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?

Why Another Virus Scanner?

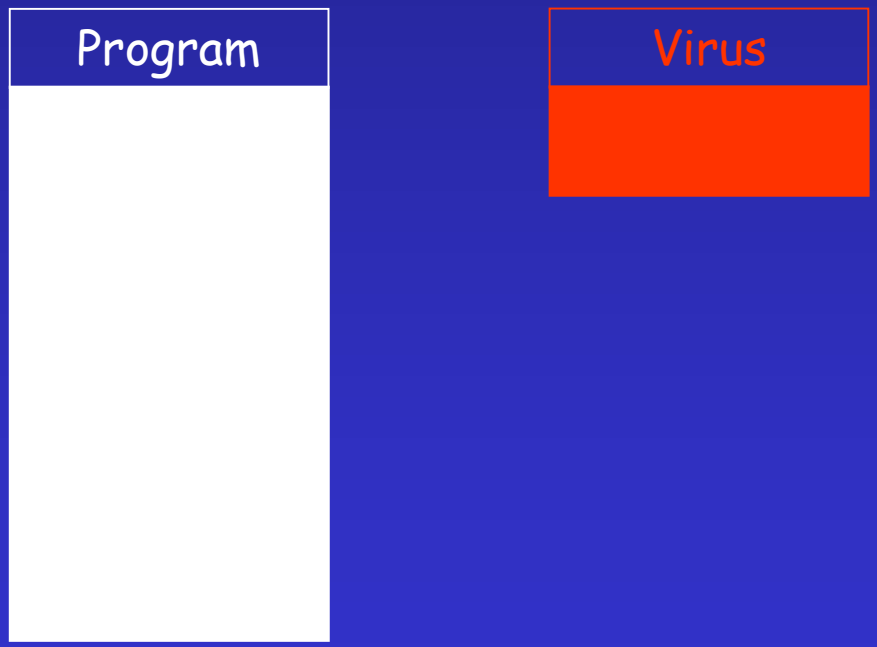
- The Problem:
  - Viruses are becoming better at hiding themselves in binaries
  - Virus writers use complex techniques to obfuscate virus code in a host program

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

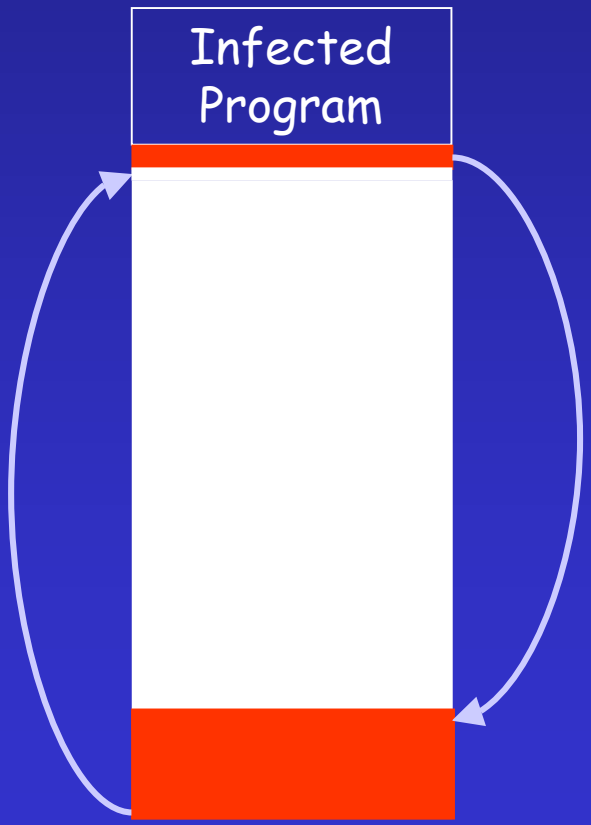


Obfuscation: Vanilla Virus

- Simple obfuscation methods

Obfuscation: Vanilla Virus

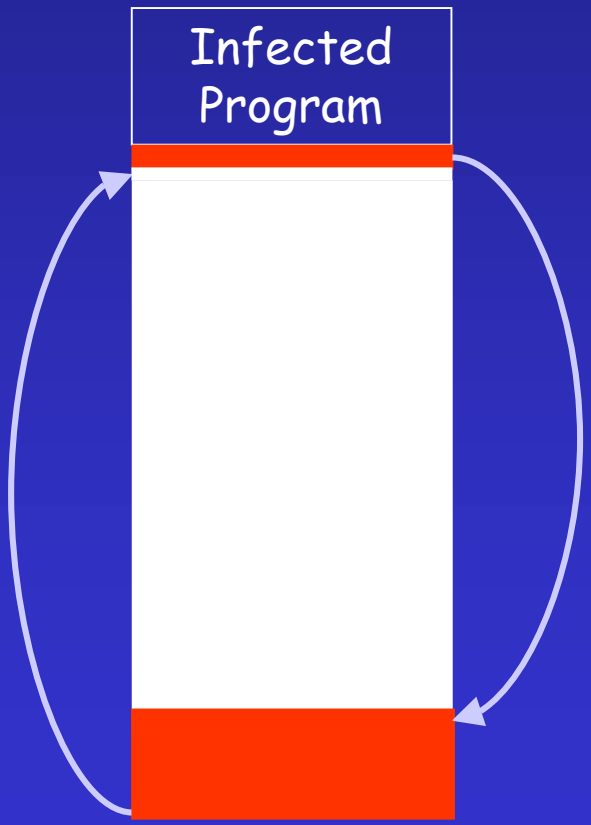
- Simple obfuscation methods



Obfuscation: Vanilla Virus

- Simple obfuscation methods

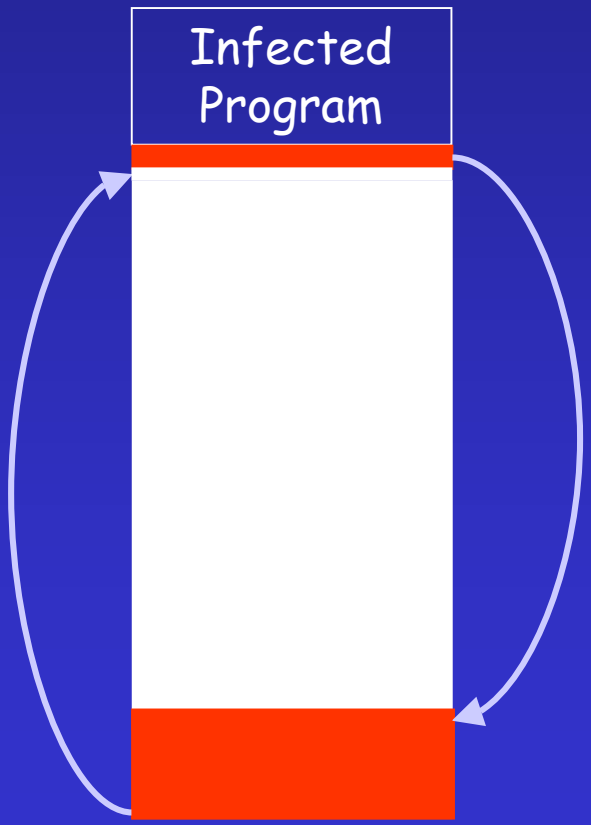
  => Easy detection:



Obfuscation: Vanilla Virus

- Simple obfuscation methods

  => Easy detection:

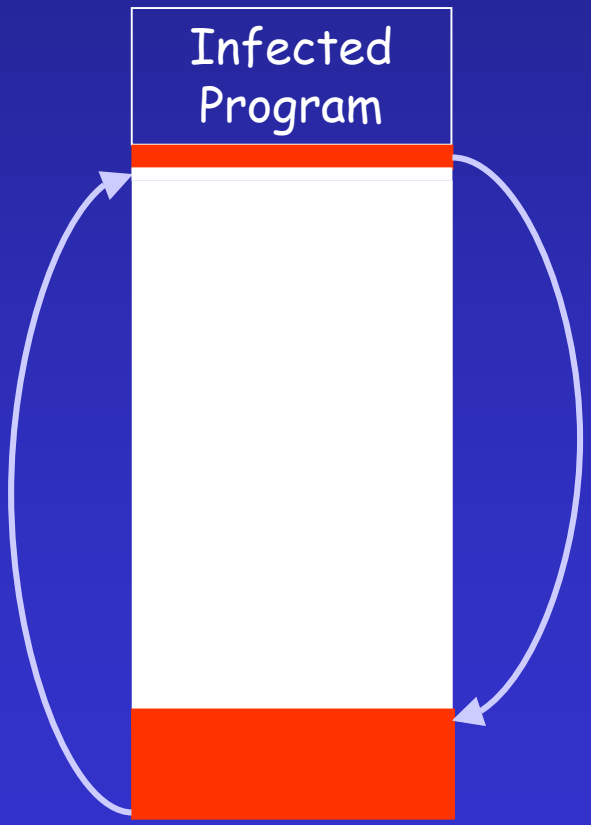


  - Signature matching

Obfuscation: Vanilla Virus

- Simple obfuscation methods

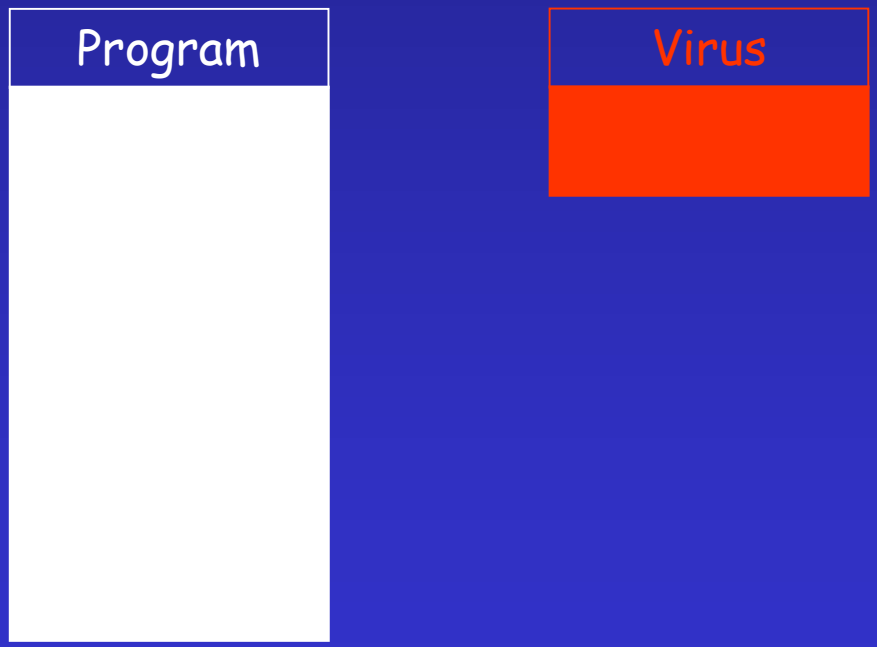
  => Easy detection:



- Signature matching
- Very successful against first-gen viruses!

Obfuscation: Polymorphism

- Encrypted virus body + morphed decryption routine

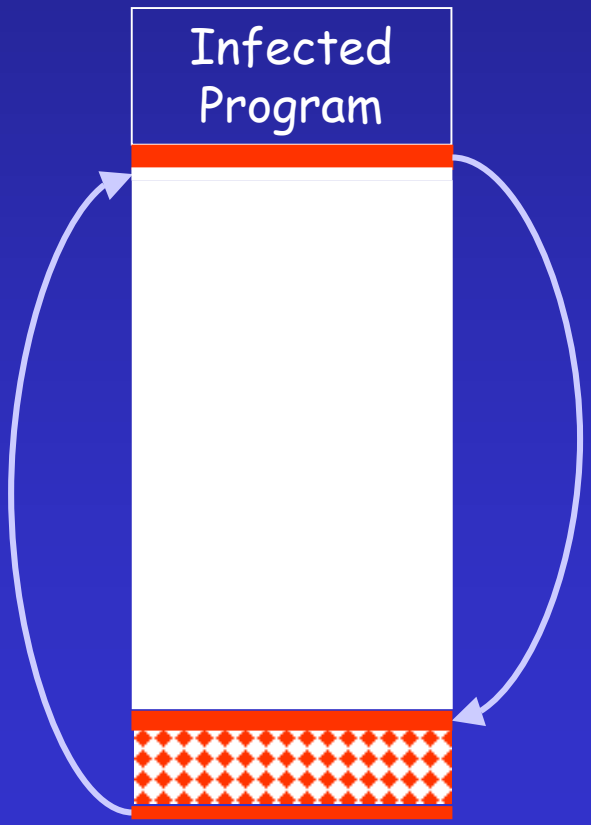


Obfuscation: Polymorphism

- Encrypted virus body +
  morphed decryption routine

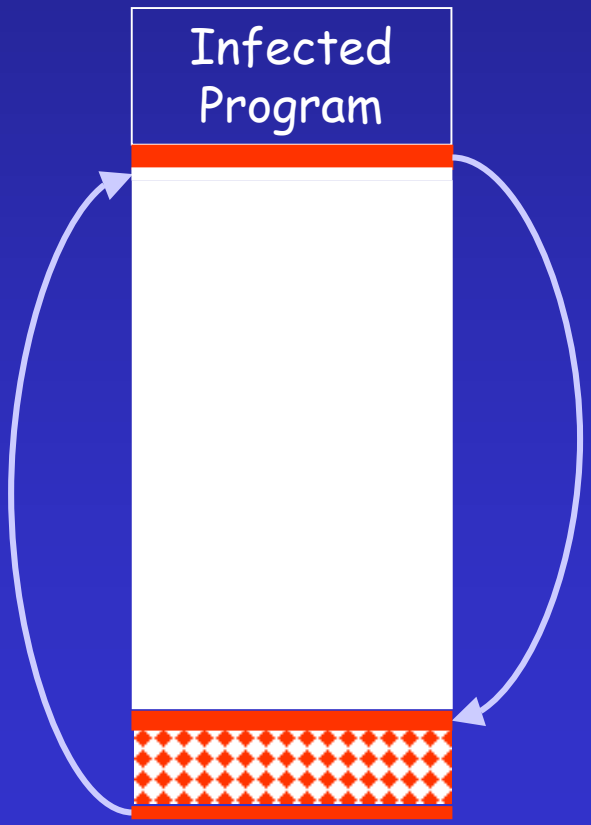
Obfuscation: Polymorphism

- Encrypted virus body + morphed decryption routine



Obfuscation: Polymorphism

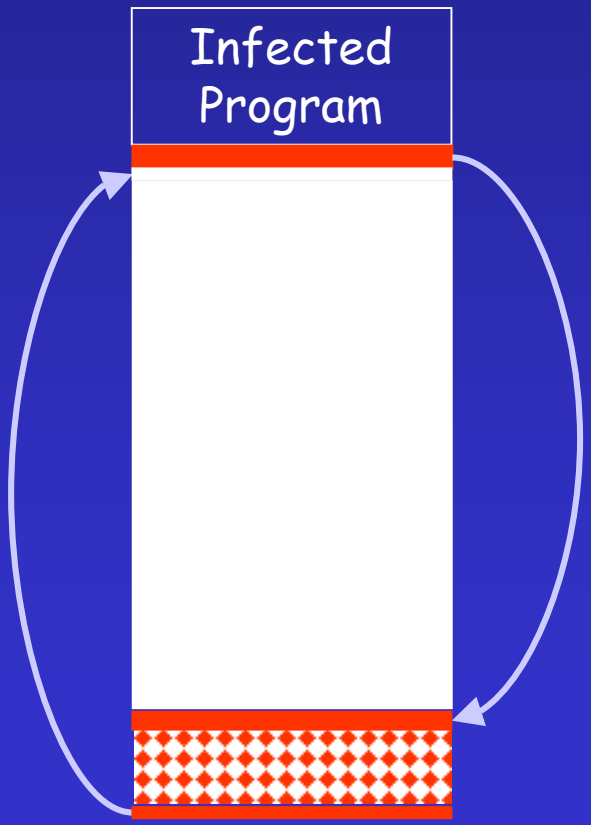
- Encrypted virus body + morphed decryption routine



=> Detection methods:

Obfuscation: Polymorphism

- Encrypted virus body + morphed decryption routine

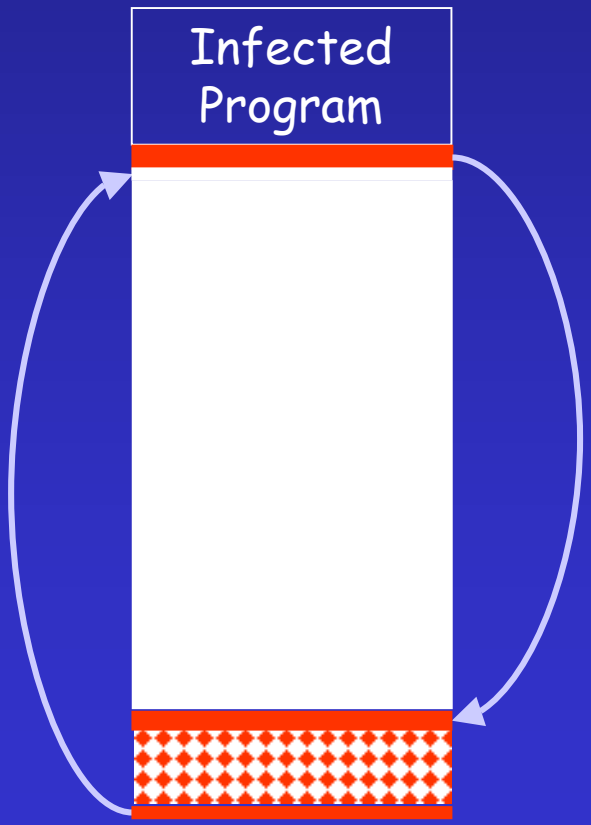


=> Detection methods:

- Heuristic detection

Obfuscation: Polymorphism

- Encrypted virus body + morphed decryption routine

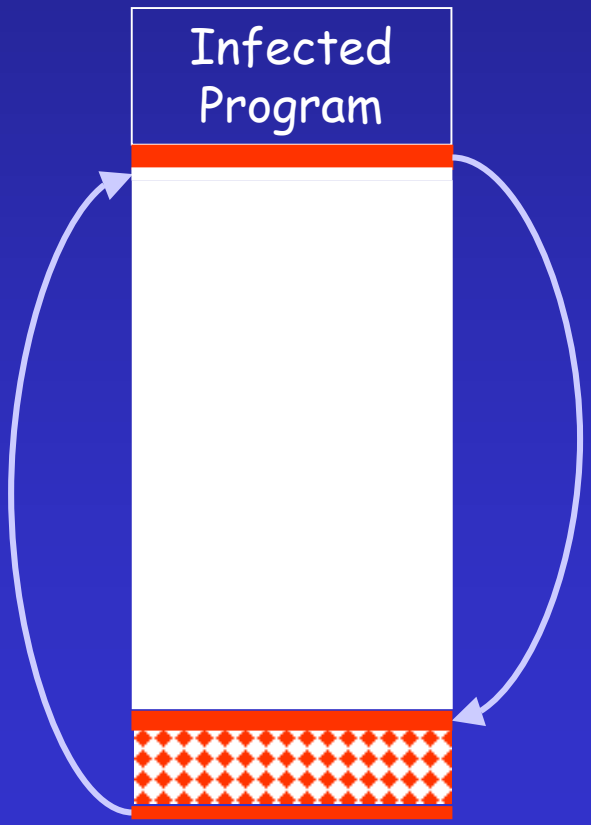


=> Detection methods:

- Heuristic detection
- Emulation

Obfuscation: Polymorphism

- Encrypted virus body + morphed decryption routine



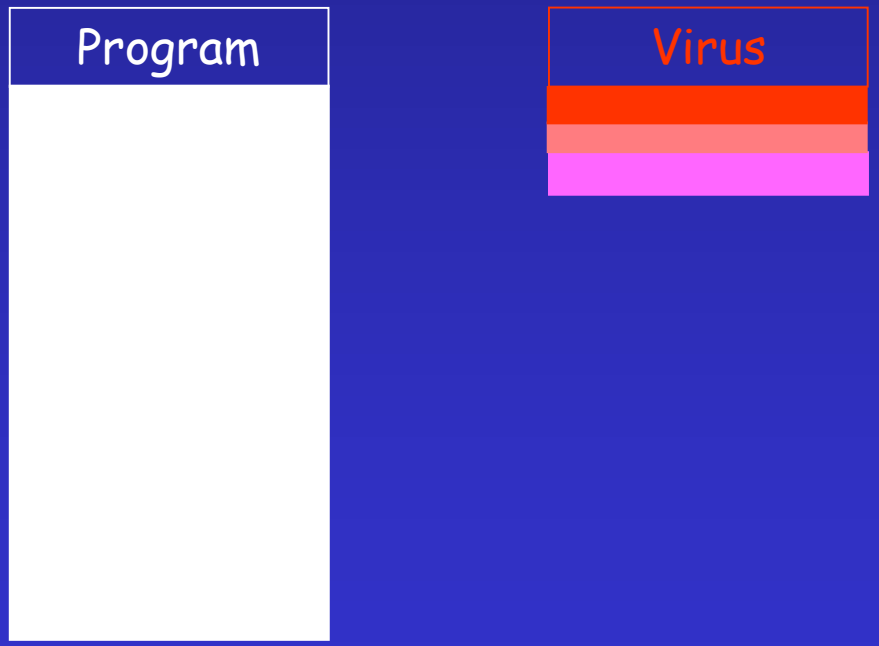
=> Detection methods:

- Heuristic detection
- Emulation

- Current state-of-the-art

Obfuscation: Metamorphism

- Metamorphic viruses:
  - Morph the whole virus body

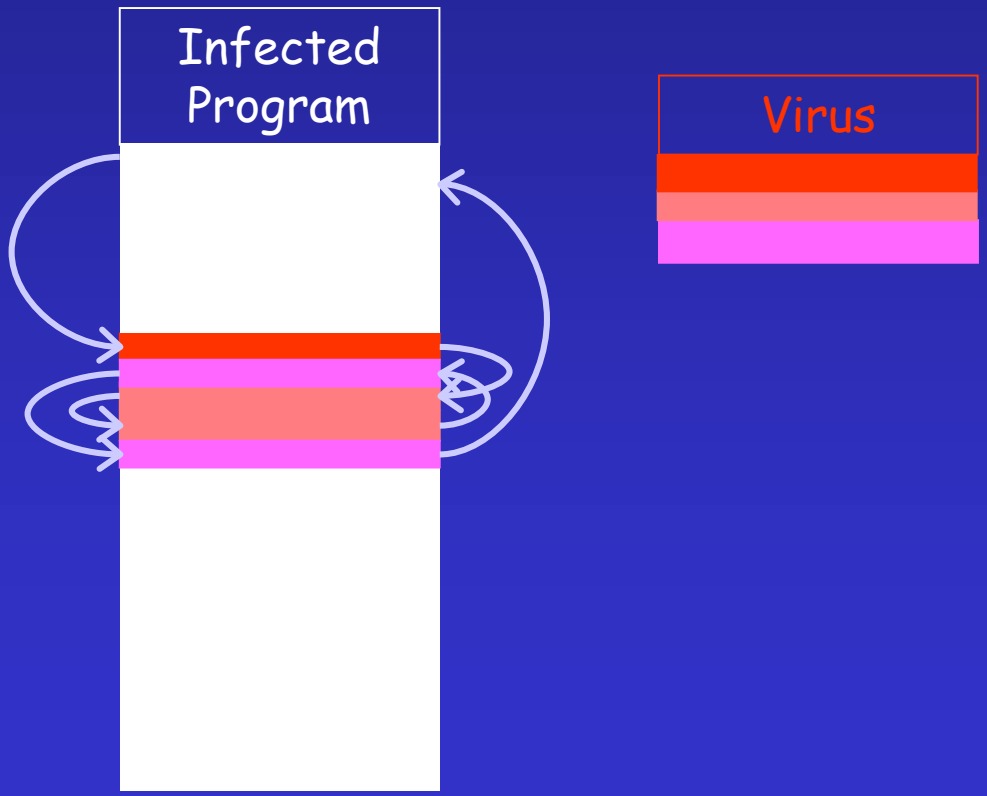


Obfuscation: Metamorphism

- Metamorphic viruses:
  - Morph the whole virus body

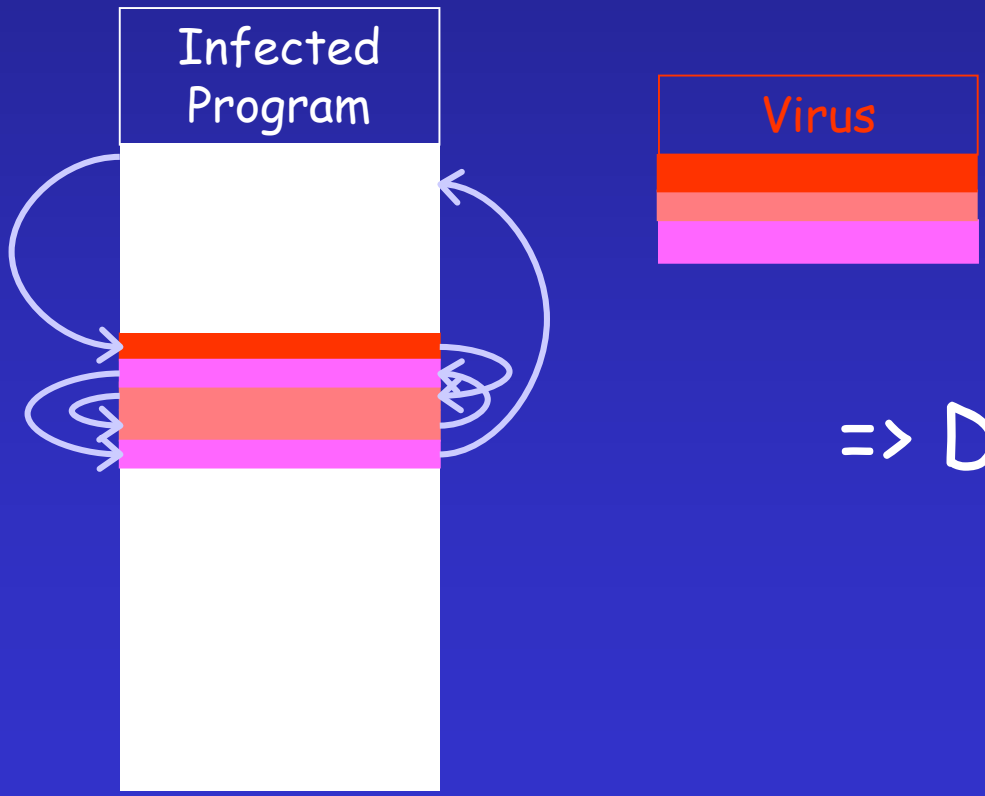
Obfuscation: Metamorphism

- Metamorphic viruses:
  - Morph the whole virus body



Obfuscation: Metamorphism

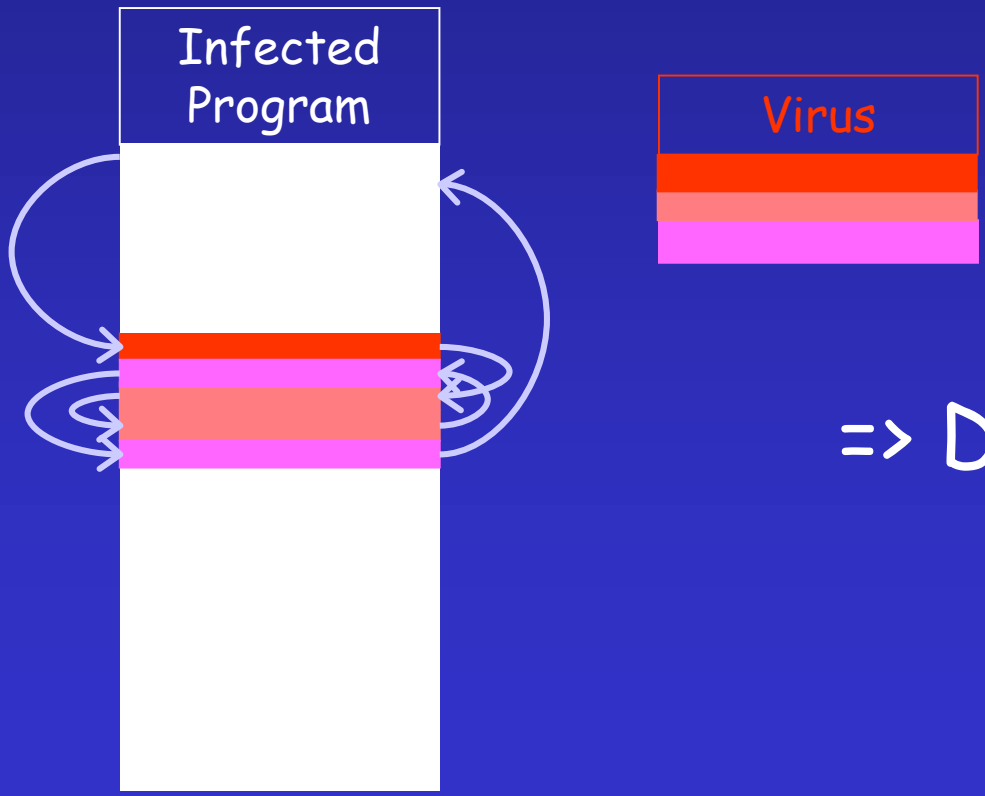
- Metamorphic viruses:
  - Morph the whole virus body



=> Detection methods

Obfuscation: Metamorphism

- Metamorphic viruses:
  - Morph the whole virus body

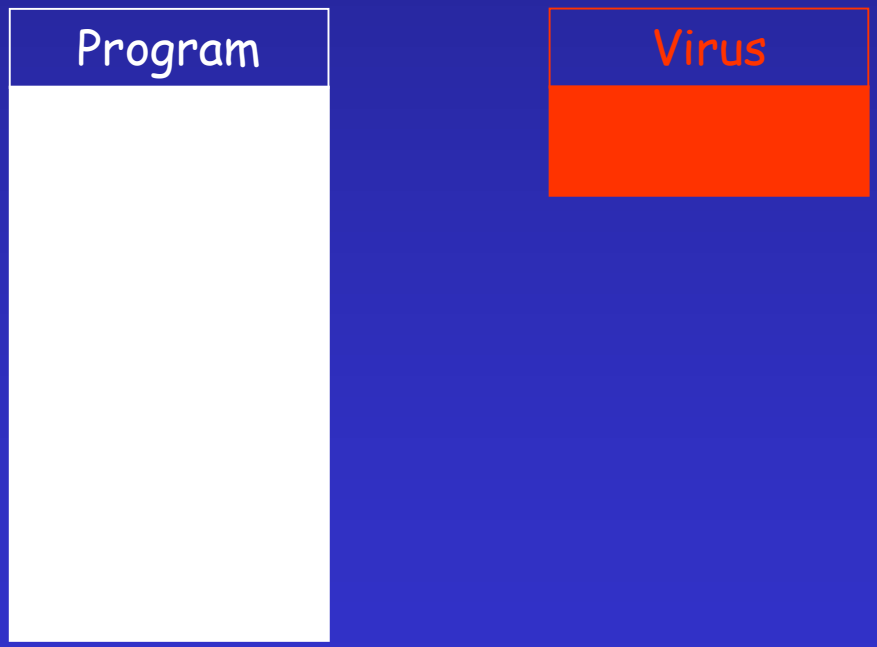


=> Detection methods

?

Obfuscation: Code Integration

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

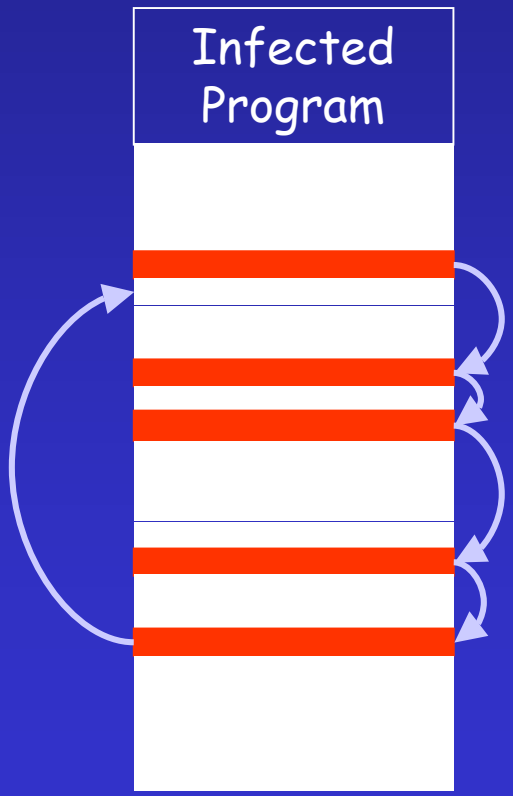


Obfuscation: Code Integration

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

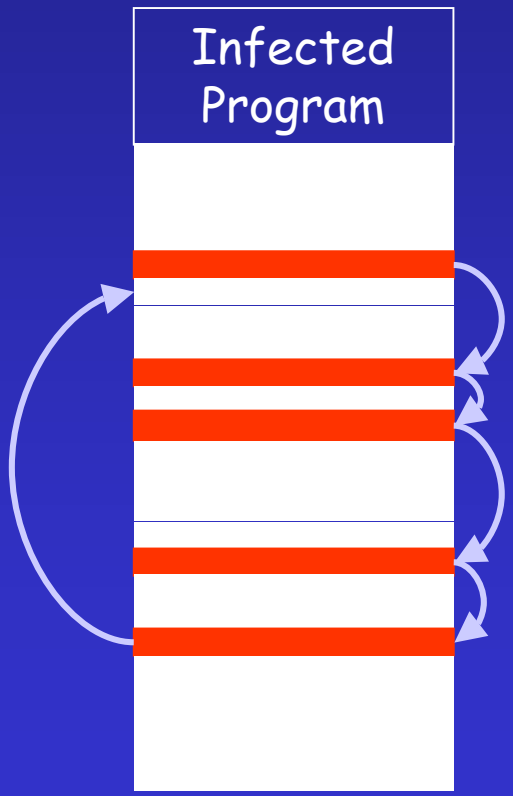
Obfuscation: Code Integration

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



Obfuscation: Code Integration

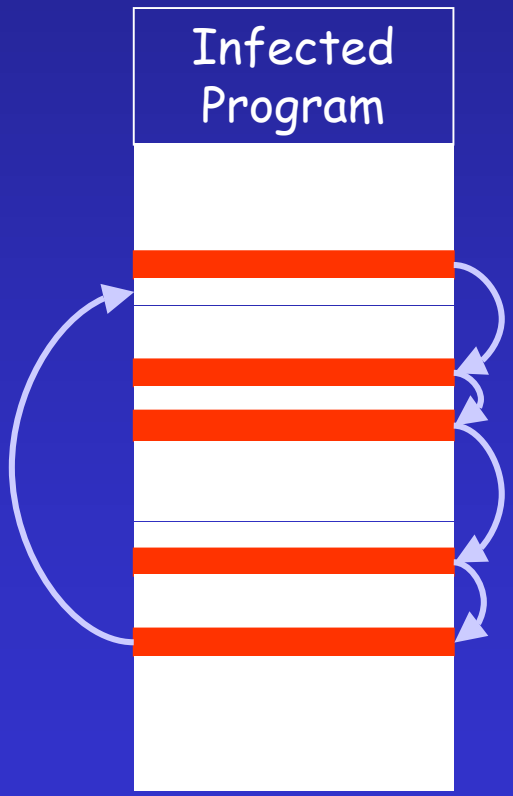
- Integration of virus and program
  - e.g. Mistfall Virus Engine



=> Detection methods

Obfuscation: Code Integration

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



=> Detection methods

?

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
```

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

        jecxz   SFModMark


        mov     esi, ecx

        mov     eax, 0d601h
        pop     edx
        pop     ecx

        call    edi


        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
            jecxz   SFModMark
            xor     ebx, ebx
            beqz    N1
N1:         mov     esi, ecx
            nop
            mov     eax, 0d601h
            pop     edx
            pop     ecx
            nop
            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
        jecxz   SFModMark
        xor     ebx, ebx
        beqz    N1
N1:     mov     esi, ecx
        nop
        mov     eax, 0d601h
        pop     edx
        pop     ecx
        nop
        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):

```
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
N2:     jmp     Loop
        jmp L4
L2:     nop
        mov     eax, 0d601h
        pop     edx
        pop     ecx
        nop
        jmp L3
L1:     jecxz   SFModMark
        xor     ebx, ebx
        beqz    N1
N1:     mov     esi, ecx
        jmp L2
L4:
```

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
N2:     jmp     Loop
        jmp L4
L2:     nop
        mov     eax, 0d601h
        pop     edx
        pop     ecx
        nop
        jmp L3
L1:     jecxz   SFModMark
        xor     ebx, ebx
        beqz    N1
N1:     mov     esi, ecx
        jmp L2
L4:
```

What to do?

What to do?

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

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

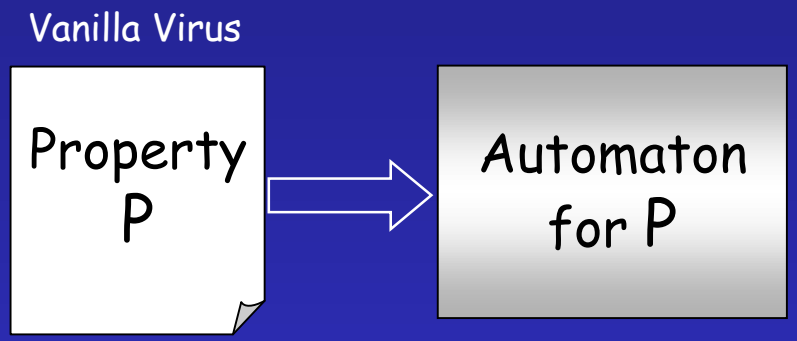
Model Checking

- Technique for checking program properties

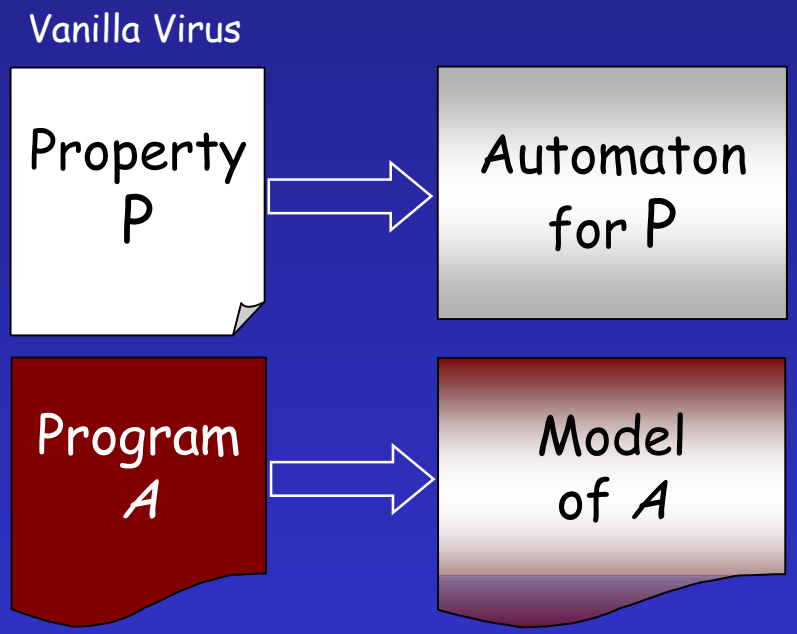
1. Build automaton for the desired property
2. Extract program model
3. Compare the model against the automaton

Model Checking

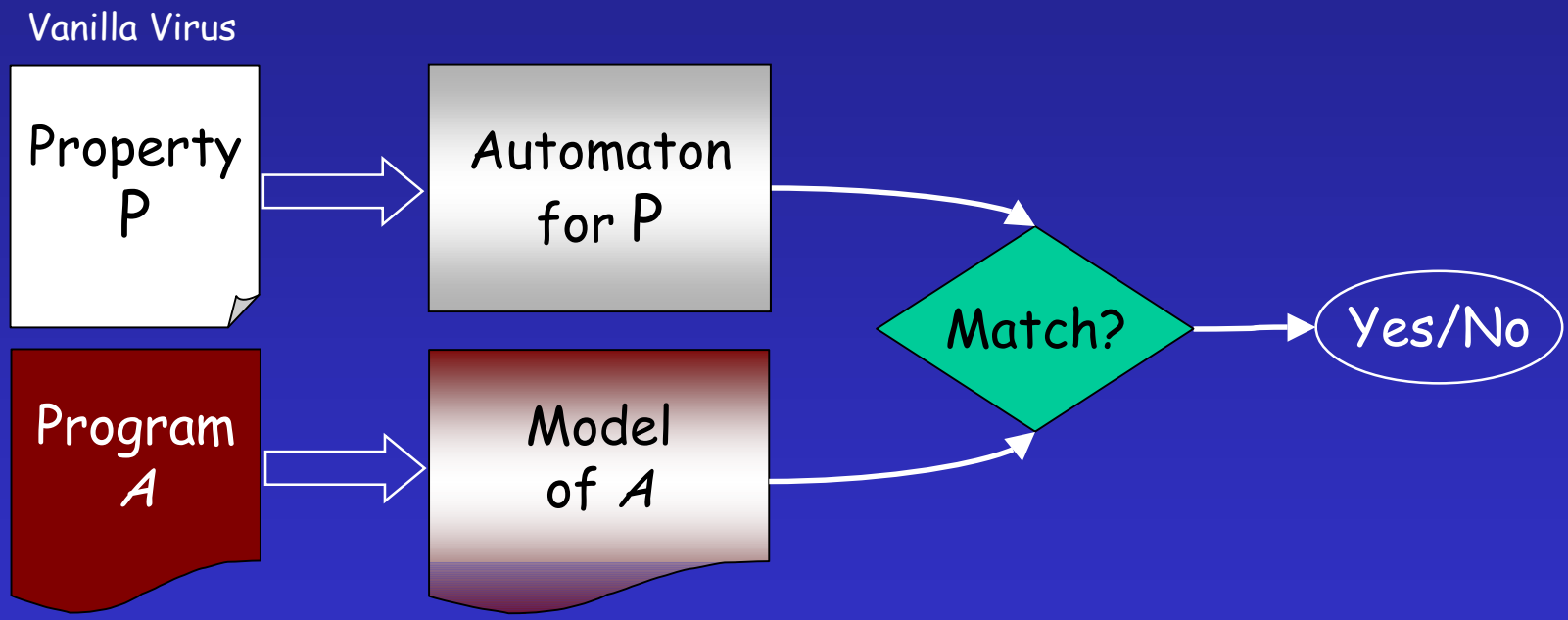
Model Checking



Model Checking

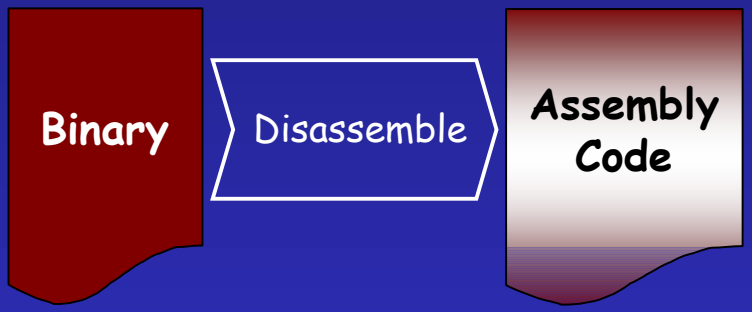


Model Checking



Model Checking Binaries

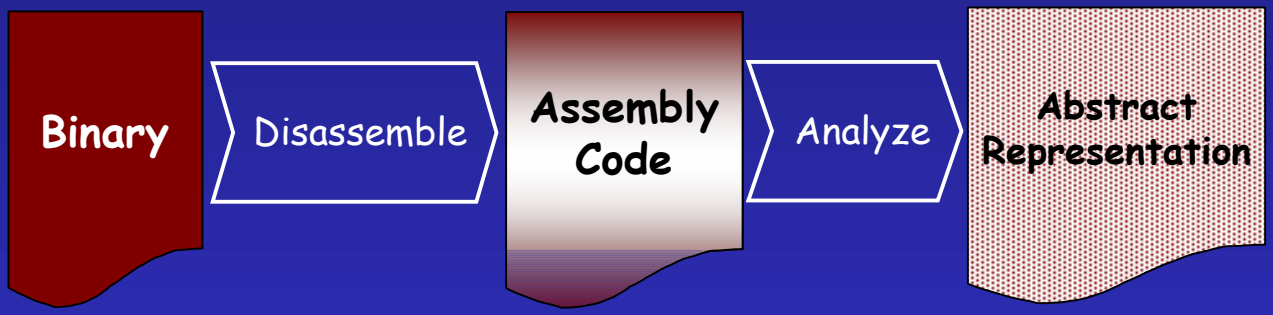
Model Checking Binaries



IDA Pro

(+ SDK plugins)

Model Checking Binaries



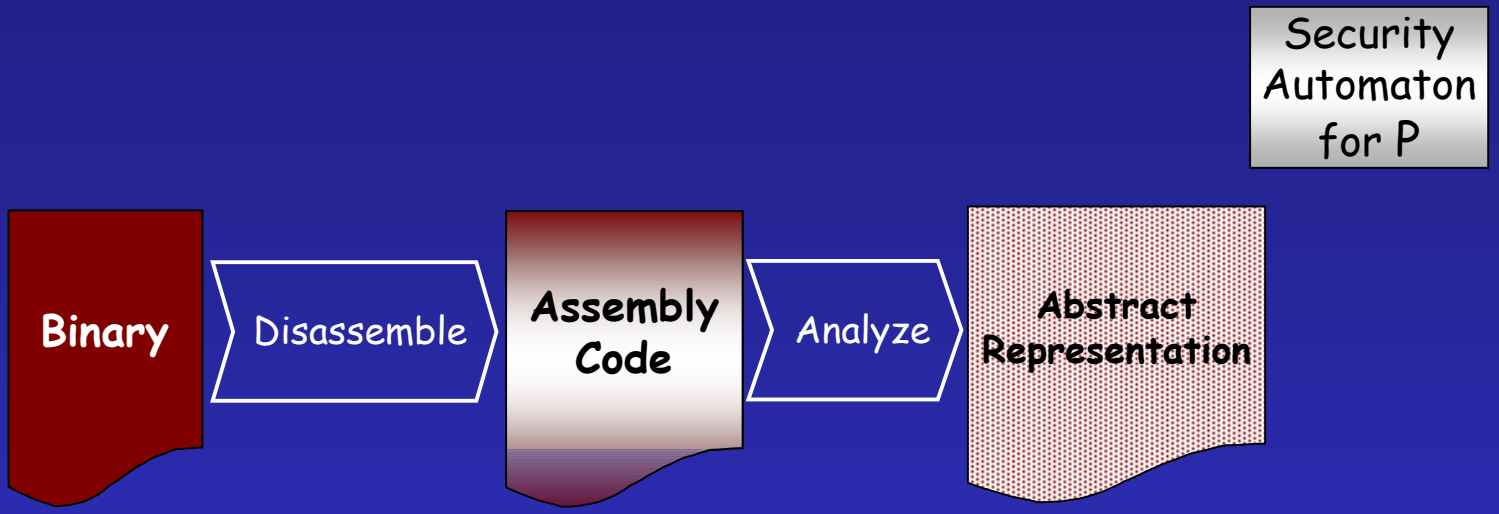
IDA Pro

(+ SDK plugins)

Transducer

(on top of CodeSurfer static analysis engine)

Model Checking Binaries



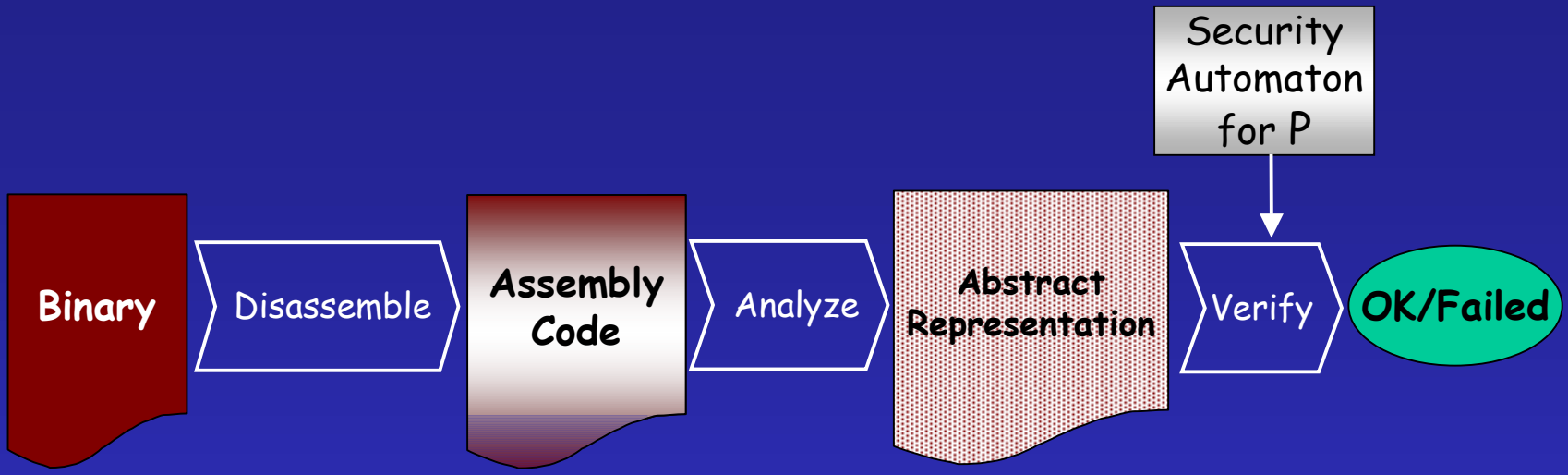
IDA Pro

(+ SDK plugins)

Transducer

(on top of CodeSurfer static analysis engine)

Model Checking Binaries



IDA Pro

(+ SDK plugins)

Transducer

(on top of CodeSurfer static analysis engine)

Model Checker

Smart Virus Scanner

Smart Virus Scanner

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

Smart Virus Scanner

1. Build automaton from vanilla virus
   - Blueprint of virus behavior
2. Build a model of the program

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:

```
push    eax
sidt    [esp-02h]
pop     ebx
add     ebx, HookNo * 08h + 04h
cli
mov     ebp, [ebx]
mov     bp, [ebx-04h]
lea     esi, MyHook - @1[ecx]
push    esi
mov     [ebx-04h], si
shr     esi, 16
mov     [ebx+02h], si
pop     esi
```

(from Chernobyl CIH 1.4 virus)

Smart Virus Scanner Example

Virus Automaton:

Virus Code:

```
push    eax
sidt    [esp-02h]
pop     ebx
add     ebx, HookNo * 08h + 04h
cli
mov     ebp, [ebx]
mov     bp, [ebx-04h]
lea     esi, MyHook - @1[ecx]
push    esi
mov     [ebx-04h], si
shr     esi, 16
mov     [ebx+02h], si
pop     esi
```

(from Chernobyl CIH 1.4 virus)

Smart Virus Scanner Example

Virus Automaton:

Virus Code:

```
push    eax
sidt    [esp-02h]
pop     ebx
add     ebx, HookNo * 08h + 04h
cli
mov     ebp, [ebx]
mov     bp, [ebx-04h]
lea     esi, MyHook - @1[ecx]
push    esi
mov     [ebx-04h], si
shr     esi, 16
mov     [ebx+02h], si
pop     esi
```

(from Chernobyl CIH 1.4 virus)

mov ebp, [ebx]

mov bp, [ebx - 04h]

lea esi, MyHook - @1[ecx]

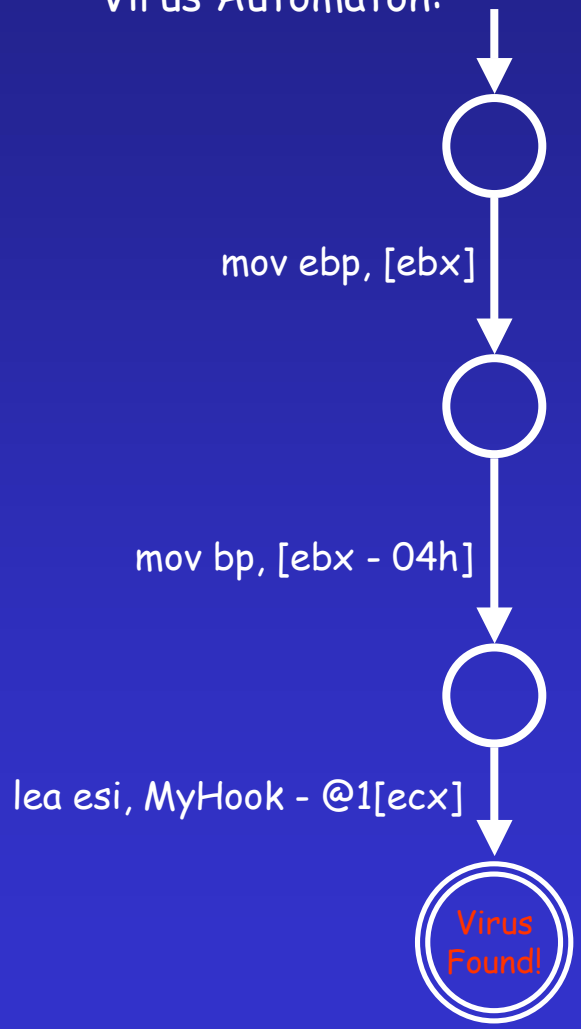
Smart Virus Scanner Example

Virus Code:

```
push    eax
sidt    [esp-02h]
pop     ebx
add     ebx, HookNo * 08h + 04h
cli
mov     ebp, [ebx]
mov     bp, [ebx-04h]
lea     esi, MyHook - @1[ecx]
push    esi
mov     [ebx-04h], si
shr     esi, 16
mov     [ebx+02h], si
pop     esi
```

(from Chernobyl CIH 1.4 virus)

Virus Automaton:

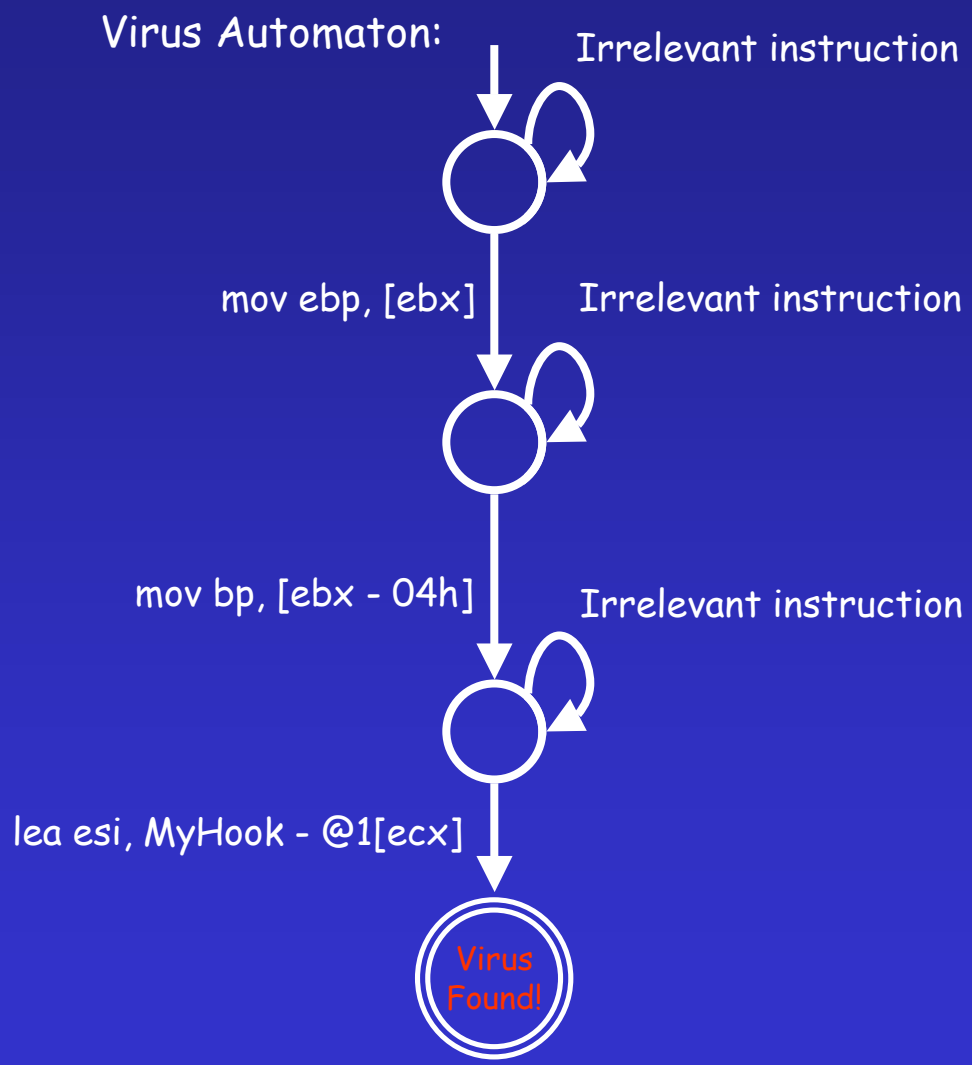


Smart Virus Scanner Example

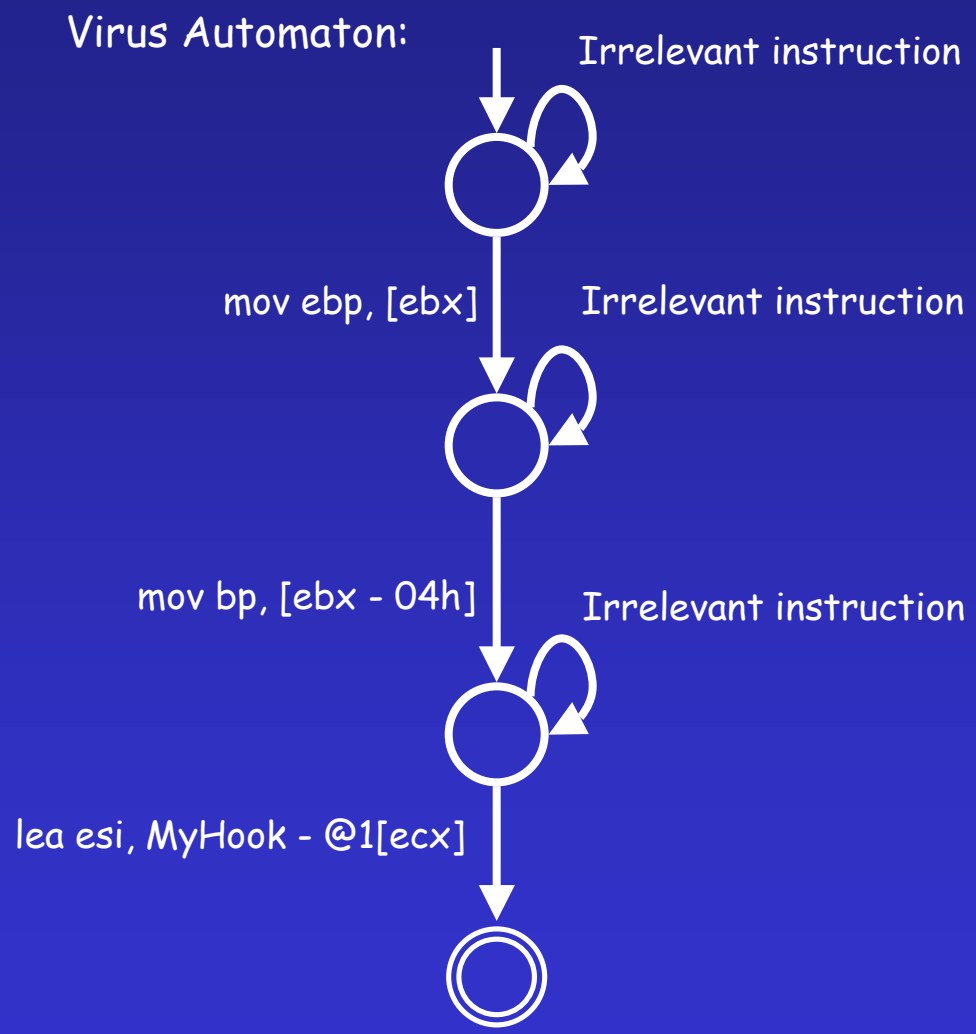
Virus Code:

```
push    eax
sidt    [esp-02h]
pop     ebx
add     ebx, HookNo * 08h + 04h
cli
mov     ebp, [ebx]
mov     bp, [ebx-04h]
lea     esi, MyHook - @1[ecx]
push    esi
mov     [ebx-04h], si
shr     esi, 16
mov     [ebx+02h], si
pop     esi
```

(from Chernobyl CIH 1.4 virus)

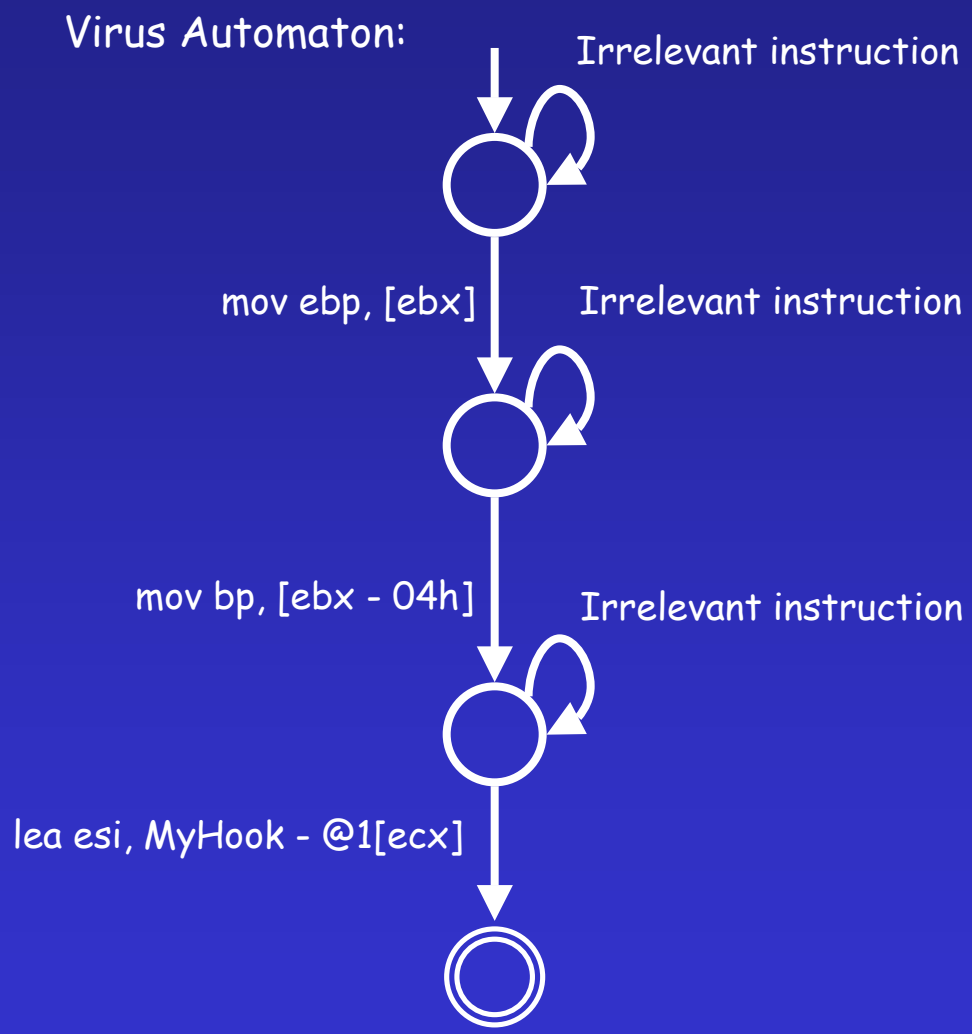


Smart Virus Scanner Example



Smart Virus Scanner Example

Virus Automaton:

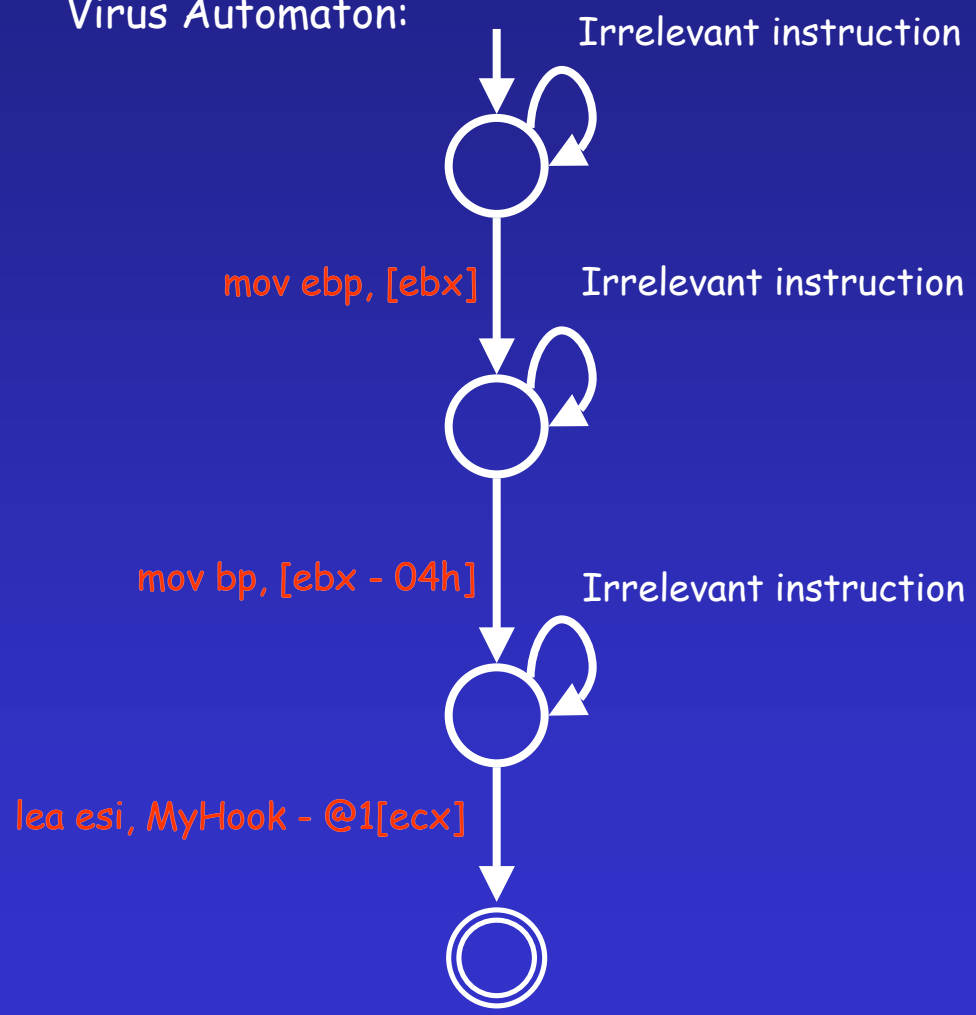


Program to be checked:

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

Smart Virus Scanner Example

Virus Automaton:

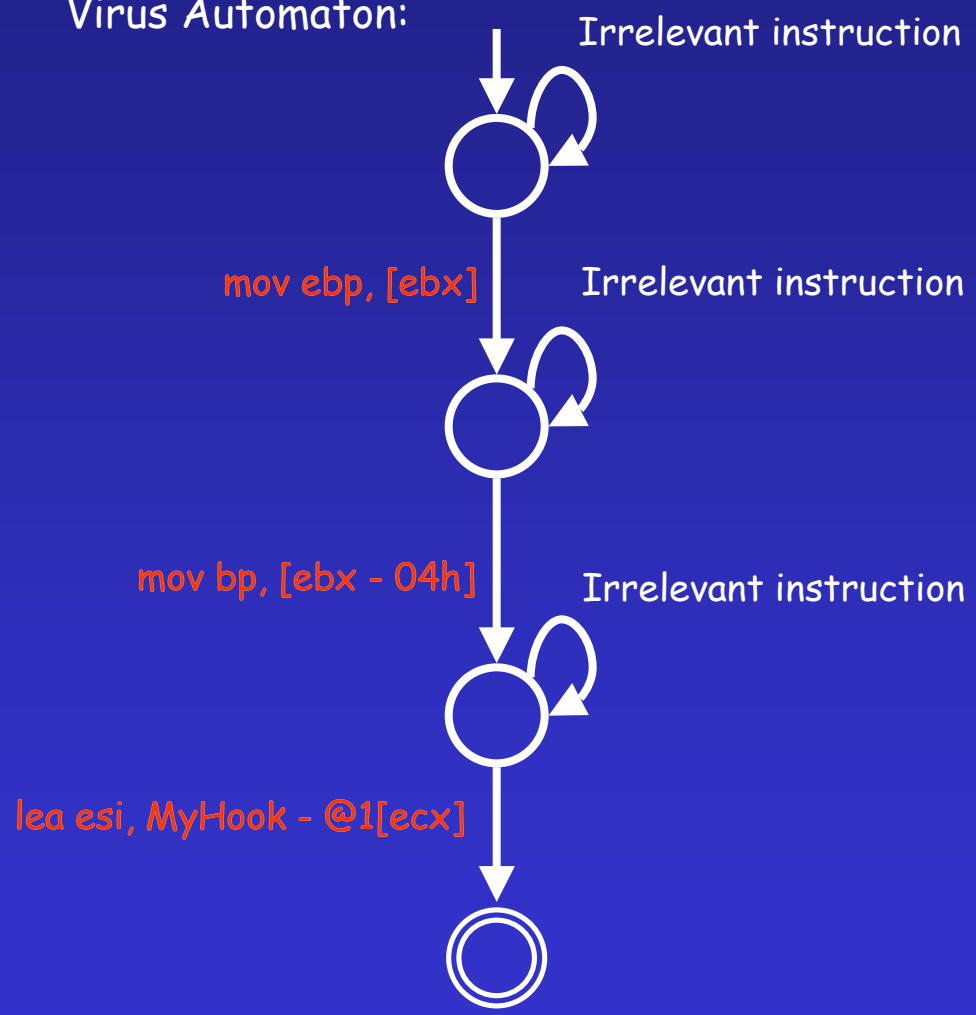


Program to be checked:

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

Smart Virus Scanner Example

Virus Automaton:



Program to be checked:

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

Smart Virus Scanner

Smart Virus Scanner

- What are *irrelevant instructions*?

Smart Virus Scanner

- What are *irrelevant instructions*?
  - NOPs

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

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

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

Current Status

- We disassemble and analyze program structure

Current Status

- We disassemble and analyze program structure
- We can detect viruses morphed in a simple manner
  - Irrelevant instructions = NOPs

Results

Results

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

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

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

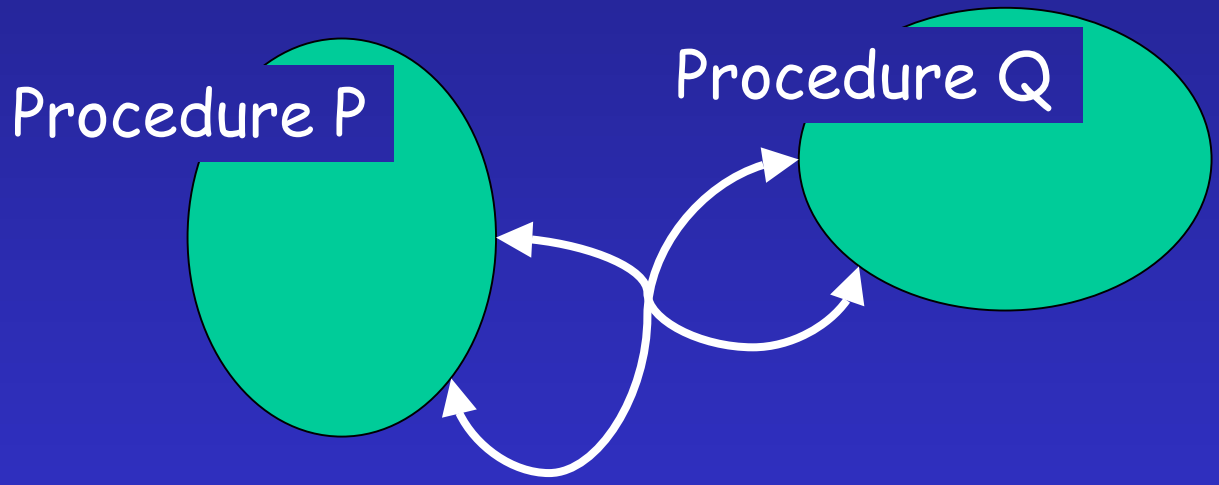
☺ Our Smart Virus Scanner catches "NOP"-morphed viruses

Current Status

- Limitations:
  - Intra-procedural only

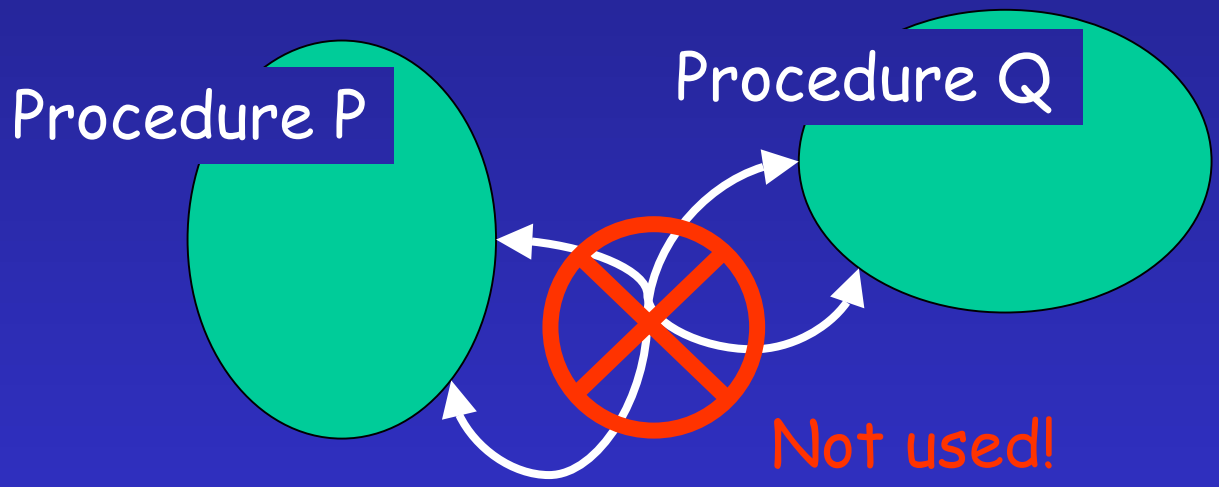
Current Status

- Limitations:
  - Intra-procedural only



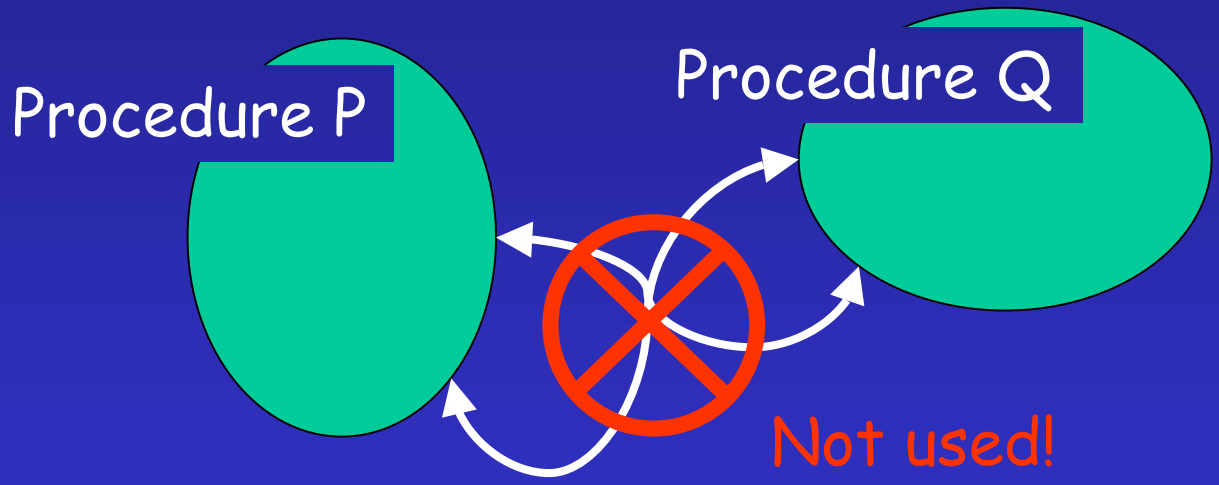
Current Status

- Limitations:
  - Intra-procedural only



Current Status

- Limitations:
  - Intra-procedural only



  - Cannot detect equivalent instruction sequences

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

Future Directions

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

Future Directions

- Context-sensitive model checking
  - Recognize virus code spread across subroutines
- Automata with uninterpreted symbols
  - Recognize virus code with different register usage

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

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

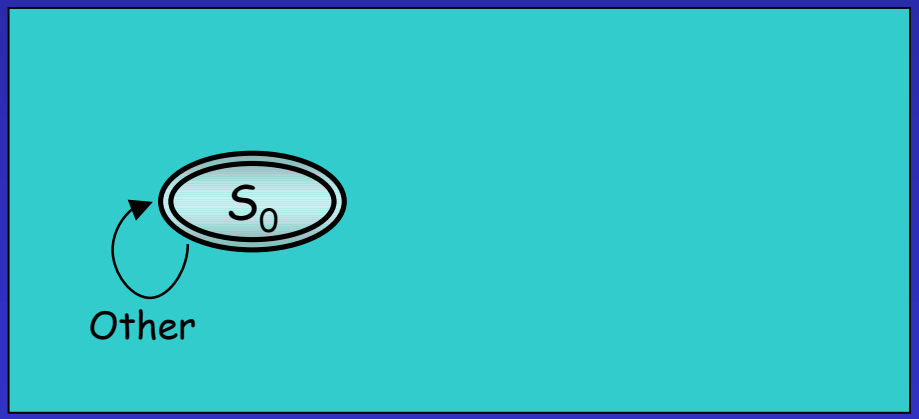
- Schneider, F.B. *Enforceable Security Policies*. TR99-1759, July 27, 1999.
- Dawson Engler, Benjamin Chelf, Andy Chou, and Seth Hallem. *Checking System Rules Using System Specific, Programmer-Written Compiler Extensions*. In Proceedings of the Fourth Symposium on Operating Systems Design and Implementation, San Diego, CA, October 2000. http://citeseer.nj.nec.com/engler00checking.html
- Péter Ször, and Peter Ferrie. *Hunting For Metamorphic*. In Proceedings of Virus Bulletin Conference, September 2001. Pp. 123 – 154. http://www.geocities.com/szorp/metamorp.pdf
- Zombie. *Zombie's Homepage*. http://z0mbie.host.sk

More Model Checking

- Security Automaton
  - For policy "Always release a resource after acquiring it."

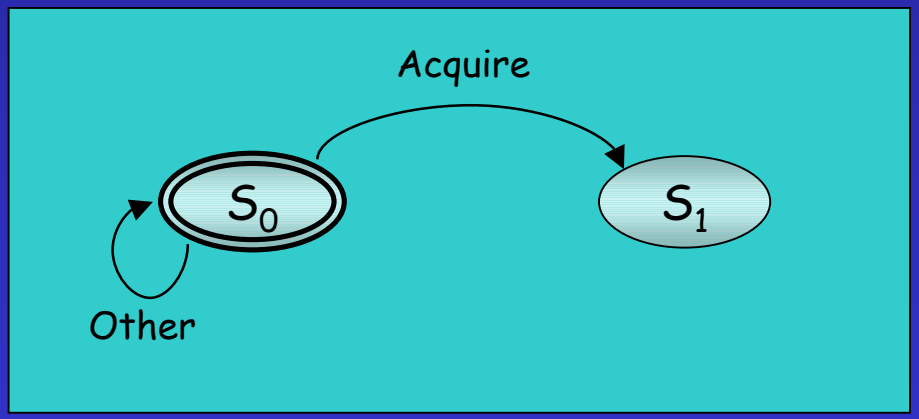
More Model Checking

- Security Automaton
  - For policy "Always release a resource after acquiring it."



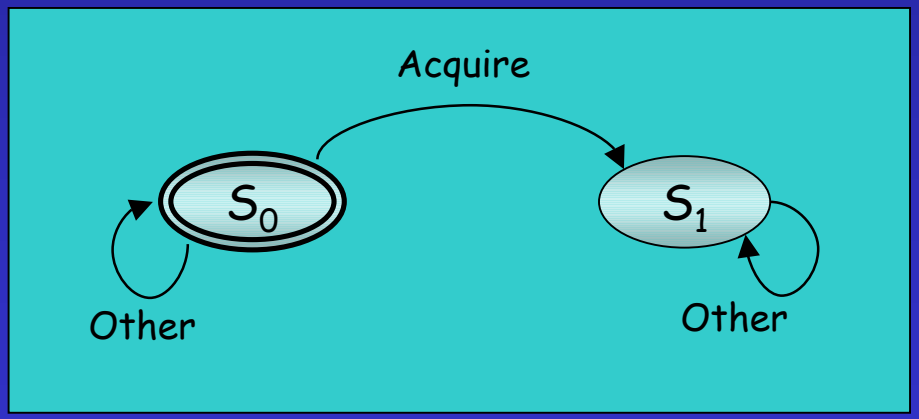
More Model Checking

- Security Automaton
  - For policy "Always release a resource after acquiring it."



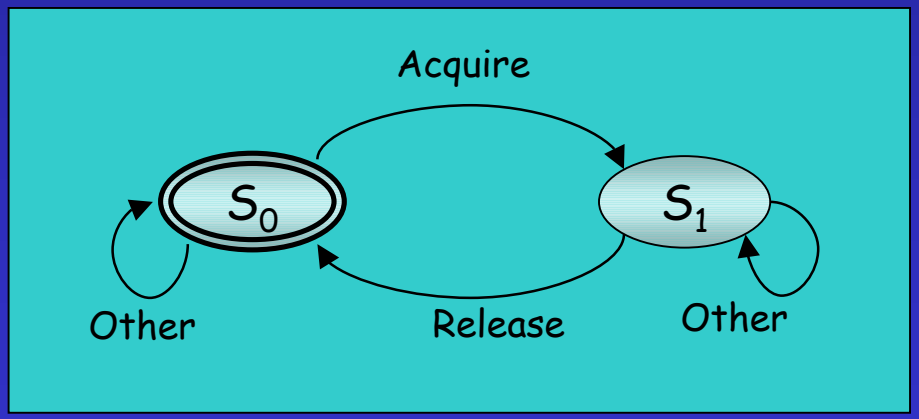
More Model Checking

- Security Automaton
  - For policy "Always release a resource after acquiring it."



More Model Checking

- Security Automaton
  - For policy "Always release a resource after acquiring it."

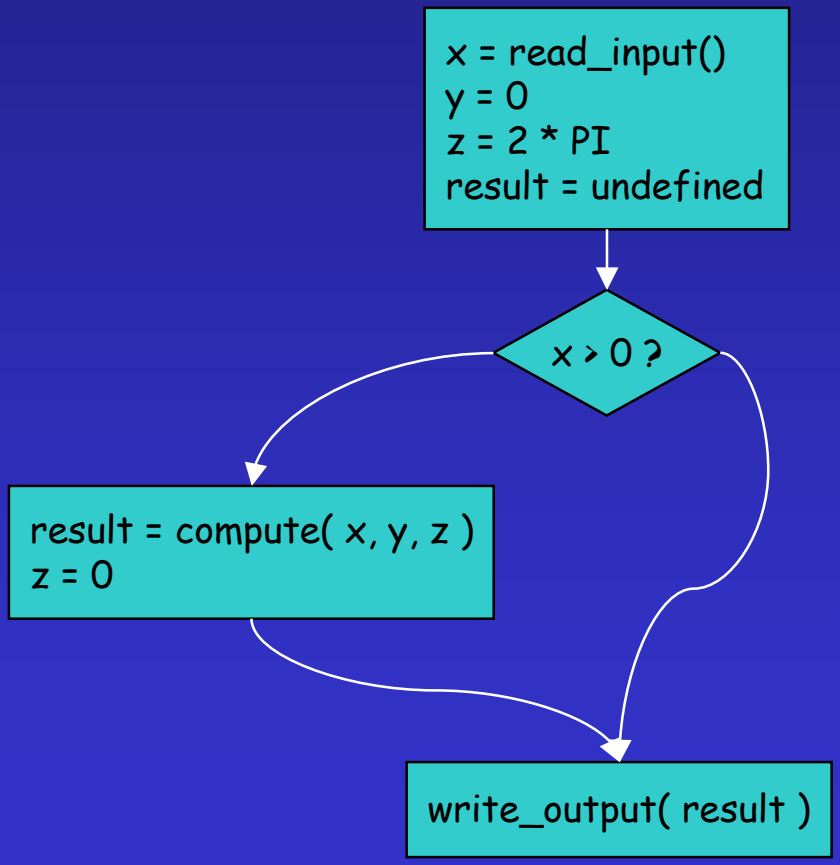


Even More Model Checking

- Abstract Representation

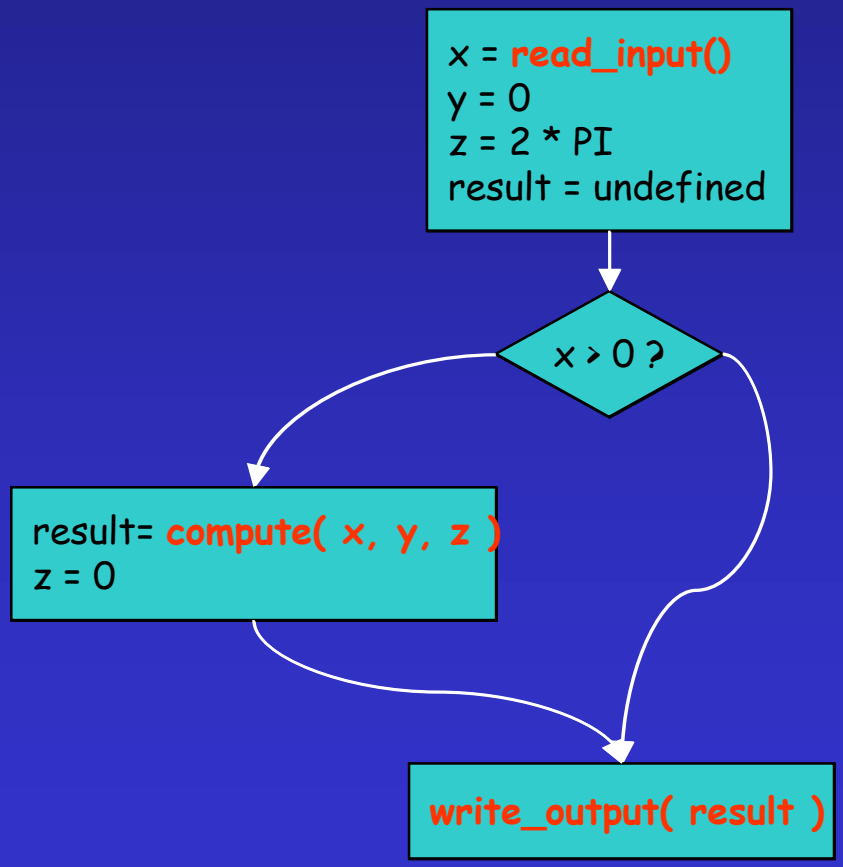
Even More Model Checking

- Abstract Representation



Even More Model Checking

- Abstract Representation



Even More Model Checking

- Abstract Representation