Virus Scanning as Model Checking

Mihai Christodorescu
mihai@cs.wisc.edu

University of Wisconsin, Madison
Overview

1. The Problem:
   - Virus writers are getting smarter!

2. Smart Virus Scanner
   - Model checking

3. Encouraging Results

4. Future Directions
Why Another Virus Scanner?

• The Problem:
  - Viruses are becoming better at hiding themselves in binaries
  - Virus writers use complex techniques to obfuscate virus code in a host program

• Current commercial virus scanners are inadequate
Obfuscation: **Vanilla Virus**

- Simple obfuscation methods

=> Easy detection:
- Signature matching
- Very successful against first-gen viruses!
Obfuscation: Polymorphism

- Encrypted virus body + morphed decryption routine

Detection methods:
- Heuristic detection
- Emulation
- Current state-of-the-art
Obfuscation: **Metamorphism**

- **Metamorphic viruses:**
  - Morph the whole virus body

=> Detection methods

Jan. 14, 2002
Mihai Christodorescu - mihai@cs.wisc.edu
Obfuscation: Code Integration

• Integration of virus and program
  - e.g. Mistfall Virus Engine

<table>
<thead>
<tr>
<th>Infected Program</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

\[ \Rightarrow \] Detection methods

?
Example

Virus Code
(from Chernobyl CIH 1.4):

<table>
<thead>
<tr>
<th>Loop:</th>
</tr>
</thead>
<tbody>
<tr>
<td>pop ecx</td>
</tr>
<tr>
<td>jecxz SFModMark</td>
</tr>
<tr>
<td>mov esi, ecx</td>
</tr>
<tr>
<td>mov eax, 0d601h</td>
</tr>
<tr>
<td>pop edx</td>
</tr>
<tr>
<td>pop ecx</td>
</tr>
<tr>
<td>call edi</td>
</tr>
<tr>
<td>jmp Loop</td>
</tr>
</tbody>
</table>

Morphed Virus Code
(from Chernobyl CIH 1.4):

<table>
<thead>
<tr>
<th>Loop:</th>
</tr>
</thead>
<tbody>
<tr>
<td>pop ecx</td>
</tr>
<tr>
<td>jecxz SFModMark</td>
</tr>
<tr>
<td>mov esi, ecx</td>
</tr>
<tr>
<td>mov eax, 0d601h</td>
</tr>
<tr>
<td>pop edx</td>
</tr>
<tr>
<td>pop ecx</td>
</tr>
<tr>
<td>pop ecx</td>
</tr>
<tr>
<td>call edi</td>
</tr>
<tr>
<td>jmp Loop</td>
</tr>
</tbody>
</table>
## Example

### Virus Code
*(from Chernobyl CIH 1.4):*

<table>
<thead>
<tr>
<th>Loop:</th>
</tr>
</thead>
<tbody>
<tr>
<td>pop ecx</td>
</tr>
<tr>
<td>jecxz SFModMark</td>
</tr>
<tr>
<td>mov esi, ecx</td>
</tr>
<tr>
<td>mov eax, 0d601h</td>
</tr>
<tr>
<td>pop edx</td>
</tr>
<tr>
<td>pop ecx</td>
</tr>
<tr>
<td>call edi</td>
</tr>
<tr>
<td>jmp Loop</td>
</tr>
</tbody>
</table>

### Morphed Virus Code
*(from Chernobyl CIH 1.4):*

<table>
<thead>
<tr>
<th>Loop:</th>
</tr>
</thead>
<tbody>
<tr>
<td>pop ecx</td>
</tr>
<tr>
<td>nop</td>
</tr>
<tr>
<td>jecxz SFModMark</td>
</tr>
<tr>
<td>xor ebx, ebx</td>
</tr>
<tr>
<td>beqz N1</td>
</tr>
<tr>
<td>N1:</td>
</tr>
<tr>
<td>mov esi, ecx</td>
</tr>
<tr>
<td>nop</td>
</tr>
<tr>
<td>mov eax, 0d601h</td>
</tr>
<tr>
<td>pop edx</td>
</tr>
<tr>
<td>pop ecx</td>
</tr>
<tr>
<td>nop</td>
</tr>
<tr>
<td>call edi</td>
</tr>
<tr>
<td>xor ebx, ebx</td>
</tr>
<tr>
<td>beqz N2</td>
</tr>
<tr>
<td>N2:</td>
</tr>
<tr>
<td>jmp Loop</td>
</tr>
</tbody>
</table>
Example

**Virus Code**
(from Chernobyl CIH 1.4):

```
Loop:
    pop     ecx
    jecxz   SFModMark
    mov     esi, ecx
    mov     eax, 0d601h
    pop     edx
    pop     ecx
    call    edi
    jmp     Loop
```

**Morphed Virus Code**
(from Chernobyl CIH 1.4):

```
Loop:
    pop     ecx
    nop
    jecxz   SFModMark
    xor     ebx, ebx
    beqz    N1
N1:  mov     esi, ecx
    nop
    mov     eax, 0d601h
    pop     edx
    pop     ecx
    pop     ecx
    pop     ecx
    call    edi
    xor     ebx, ebx
    beqz    N2
N2:  jmp     Loop
```
Example

Virus Code
(from Chernobyl CIH 1.4):

```
Loop:
    pop    ecx
    jecxz  SFModMark
    mov    esi, ecx
    mov    eax, 0d601h
    pop    edx
    pop    ecx
    call   edi
    jmp    Loop
```

Morphed Virus Code
(from Chernobyl CIH 1.4):

```
Loop:
    pop    ecx
    nop
    call   edi
    xor    ebx, ebx
    beqz   N2

N2:
    jmp    Loop
    nop
    mov    eax, 0d601h
    pop    edx
    pop    ecx
    nop
    jecxz  SFModMark
    xor    ebx, ebx
    beqz   N1

N1:
    mov    esi, ecx
```
**Example**

### Virus Code
*from Chernobyl CIH 1.4:*

<table>
<thead>
<tr>
<th>Loop:</th>
</tr>
</thead>
<tbody>
<tr>
<td>pop ecx</td>
</tr>
<tr>
<td>jecxz SFModMark</td>
</tr>
<tr>
<td>mov esi, ecx</td>
</tr>
<tr>
<td>mov eax, 0d601h</td>
</tr>
<tr>
<td>pop edx</td>
</tr>
<tr>
<td>pop ecx</td>
</tr>
<tr>
<td>call edi</td>
</tr>
<tr>
<td>jmp Loop</td>
</tr>
</tbody>
</table>

### Morphed Virus Code
*from Chernobyl CIH 1.4:*

<table>
<thead>
<tr>
<th>Loop:</th>
</tr>
</thead>
<tbody>
<tr>
<td>pop ecx</td>
</tr>
<tr>
<td>nop</td>
</tr>
<tr>
<td>jmp L1</td>
</tr>
<tr>
<td>L3: call edi</td>
</tr>
<tr>
<td>xor ebx, ebx</td>
</tr>
<tr>
<td>beqz N2</td>
</tr>
<tr>
<td>N2: jmp Loop</td>
</tr>
<tr>
<td>L2: nop</td>
</tr>
<tr>
<td>mov eax, 0d601h</td>
</tr>
<tr>
<td>pop edx</td>
</tr>
<tr>
<td>pop ecx</td>
</tr>
<tr>
<td>nop</td>
</tr>
<tr>
<td>jmp L3</td>
</tr>
<tr>
<td>L1: jecxz SFModMark</td>
</tr>
<tr>
<td>xor ebx, ebx</td>
</tr>
<tr>
<td>beqz N1</td>
</tr>
<tr>
<td>N1: mov esi, ecx</td>
</tr>
<tr>
<td>L4: jmp L2</td>
</tr>
</tbody>
</table>
Example

Virus Code
(from Chernobyl CIH 1.4):

Loop:
- pop ecx
- jecxz SFModMark
- mov esi, ecx
- mov eax, 0d601h
- pop edx
- pop ecx
- call edi
- jmp Loop

Morphed Virus Code
(from Chernobyl CIH 1.4):

Loop:
- pop ecx
- nop
- jmp L1
L3:
- call edi
- xor ebx, ebx
- beqz N2
L2:
- nop
- mov eax, 0d601h
- pop edx
- pop ecx
- nop
- jmp L3
L1:
- jecxz SFModMark
- xor ebx, ebx
- beqz N1
N1:
- mov esi, ecx
L4:
- jmp L2
What to do?

• Better virus detection tool
  - Analyze the program structure (instead of signature matching)
  - More flexible

• Check whether viral properties are present in a given program
  - e.g.: “program writes to an executable file”
  - e.g.: “program monitors as executables are loaded into memory and changes them”
  - e.g.: “program behaves just like virus XYZ”
Overview

1. The Problem:
   - Virus writers are getting smarter!

☞ 2. Smart Virus Scanner
   - Model checking

3. Encouraging Results

4. Future Directions
Use Model Checking

• Consider the vanilla virus code as a set of one or more properties

• Check that the program exhibits those properties
  - If YES => infected
Model Checking

• Technique for checking program properties

1. Build automaton for the desired property
2. Extract program model
3. Compare the model against the automaton
Model Checking

Vanilla Virus

- Property P
- Program A

Automaton for P

Model of A

Match?

Yes/No
Model Checking Binaries

Binary → Disassemble → Assembly Code → Analyze → Abstract Representation → Verify

Security Automaton for P

OK/Failed

IDA Pro (+ SDK plugins)

Transducer (on top of CodeSurfer static analysis engine)

Model Checker
Smart Virus Scanner

1. Build automaton from vanilla virus
   - Blueprint of virus behavior

2. Build a model of the program

3. Verify that model does not match the blueprint
Smart Virus Scanner Example

Virus Code:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>push eax</td>
<td></td>
</tr>
<tr>
<td>sidt [esp-02h]</td>
<td></td>
</tr>
<tr>
<td>pop ebx</td>
<td></td>
</tr>
<tr>
<td>add ebx, HookNo * 08h + 04h</td>
<td></td>
</tr>
<tr>
<td>cli</td>
<td></td>
</tr>
<tr>
<td>mov ebp, [ebx]</td>
<td></td>
</tr>
<tr>
<td>mov bp, [ebx-04h]</td>
<td></td>
</tr>
<tr>
<td>lea esi, MyHook - @1[ecx]</td>
<td></td>
</tr>
<tr>
<td>push esi</td>
<td></td>
</tr>
<tr>
<td>mov [ebx-04h], si</td>
<td></td>
</tr>
<tr>
<td>shr esi, 16</td>
<td></td>
</tr>
<tr>
<td>mov [ebx+02h], si</td>
<td></td>
</tr>
<tr>
<td>pop esi</td>
<td></td>
</tr>
</tbody>
</table>

(from Chernobyl CIH 1.4 virus)
Smart Virus Scanner Example

Virus Automaton:

Program to be checked:

```
mov ebp, [ebx]
mov bp, [ebx - 04h]
test ebx beqz next
next: lea esi, MyHook - @1[ecx]
```
Smart Virus Scanner

• What are irrelevant instructions?
  - NOPs
  - Control flow instructions that do not change the control flow
    • e.g.: jumps/branches to the next instructions
  - Instructions that modify dead registers
  - Sequences of instructions that do not modify architectural state
    • e.g.:
      `- add ebx, 1`
      `- sub ebx, 1`
Overview

1. The Problem:
   - Virus writers are getting smarter!

2. Smart Virus Scanner
   - Model checking

3. Encouraging Results

4. Future Directions
Current Status

• We disassemble and analyze program structure

• We can detect viruses morphed in a simple manner
  - Irrelevant instructions = NOPs
Results

- **Testing**
  - Viruses used: Chernobyl, Hare
  - AntiVirus utilities
    - Command AntiVirus (F-Prot)
    - Norton AntiVirus (Symantec)

- Not surprising!
  - Norton and Command AV do not detect "NOP"-morphed viruses

- Our Smart Virus Scanner catches "NOP"-morphed viruses
Current Status

• Limitations:
  - Intra-procedural only
  - Cannot detect equivalent instruction sequences

Procedure P

Not used!

Procedure Q
Overview

1. The Problem:
   - Virus writers are getting smarter!

2. Smart Virus Scanner
   - Model checking

3. Encouraging Results

4. Future Directions
Future Work

• **Main focus:**
  - Improve detection of “irrelevant insn’s”

• **More (precise) information from static analysis**
  - live range information
  - alias/points-to analysis

• Library of equivalent instructions sequences
Future Directions

• Context-sensitive model checking
  - Recognize virus code spread across subroutines

• Automata with uninterpreted symbols
  - Recognize virus code with different register usage

• Virus scanning for component-based systems
  - Recognize virus code distributed across components

• Scan for multiple viruses at the same time
References


• Zombie. Zombie’s Homepage. http://z0mbie.host.sk
More Model Checking

• Security Automaton
  - For policy “Always release a resource after acquiring it.”

![Security Automaton Diagram]
Even More Model Checking

• Abstract Representation

\[ x = \text{read\_input()} \]
\[ y = 0 \]
\[ z = 2 \times \pi \]
\[ \text{result} = \text{undefined} \]

\[ x > 0 ? \]

\[ \text{result} = \text{compute}(x, y, z) \]
\[ z = 0 \]

\[ \text{write\_output}(\text{result}) \]

\[ \text{read\_input()} \]

\[ \text{compute}(x, y, z) \]

\[ \text{write\_output(result)} \]