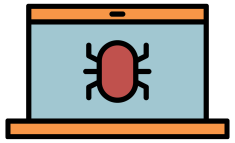


Malware Prevalence

New features a regular basis for attacks, e.g.

- PDF contains embedded, malicious Flash movie which exploits Acrobat Reader's flash interpreter
- Compromises the system
- Phones home to controller



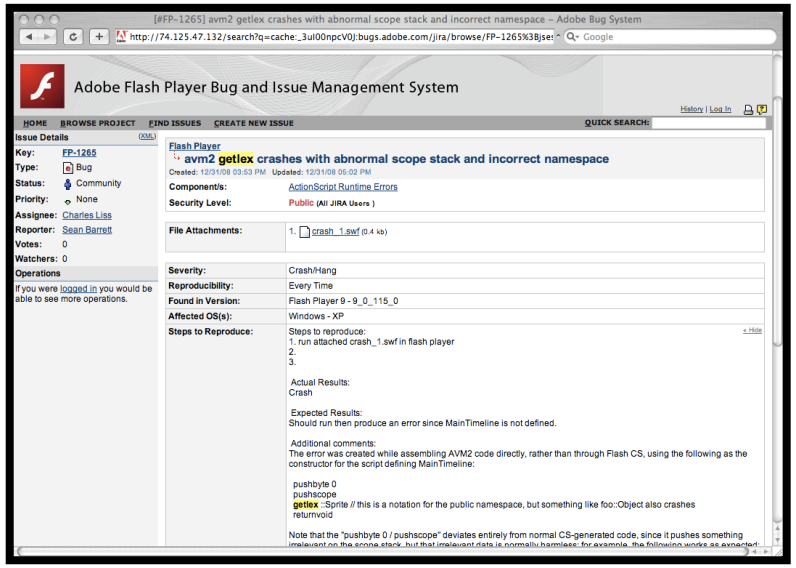


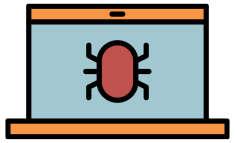
Malware Prevalence

New features a regular basis for attacks, e.g.

- Soon after, compromised, legitimate websites found hosting drive-by attacks that use the same flaw to exploit Flash Player
- Vulnerability traced back to bug reported to Adobe eight months prior

File **Install_2006-40.exe** received on **2009.05.07 18:04:01 (UTC)**
Current status: **finished**
Result: **1/40 (2.50%)**



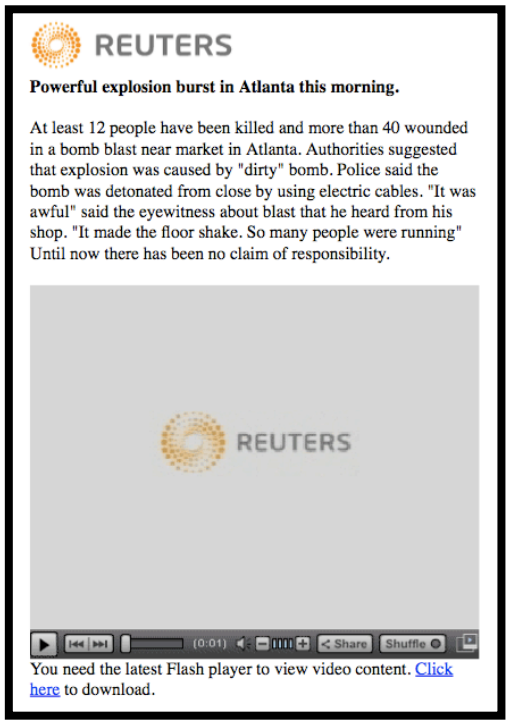


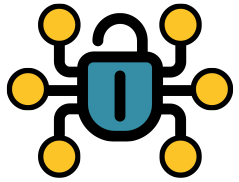
Malware Prevalence

Users remain uninformed (vulnerable)

Waledac's email campaigns

- Use of geo-location, temporally relevant events (e.g., bomb blast in <your city>, July 4th fireworks videos) to make attacks more compelling





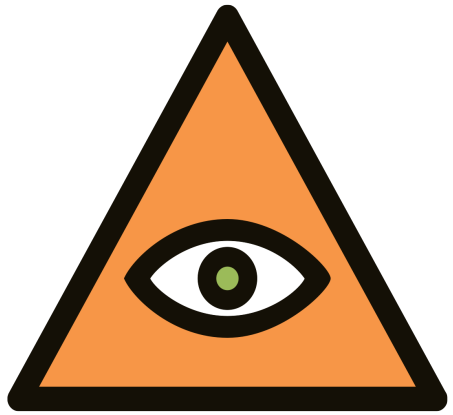
Malware Evolution

Traditional Defense-in-Depth

Network-Level Protection

Firewall

- Evaded by C&C protocol congruency

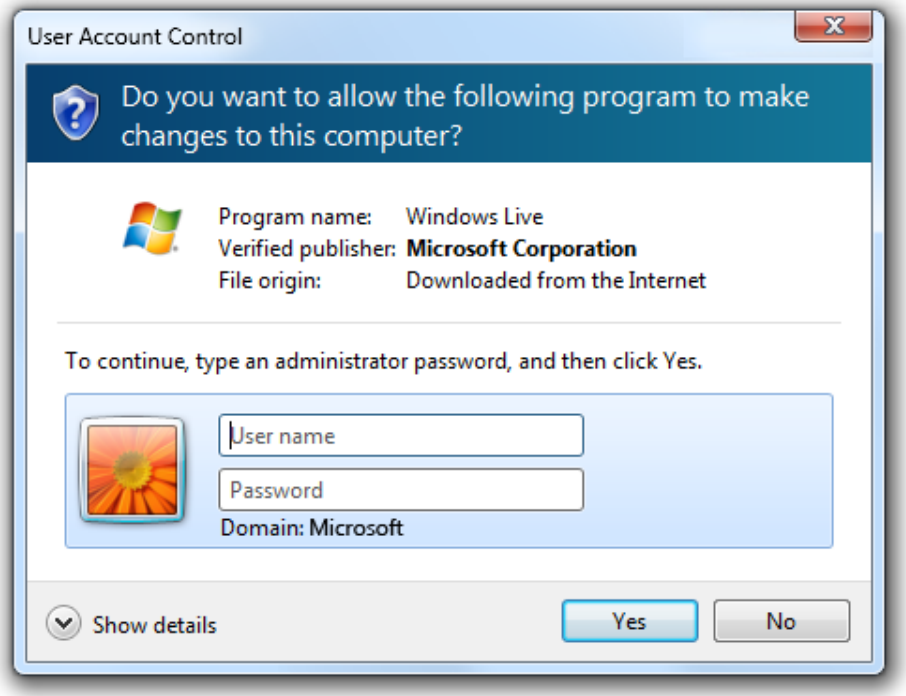


IPS/IDS

- Evaded by custom encodings

Malware Evolution

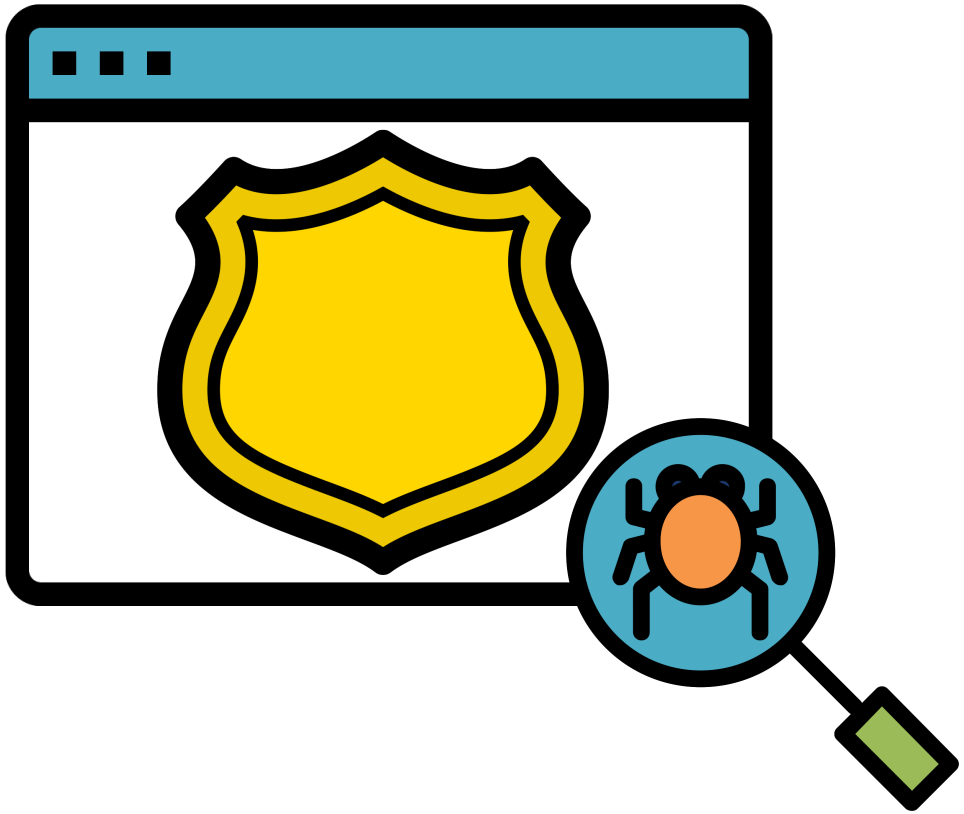
Host-Level Protection



User Access Control

- Analogous to "informed consent"

Malware Evolution

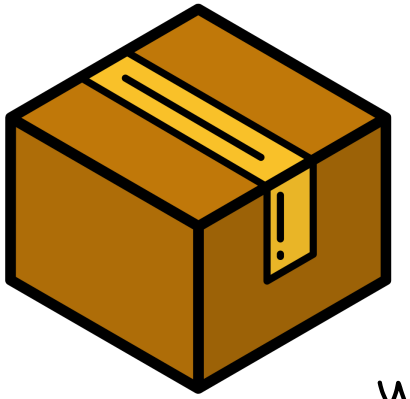


Antivirus

- Uses complex, heuristics-based detection along with signature matching”

Malware Obfuscation Quiz

Which of the following statements are true?



Definition of Packing: a technique whereby parts or all of an executable file are compressed, encrypted, or transformed in some fashion

- [x] Code that reverses the pre-runtime transformation is included in the executable.
- [ ] Each instance looks different, but there is a signature or pattern across all instances.
- [x] A signature scanner that tries to identify malware by its unique strings would not be effective.

Malware Obfuscation: Packing

Machine Code

Transformed Machine Code (appears as data)

Push EBP
MOV EBP, ESP
SUBESP,8
CALL 00401170
…

Encrypt/ Compress/ Transform

<Unpack Code>
♏♋♦●❑❑♓
♏♒♓●●♓❑
💣❑♌♏❑◆
♏♓♐♋♎♎
…

Program A

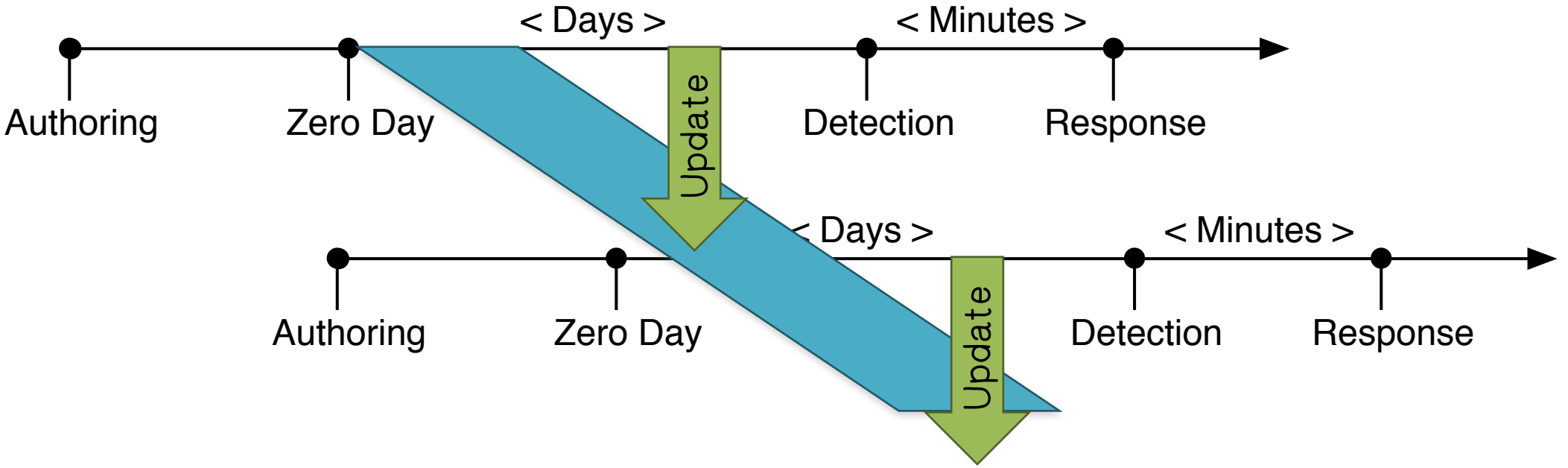
Packing Tool

Program A'

Malware Obfuscation

Server Side Polymorphism

- Attacks the heart of the traditional host-based AV model by automating mutations



Malware Obfuscation

When done professionally: Waledac

Collected on 12/30/2008

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

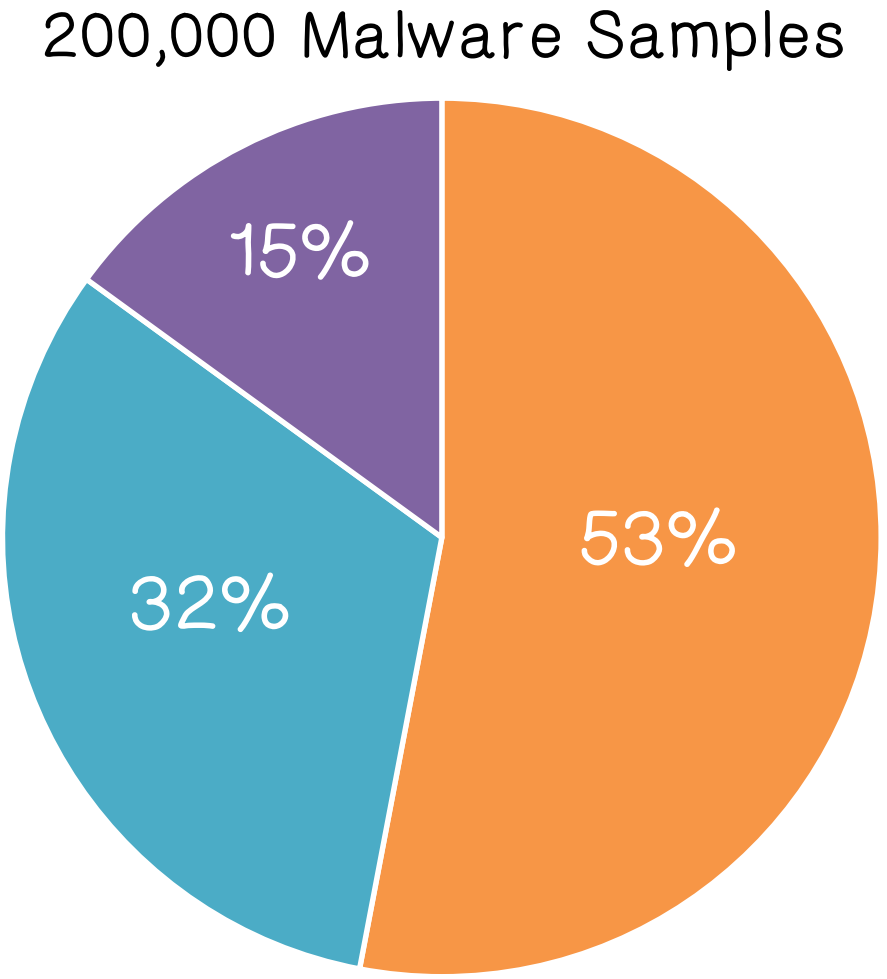
Collected on 2/25/2009

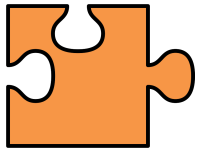
File **disc.exe** received on **02.25.2009 21:53:13 (CET)**
Current status: **finished**
Result: **11/39 (28.21%)**

Impact on Antivirus Software

Surveyed McAfee AV using 200,000 malware samples collected over six months

- 53% detected on first day
- 32% detected with delay (approx. 54 days)
- 15% still not detected 6 months later





Infrastructure Quiz

Given the following obfuscation techniques, which hiding from Users or security or researchers?

Techniques:

1. Users
2. Security
3. Researchers

Descriptions:

1. Rookits
2. Thoroughly mapping security sites and honey pots so as to avoid them
3. Using nonce-based encryption methods

Malware Analysis

Understanding malware behaviors

- Network and host level detection/blocking
- Forensics and asset Remediation
- Threat/trend analysis

Malware authors make analysis very challenging

Increasingly sophisticated techniques

Malware Analysis

Why Automation?

- DIY kits, packing tools, server-side polymorphism vastly increase volume of samples

There are hundreds of thousands of new instances each day

- Collected from crawlers, mail filters, honeypots, user submissions, and malware exchanges

Volume makes manual analysis untenable

Malware Analysis

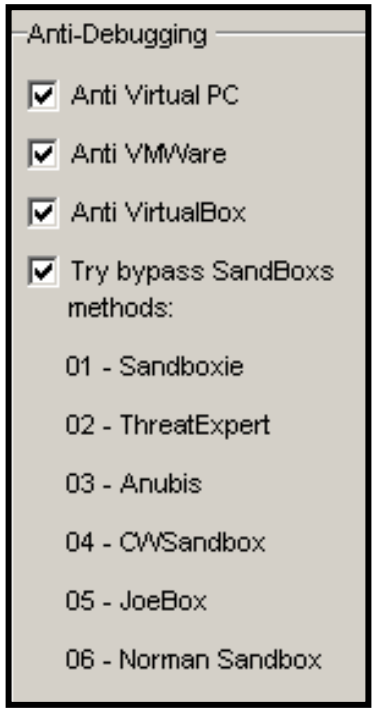
The Malware Uncertainty Principle

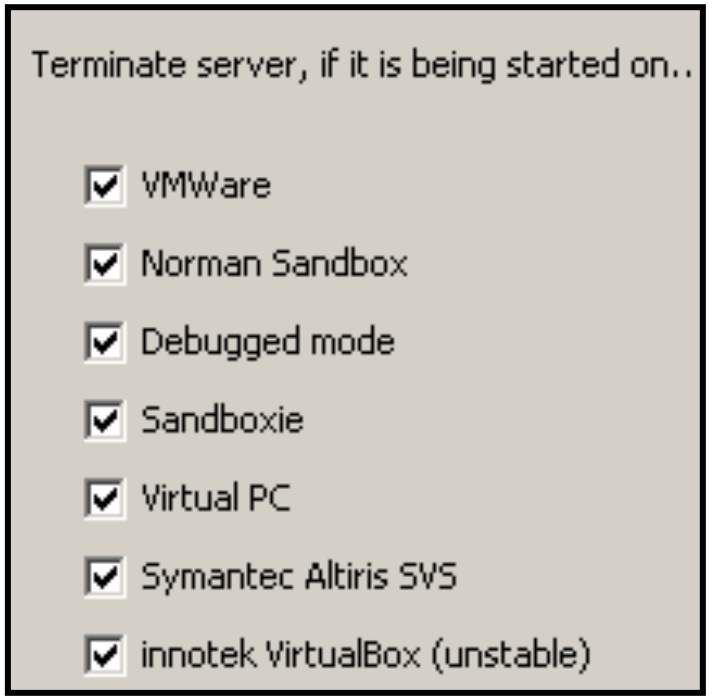
- Observer affecting the observed environment
- Robust and detailed analyzers are typically invasive
  - In-memory presence
  - Hooks
  - CPU Emulation
- Malware will refuse to run

Malware Analysis

The Malware Uncertainty Principle, Commercialized

- Dynamic analyzer detection is a standard malware feature





Malware Analysis

Transparency Requirements

- Higher Privilege
- No non-privileged side effects
- Same instruction execution semantics
- Identical exception handling
- Identical notion of time

Malware Analysis

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

Fulfilling the Requirements

Malware Analysis

Same Instruction Execution Semantics

State-of-the-art malware analyzers use emulation-based approaches

- There have been attacks that detect full system emulator approaches by exploiting incomplete emulation
- There is no way to guarantee the absence of such attacks
- Determining whether the languages of two Turing machines (L and L') are equal is known as the undecidable problem EQ_{TM}

Malware Analysis

Same Instruction Execution Semantics (Example of Attacks)

0xf30xf3...0x90

- 15 0xf3s before 0x90: execute no-op 15 times
- But the instruction (16 bytes) is illegal because max x86 instruction length is 15 bytes
  - In bare metal, "illegal instruction" exception
  - In QEMU, no exception

Identical Notion of Time

Network-based timing measurements

- Impossible to identify all
- Equivalent to the problem of detecting and removing all covert channels

Equivalent to the problem of detecting and removing all covert channels

- Undecidable

Analysis Difficulty Quiz

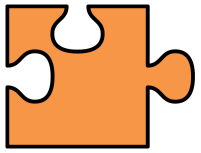
Rank the four categories from easiest (1) to hardest (4):

2 Static Properties Analysis: examine the static properties of the malware. Static properties include: metadata, strings embedded in the malware, header details, etc.

4 Manual Code Reversing: use a disassembler and decompiler to recreate the malware code.

1 Fully Automated Analysis: use fully automated tools to analyze the malware.

3 Interactive Behavior Analysis: running the malware in a protected and isolated environment.



Analysis Technique Results Quiz

Rank the four categories from the most information is obtained (1) to least information is obtained (4):

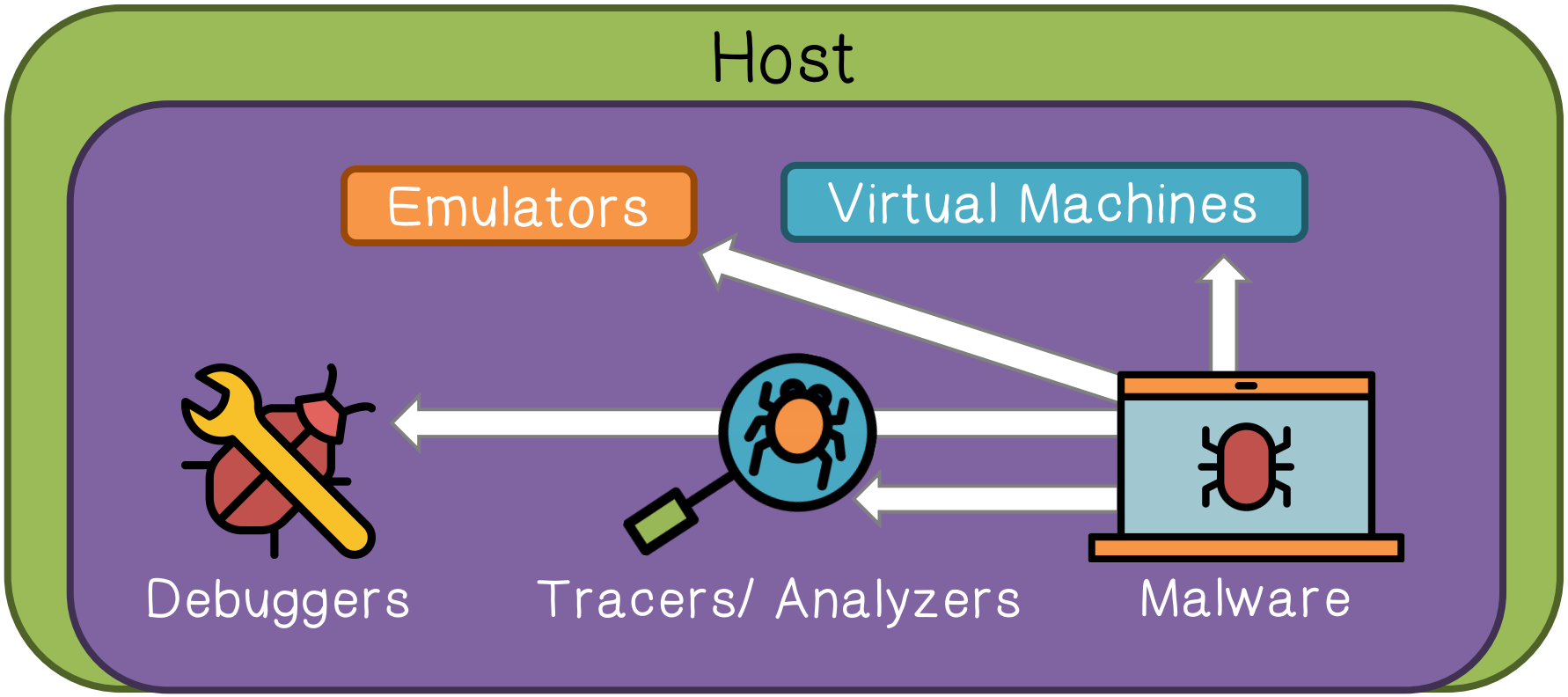
3 Static Properties Analysis: examine the static properties of the malware. Static properties include: metadata, strings embedded in the malware, header details, etc.

1 Manual Code Reversing: use a disassembler and decompiler to recreate the malware code.

4 Fully Automated Analysis: use fully automated tools to analyze the malware.

2 Interactive Behavior Analysis: running the malware in a protected and isolated environment.

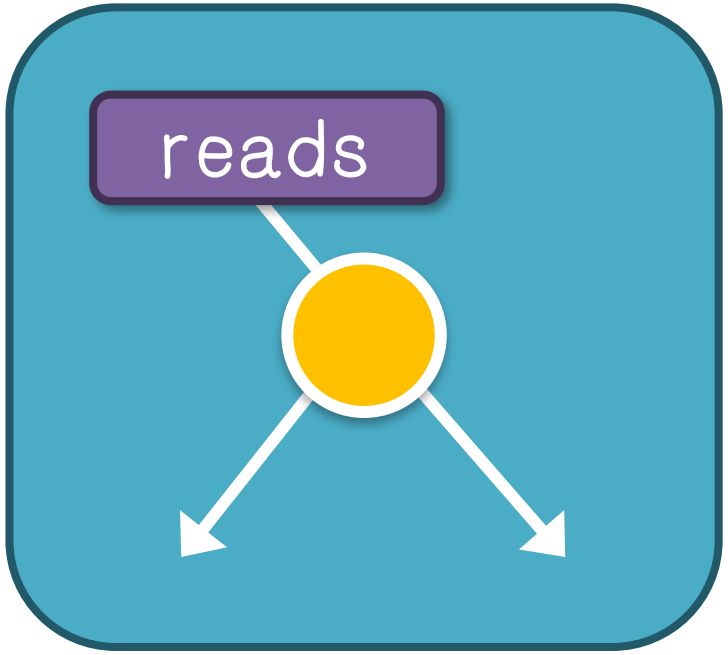
Robust and Efficient Malware Analysis



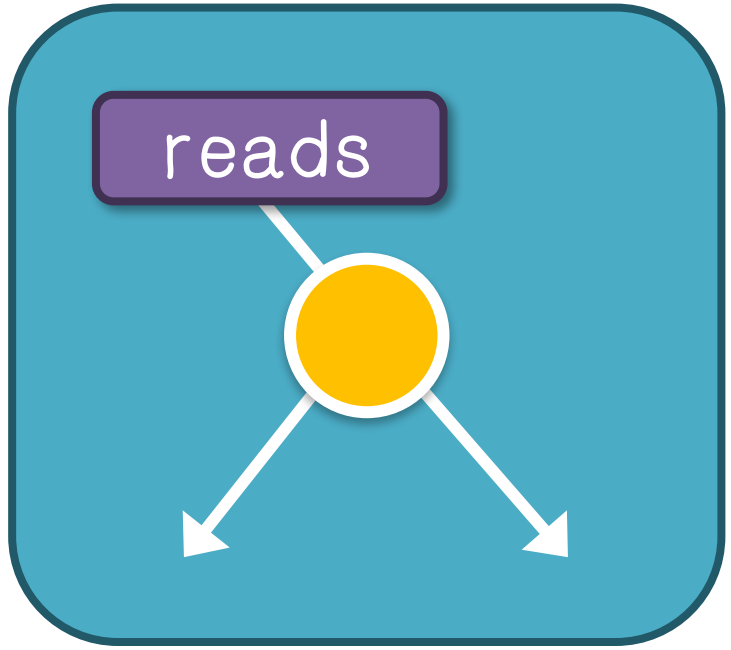
Analysis Environments

Malware may detect differences in analysis environment and evade analysis.

Transparent Malware Analysis



Execution under analysis



Execution in real system

Formal Requirements

- Higher Privilege
- No non-privileged side-effects
- Identical basic instruction execution semantics
- Transparent exception handling
- Identical measure of time

- Analyzer has to have higher privilege over malware
- Privilege levels can be enforced by hardware or virtual machine

Formal Requirements

- Higher Privilege
- **No non-privileged side-effects**
- Identical basic instruction execution semantics
- Transparent exception handling
- Identical measure of time

- Minimal side-effects should be introduced
- Analyzer should enforce privileged access to side-effects

Formal Requirements

- Higher Privilege
- No non-privileged side-effects
- **Identical basic instruction execution semantics**
- Transparent exception handling
- Identical measure of time

- Hardware semantics and emulation

Formal Requirements

- Higher Privilege
- No non-privileged side-effects
- Identical basic instruction execution semantics
- **Transparent exception handling**
- Identical measure of time

- Exception handling should be transparent or same

Formal Requirements

- Higher Privilege
- No non-privileged side-effects
- Identical basic instruction execution semantics
- Transparent exception handling
- Identical measure of time

- Timing for each instruction
- Timing for I/O, exception handling

Formal Requirements

- Higher Privilege
- No non-privileged side-effects
- Identical basic instruction execution semantics
- Transparent exception handling
- Identical measure of time

Ether Malware Analyzer

- Higher Privilege
- No non-privileged side-effects
- Identical basic instruction execution semantics
- Transparent exception handling
- Identical measure of time

Built using Intel VT (Hardware Virtualization)

- Hypervisor has higher privilege over kernel
- Several hardware supported traps – VM Exits

Ether Malware Analyzer

- Higher Privilege
- No non-privileged side-effects
- Identical basic instruction execution semantics
- Transparent exception handling
- Identical measure of time

- Analyzer is outside environment
- Minimal side-effects – trap flag, system call handling

Ether Malware Analyzer

- Higher Privilege
- No non-privileged side-effects
- Identical basic instruction execution semantics
- Transparent exception handling
- Identical measure of time

- Instruction execution semantics same

Ether Malware Analyzer

- Higher Privilege
- No non-privileged side-effects
- Identical basic instruction execution semantics
- Transparent exception handling
- Identical measure of time

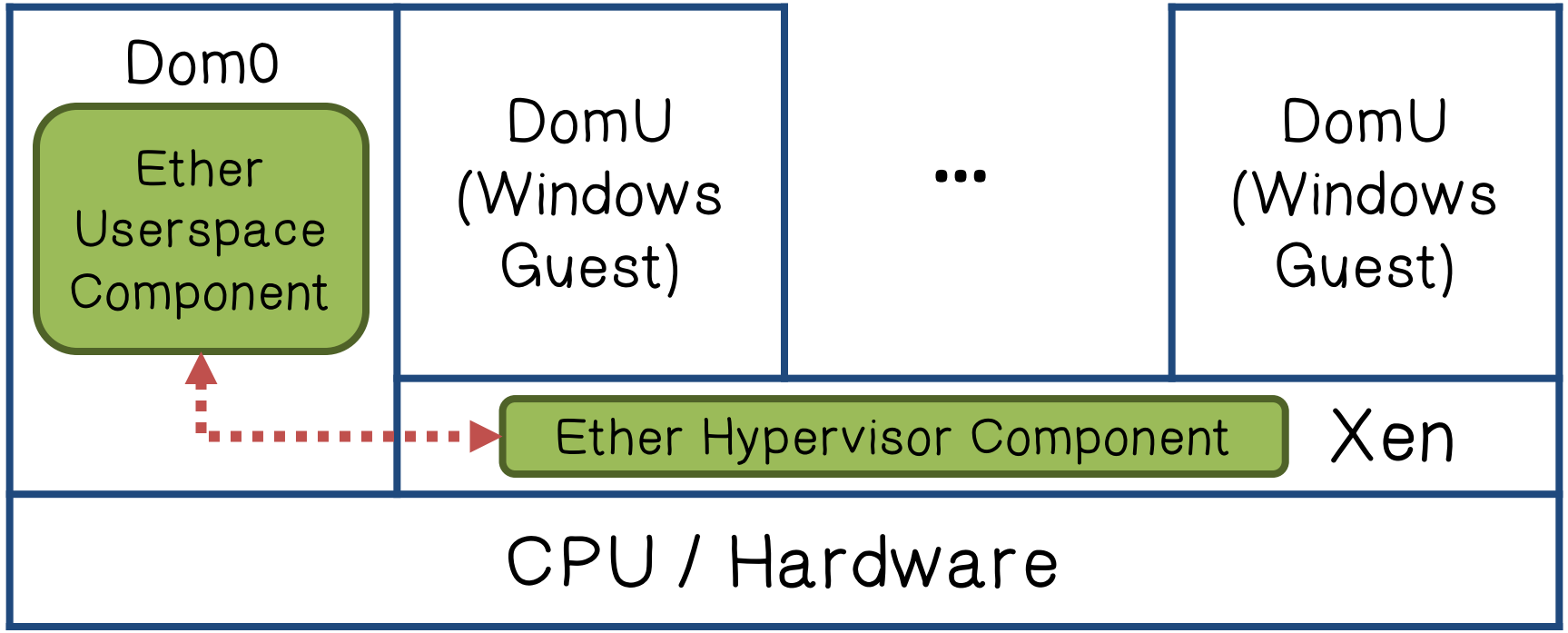
- Hypervisor pre-empts before OS exception handling

Ether Malware Analyzer

- Higher Privilege
- No non-privileged side-effects
- Identical basic instruction execution semantics
- Transparent exception handling
- Identical measure of time

- RDTSC

Ether Malware Analyzer: Implementation



Use Intel VT hardware virtualization extensions to provide instruction execution on actual hardware

Extend the Xen hypervisor to leverage Intel VT for malware analysis

Ether Malware Analyzer: Experiments

EtherUnpack: extracts hidden code from obfuscated malware

- We compared how well current tools extract hidden code by obfuscating a test binary and looking for a known string in the extracted code

EtherTrace: Records system calls executed by obfuscated malware

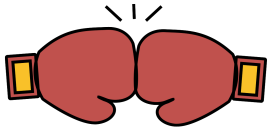
- We obfuscated a test binary which executes a set of known operations, and then observe if they were logged by the tool

Ether Malware: Unpack Results

Packing Tool	PolyUnpack	Renovo	EtherUnpack
Armadillo	no	no	yes
Aspack	no	yes	yes
Asprotect	yes	yes	yes
FSG	yes	yes	yes
MEW	yes	yes	yes
MoleBox	no	yes	yes
Morphine	yes	yes	yes
Obsidium	no	no	yes
PECompact	no	yes	yes
Themida	no	yes	yes
Themida VM	no	no	yes
UPX	yes	yes	yes
UPX Scrambled	yes	yes	yes
WinUPack	no	yes	yes
Yoda's Protector	no	yes	yes

Ether Malware: Ether Trace Results

Packing Tool	Norman Sandbox	Anubis	EtherTrace
None	yes	yes	yes
Armadillo	no	no	yes
UPX	yes	yes	yes
Upack	yes	yes	yes
Themida	yes	yes	yes
PECompact	yes	yes	yes
ASPack	yes	yes	yes
FSG	yes	yes	yes
ASProtect	yes	no	yes
WinUpack	yes	yes	yes
tElock	yes	no	yes
PKLITE32	yes	yes	yes
Yoda's Protector	no	yes	yes
NsPack	yes	yes	yes
MEW	yes	yes	yes
nPack	yes	yes	yes
RLPack	yes	yes	yes
RCryptor	yes	yes	yes



Malware Analysis vs. Obfuscations

Analysis

Static Analysis based approaches

Dynamic malware analysis

Dynamic multipath exploration
(Moser et al. 2007)

Bitscope *(Brumley et al. 2007)*

EXE *(Cadar et al. 2006)*

Forced execution *(Wilhelm et al. 2007)*

Obfuscations

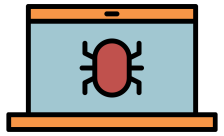
Polymorphism, metamorphism, packing, opaque predicates, anti-disassembly

Trigger-based behavior
(Logic bombs, time bombs, anti-debugging, anti-emulation, etc.)

Response

Response

The battle continues …

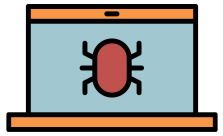


Malware Emulators

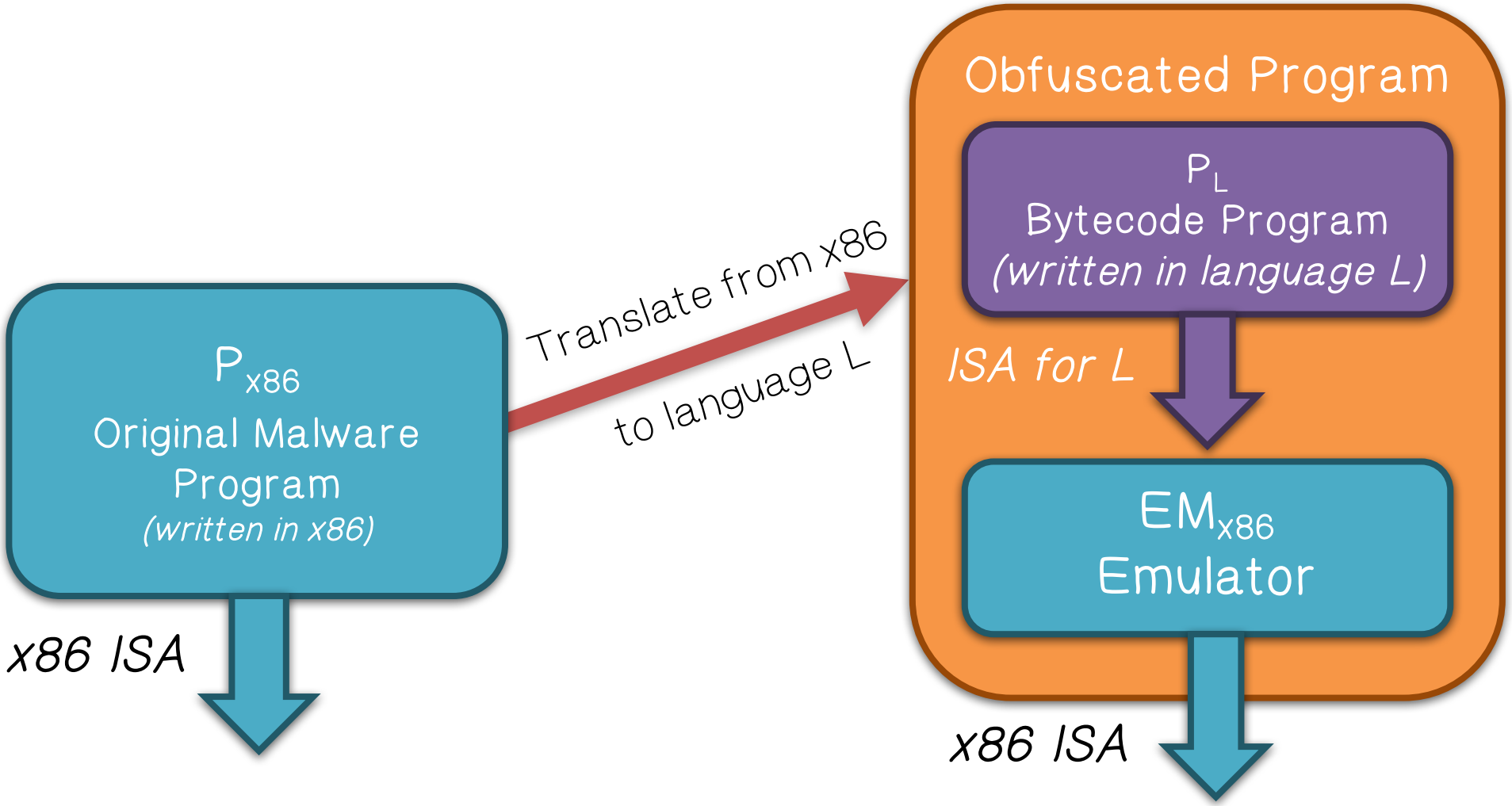
Recent move towards emulator-based obfuscation

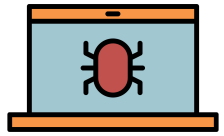
- An instruction-level obfuscation approach
- Several commercial packers support emulator based obfuscation, including Code Virtualizer and VMProtect

Emulation techniques maturing, widespread adoption is possible



Emulator-Based Obfuscation





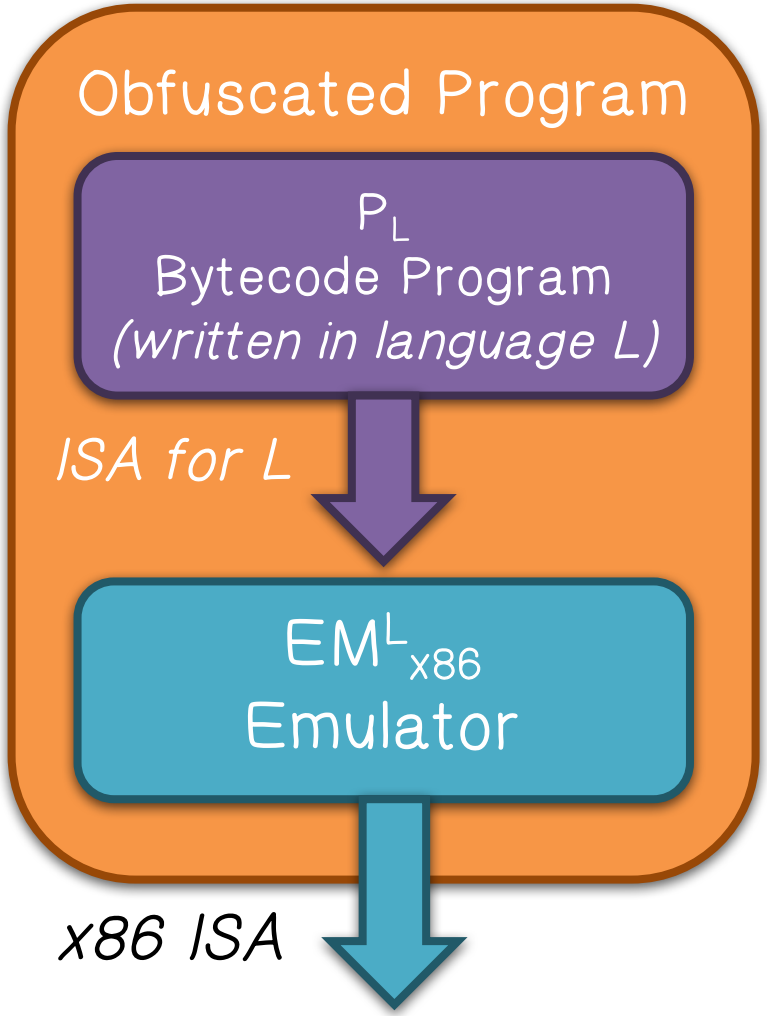
Impacts on Existing Malware Analysis

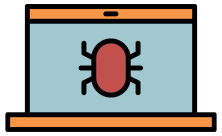
Unknown Language L

- L can be randomly generated

Pure Static Analysis *(whitebox)*

- Completely thwarted
- Only emulator code is analyzable
- P_L is considered as data by analyzer

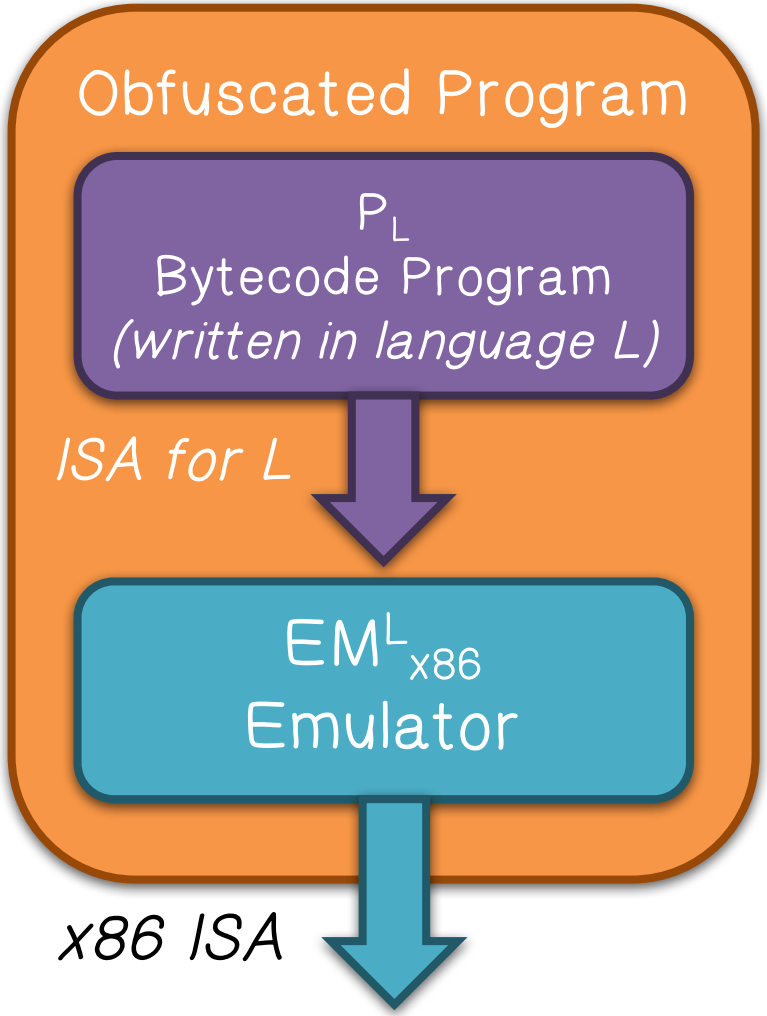


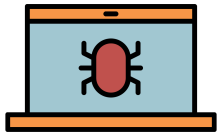


Impacts on Existing Malware Analysis

Greybox methods

- Includes instruction level analyzers, information-flow, dynamic tainting, multi-path exploration etc.
- Analysis is inaccurate
- For example, paths may explored in the emulator, but not the malware

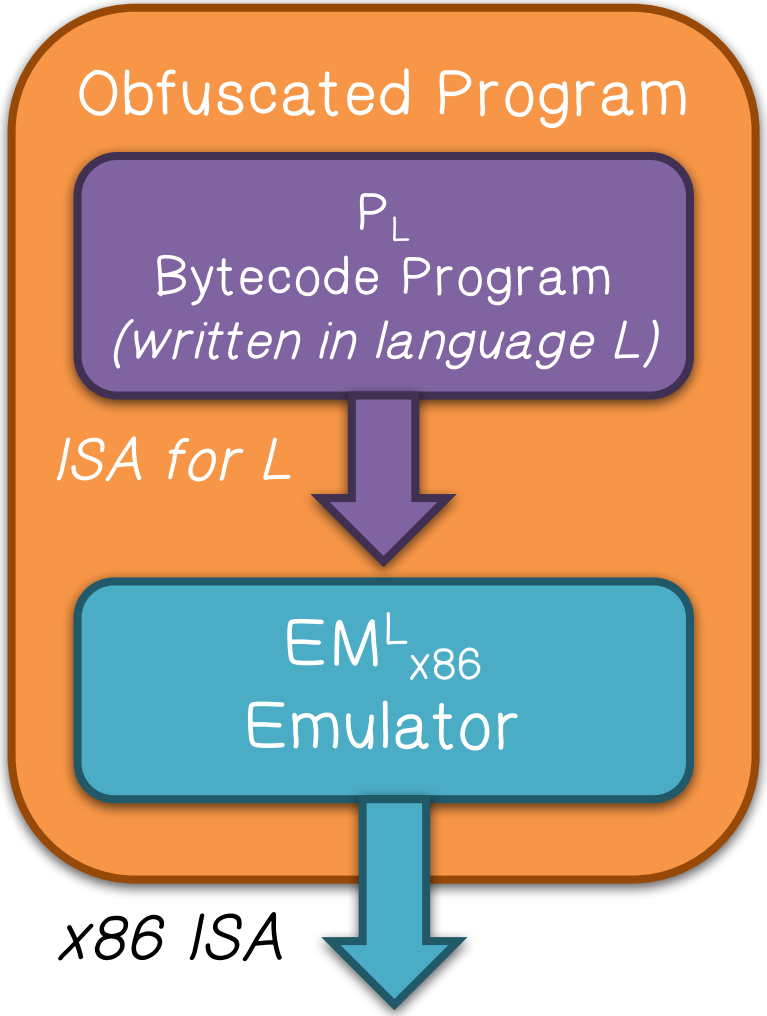


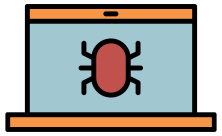


Impacts on Existing Malware Analysis

Manual reverse-engineering methods cannot scale

- Manual reverse-engineering takes time
- Each malware instance can have new bytecode language and emulator, making reverse-engineered information obsolete

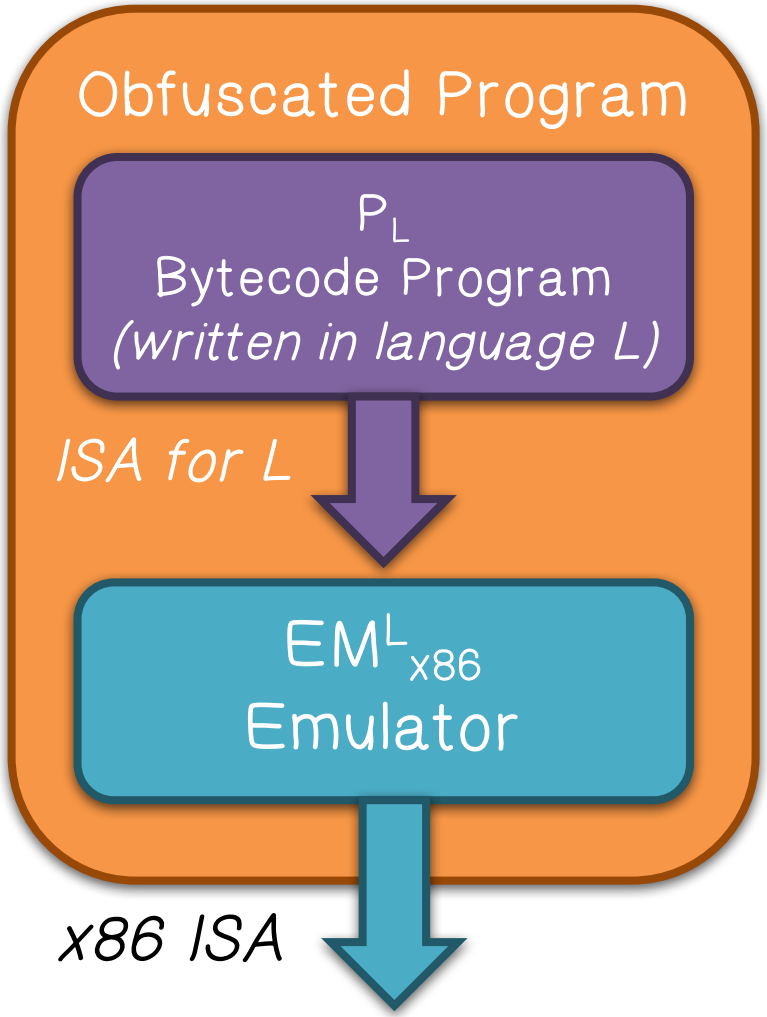


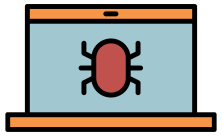


Impacts on Existing Malware Analysis

Need automated techniques to reverse emulator

- Should not require any knowledge about bytecode
- Should be generic and work for a large class of emulators

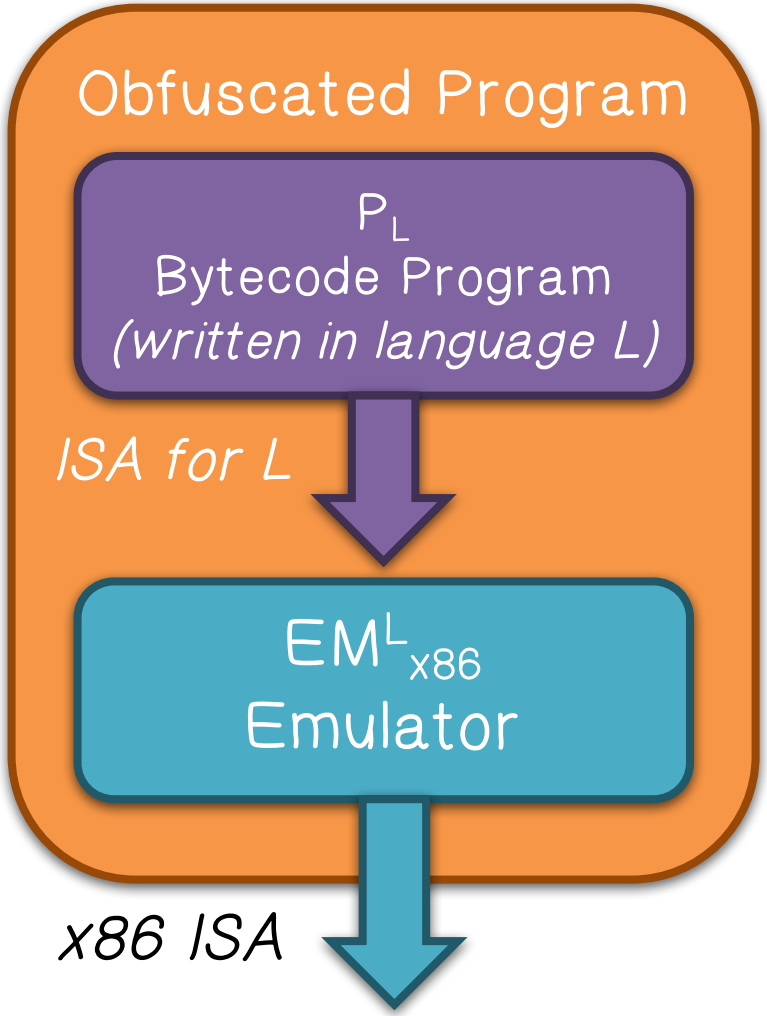




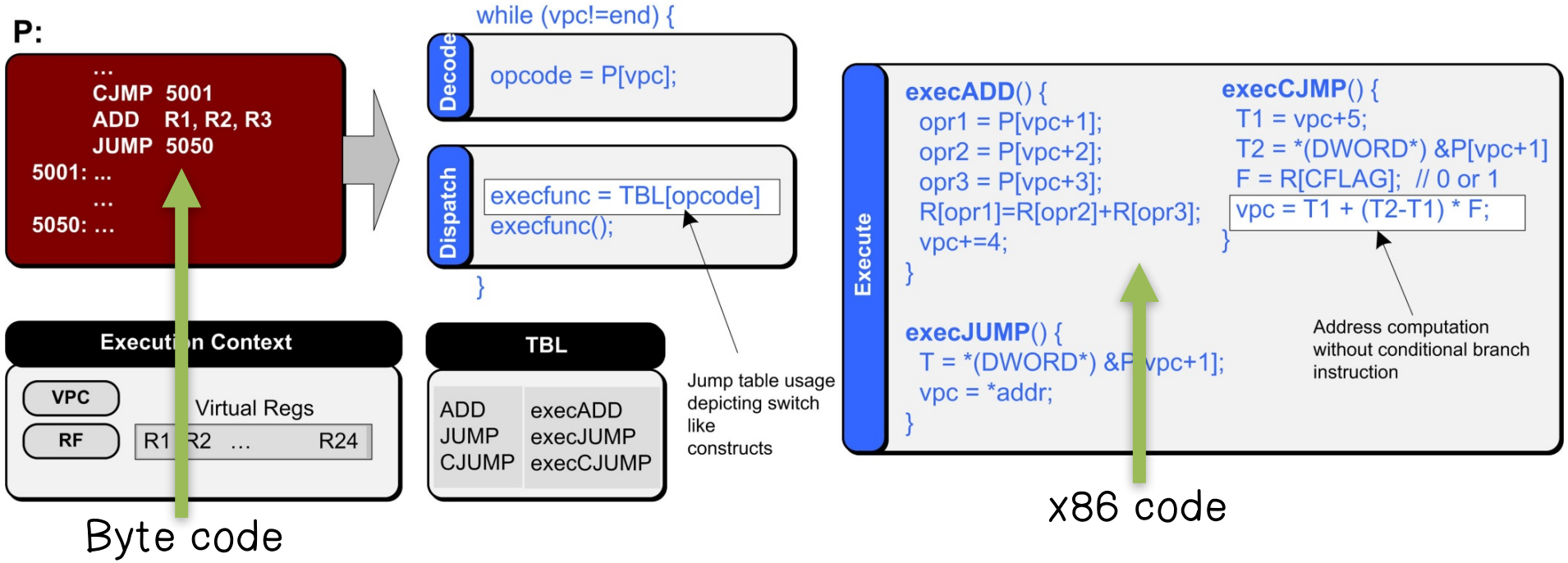
Impacts on Existing Malware Analysis

Is automated reverse engineering possible?

- Theoretically, it is an undecidable problem
- However, from intuition, the emulator's fetch-decode-execute behavior can be identified at runtime



Approach of Emulation



Decode-dispatch emulation utilizes a central loop to fetch, decode and execute the bytecode instructions

The VPC (Virtual Program Counter) is maintained to point to the next bytecode instruction to fetch from

Challenges of Reverse Engineering

No knowledge of bytecode program

- The location of the bytecode program in the obfuscated program's memory is not known

No knowledge of emulator's code

- The code that corresponds to decode, dispatch and execute phases of the emulator is not known

Challenges of Reverse Engineering

Intentional variations

- Context can be maintained in many different ways
- An attacker may complicate VPC identification by maintaining it in different correlated variables
- Bytecode program may be stored in non-contiguous memory

Approach Overview

Abstract Variable Binding

- Identify pointer variables within raw memory of emulator using access patterns of memory reads and writes
- It is a combination of forward and backward data-flow analysis

Identifying Candidate VPCs

- Cluster memory reads according to their bound variables
- Each cluster provides a candidate VPC

Identify Emulator Phases

- Identify decode-dispatch loop and emulator phases

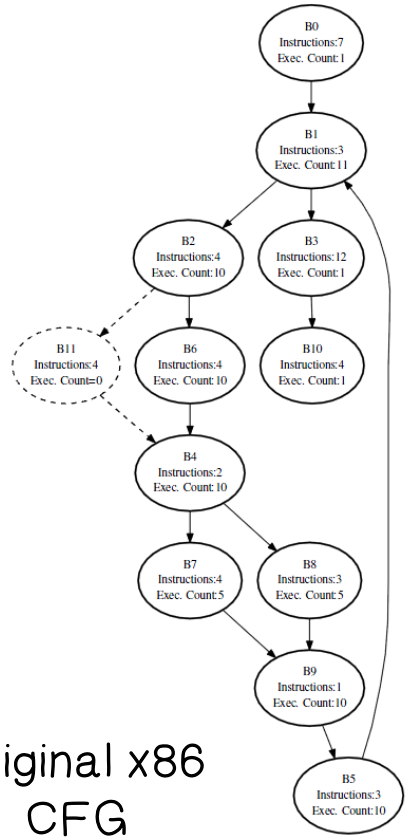
Approach Overview

Extract Bytecode Syntax and Semantics

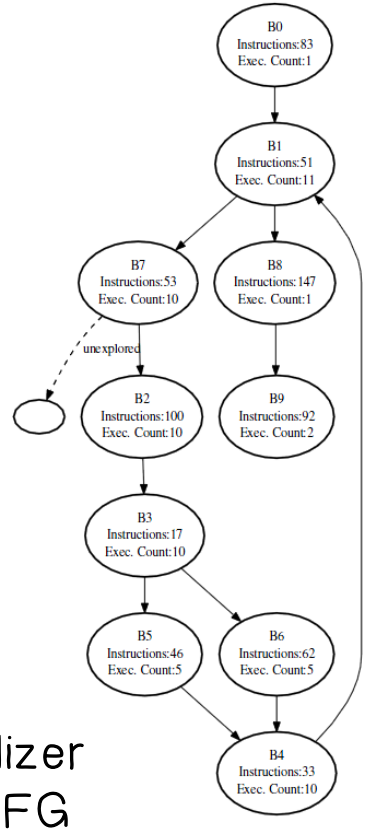
- Identify fetched instruction syntax – opcode and operands
- Identify execute routine related to instruction – this is the semantics of bytecode instruction
- This information is subsequently used to generate CFGs

Experimental Evaluation

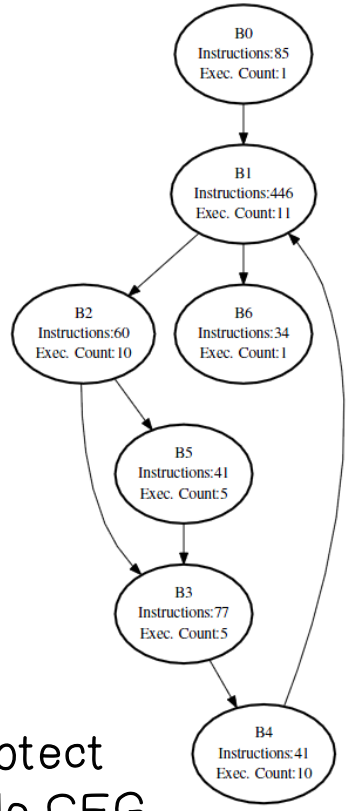
Bytecode syntax and semantics identification evaluation (synthetic program)



Original x86 CFG



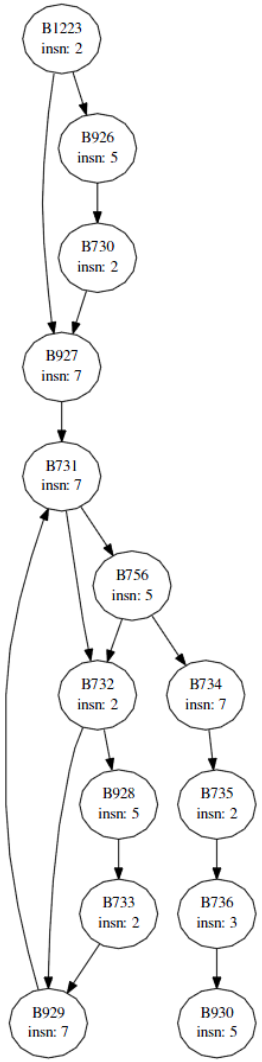
Code Virtualizer Bytecode CFG



VMProtect Bytecode CFG

Experimental Results

Unpacked real program experiments
(a function of NOTEPAD.EXE)



Original x86
CFG

Extracted
Bytecode CFG
(VMProtect)