Testing Malicious Code Detection Tools

Mihai Christodorescu
mihai@cs.wisc.edu

WiSA http://www.cs.wisc.edu/wisa
University of Wisconsin, Madison

Problem

- Wrong focus: Testing looks only at today's malware
  - What about tomorrow's malware?
- Efficacy: How does one compare the efficacy of several malware detection tools?
  - Lack of openness about implementations

Our Solution

Test Case Generation through Obfuscation

- Generate new malware test cases
  - Using obfuscation transformations
  - Based on existing malware instances
- Test detection capabilities across a wide range of malware types

Milestones

- ✓ Binary rewriting infrastructure
  - For IA-32/Windows
  - For Visual Basic
- ⇢ Suite of obfuscations
- ⇢ Comparison metrics

Complete test suite

Overview

- Goals
- State of the art
- Our approach
- Testing environment
- Evaluation
- Conclusions

Testing Malware Detectors

Malware Detection Tool's Goal:
Detect malicious code!

- Focus: *executable code that replicates*
  - Viruses, worms, trojans
  - Not buffer overflow attacks
  - Not spyware
- Code is mobile

Testing Goals

1. Measure detection rate for existing malware
   - → False negatives
2. Measure detection rate for benign software
   - → False positives
3. Measure detection rate of new malware
   - → Resilience to new malicious code

State of the art

- Several testing labs
  - Commercial
    - Virus Bulletin
    - International Computer Security Association (ICSA)
    - West Coast Lab's CheckMark
  - Independent
    - University of Hamburg Virus Test Center (VTC)
    - University of Magdeburg

Sample certification req's

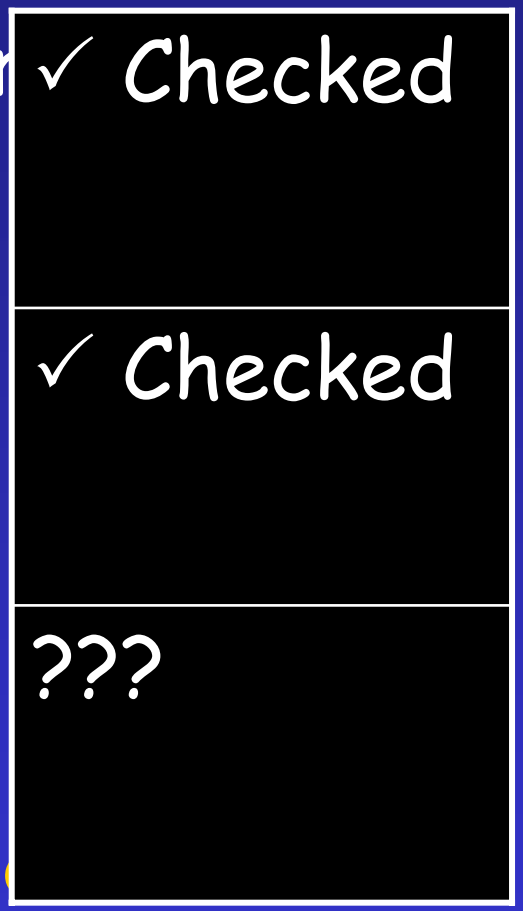
Virus Bulletin 100% Award

- Detect all *In The Wild* viruses during on-demand and on-access tests
- Generate no false positives

ITW virus lists are maintained by WildList Organization International

Testing Goals

1. Measure detection rate for malware
   - → False negatives
2. Measure detection rate of software
   - → False positives
3. Measure detection rate of malware
   - → Resilience to new malicious c



Testing against Future Malware

- First attempt:
  Andreas Marx "Retrospective Testing"
  - Test 3- and 6-month old virus scanners

Malware Type	*Detection Rate*
Macro viruses	74% - 94%
Script viruses	35% - 82%
Win32 file viruses	24% - 79%
Win32 worms, trojans, backdoors	8% - 37%

Testing against Future Malware

- We can learn from the past:
  - Often, old malicious code is slightly changed and re-launched
- Sobig e-mail worm

	Sobig.C	*Sobig.D*	*Sobig.E*
Fake "From"	`bill@microsoft.com`	`admin@support.com`	`support@yahoo.com`
Distribution	Mass e-mail, Copy to shared drive	Mass e-mail, Copy to shared drive	Mass e-mail, Copy to shared drive
Deactivation	June 8	July 2	July 14
Update path	Geocities-hosted page	Hard-coded IPs	Hard-coded IPs

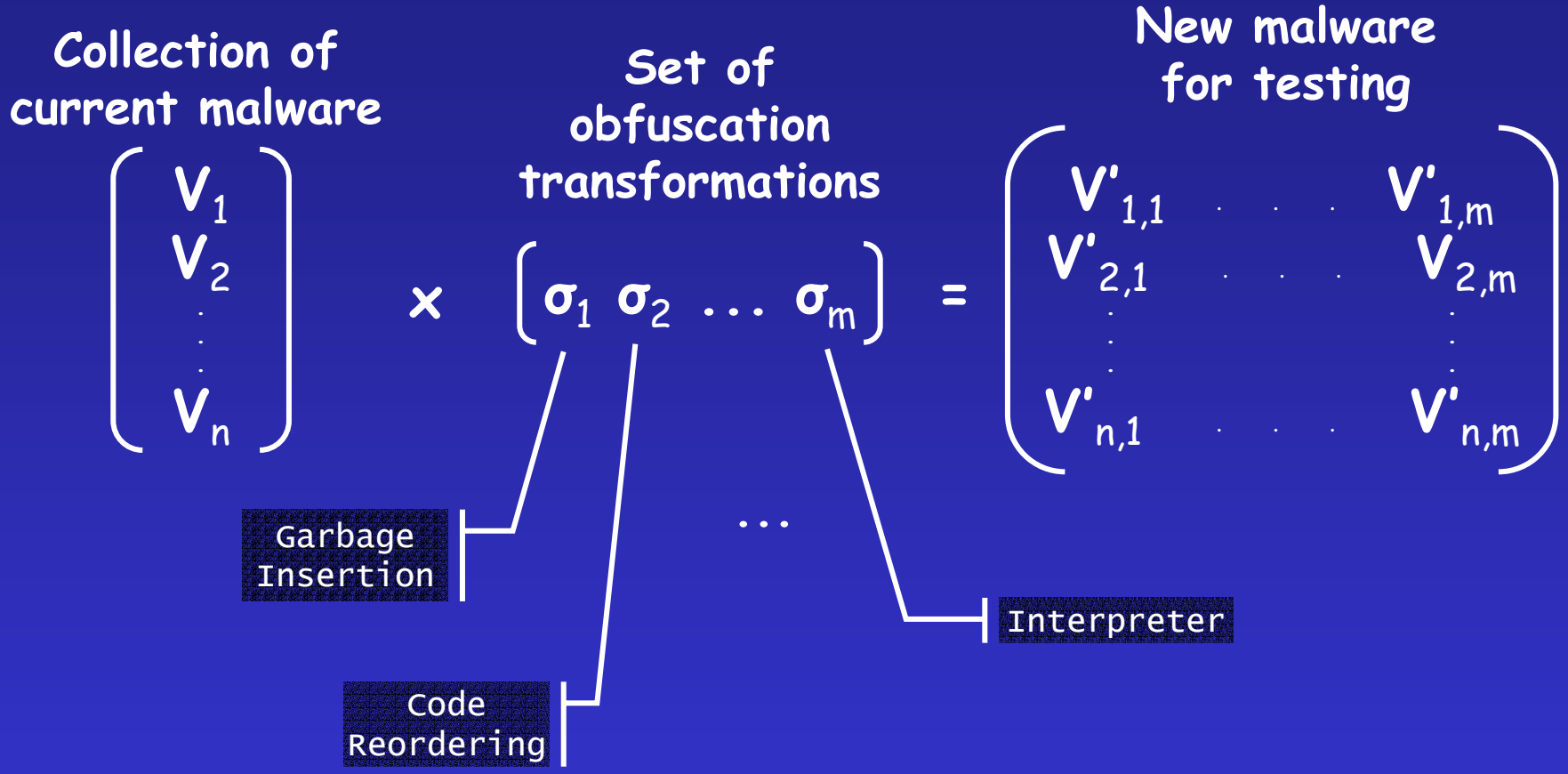
Overview

- Goals
- State of the art
- Our approach
- Testing environment
- Evaluation
- Conclusions

Obfuscation = Test Case Generation

- Obfuscate current known malware to obtain new malware test cases
- Why?
  - Many viruses / worms / trojans reuse code from older malware
  - Simulate self-mutating malware
  - Measure the ability of anti-virus tools to detect malicious behavior, not just malicious code instances

Test Case Generation



Mihai Christodorescu <mihai@cs.wisc.edu>
WiSA http://www.cs.wisc.edu

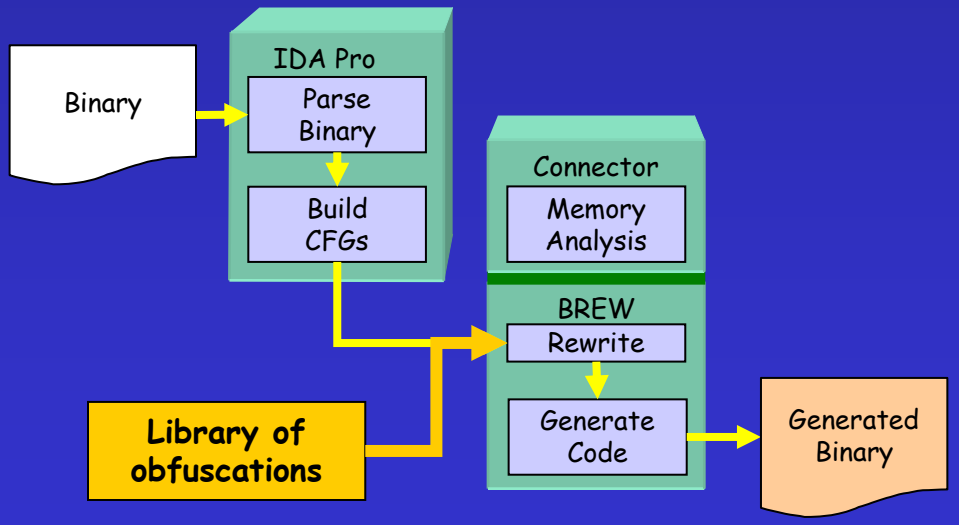
Test Case Generation

Collection of current malware

Set of obfuscation transformations

New malware for testing

$$\begin{pmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{pmatrix} \times \begin{pmatrix} \sigma_1 & \sigma_2 & \dots & \sigma_m \end{pmatrix} = \begin{pmatrix} V'_{1,1} & \dots & V'_{1,m} \\ V'_{2,1} & \dots & V_{2,m} \\ \vdots & & \vdots \\ V'_{n,1} & \dots & V'_{n,m} \end{pmatrix}$$



Mihai Christodorescu <mihai@cs.wisc.edu>
WiSA http://www.cs.wisc.edu

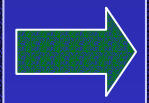
Sample Obfuscations

- Change data:
  - New strings, new dates, new constants
  - Encode / encrypt constants
- Change control:
  - Insert garbage
  - Encode / encrypt code fragments
  - Reorder code
  - Add new features
  - …

Sample Obfuscations

- Encode / encrypt constants

```
Message="Read this.."
Message.Send
```



```
Message=Decode("13FQ...")
Message.Send
...
Sub Decode(...)
...
```

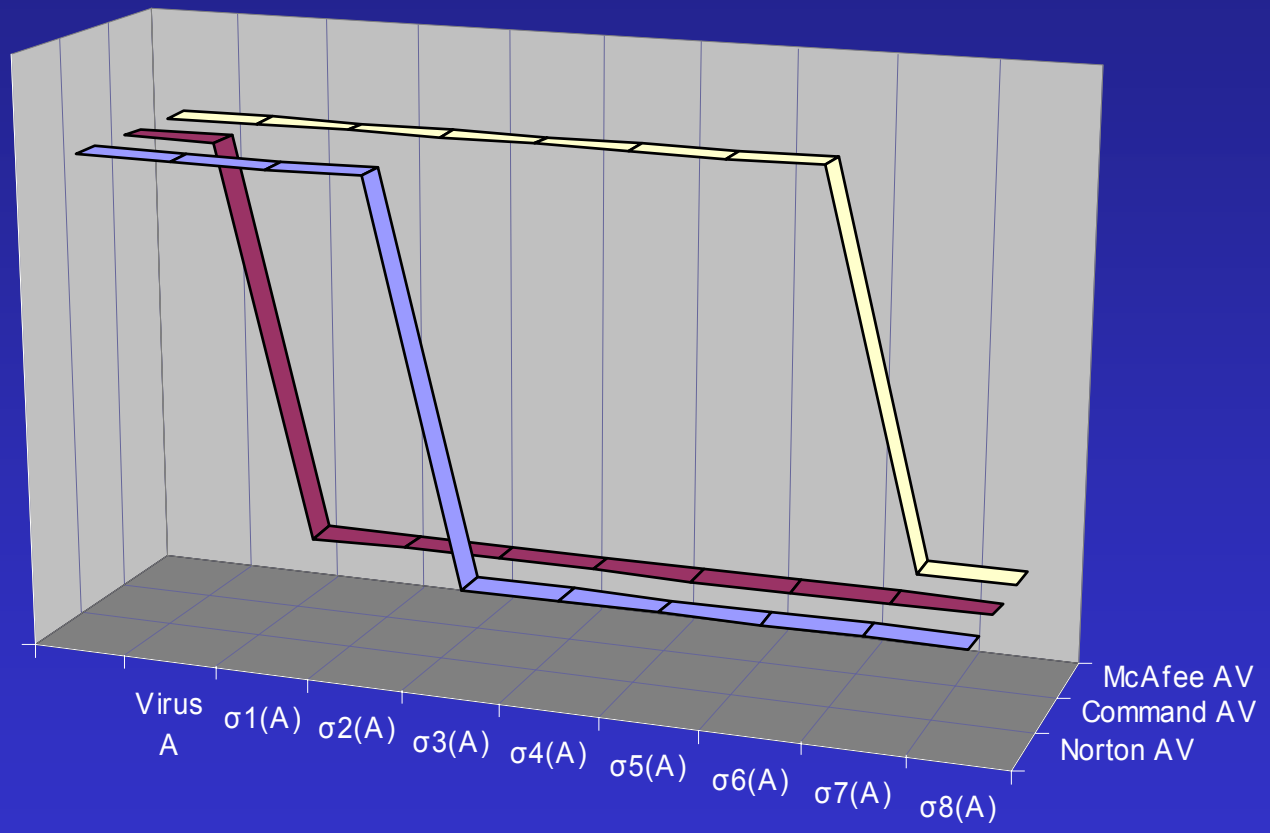

Mihai Christodorescu <mihai@cs.wisc.edu>
WiSA http://www.cs.wisc.edu

Earlier Obfuscation Results

Commercial anti-virus tools vs. morphed versions of known viruses

	Norton AntiVirus	COMMAND Software Systems	McAfee VirusScan
Chernobyl-1.4	× Not detected	× Not detected	× Not detected
f0sf0r0	× Not detected	× Not detected	× Not detected
Hare	× Not detected	× Not detected	× Not detected
z0mbie-6.b	× Not detected	× Not detected	× Not detected

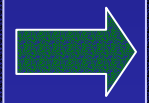
Ideal Testing Results



Parametrized Obfuscation

Encoding data:

```
Message="Read this.."
Message.Send
```



```
Message=Decode("13FQ...")
Message.Send

...
Sub Decode(...)
...
```

Parameters:

- Data to obfuscate
- Type of encoding / encryption
- Encryption key
- Location of obfuscation

Testing Environment

- Simple and complex obfuscations applied to known (detected) malware
- Multiple malware detection tools:
  - McAfee
  - Norton
  - Sophos
  - Kasperski

Overview

- Goals
- State of the art
- Our approach
- Testing environment
- Evaluation
- Conclusions

Evaluation

- Test against a set of malware detected in original form by all tools
- Test using the same obfuscations and the same parameters

- Obfuscation hierarchy
  - From simple to most complex

Metrics

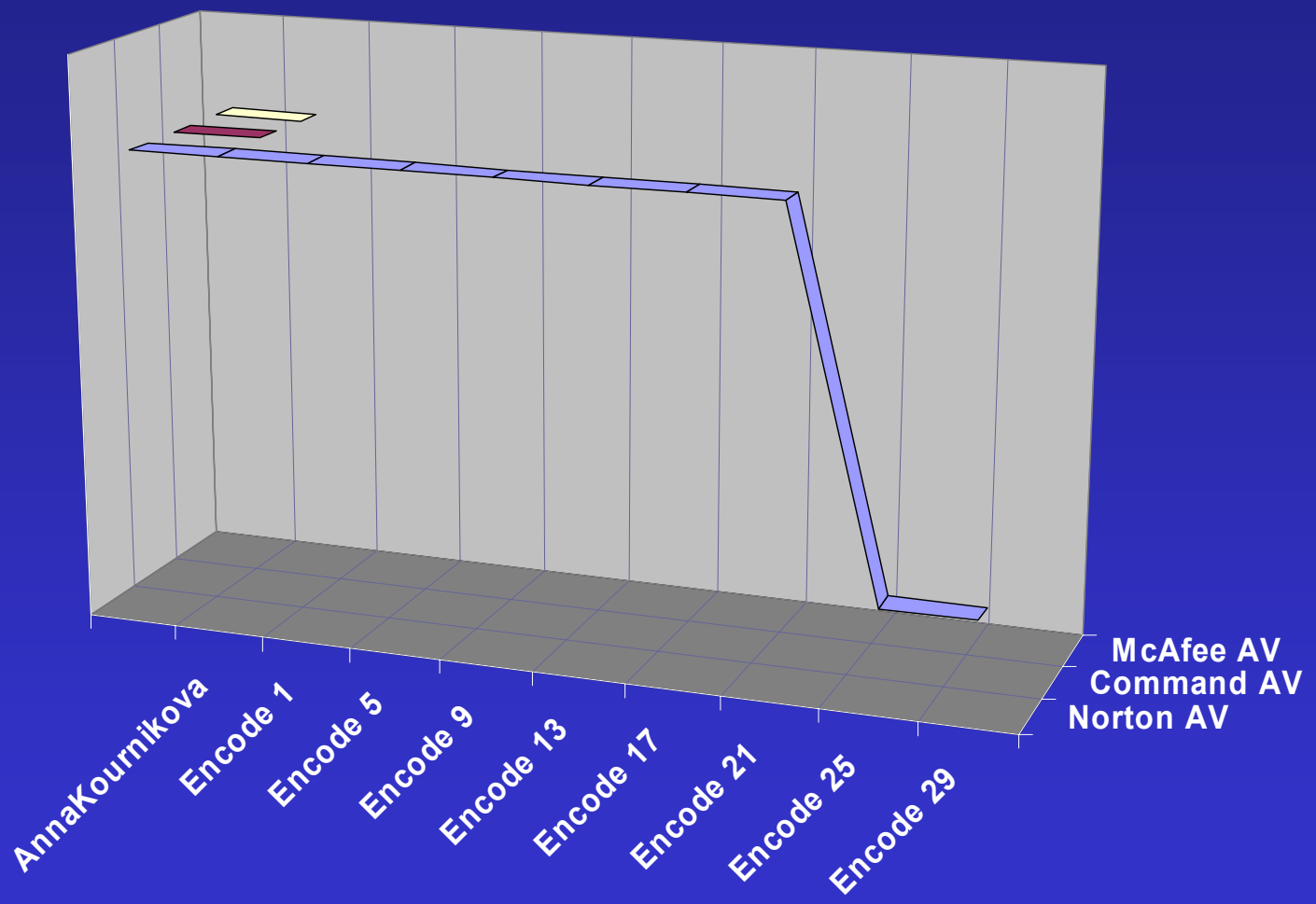
- Minimum obfuscation level:

  For a given obfuscation σ with parameters $(x_1, \ldots, x_k)$, what are the least values for each x_i that generates a false negative?

- Minimal combination of obfuscations:

  What is the smallest set of obfuscations $\{\sigma_1, \ldots, \sigma_k\}$ that generates a false negative?

Preliminary Results



Lessons Learned

- Malware spreads instantaneously
  - Arms race between malware distribution and malware detection tool update
- In testing malware detection tools, one must use a virus writer's mindset
  - It is a game → need to think several moves ahead

Future Work

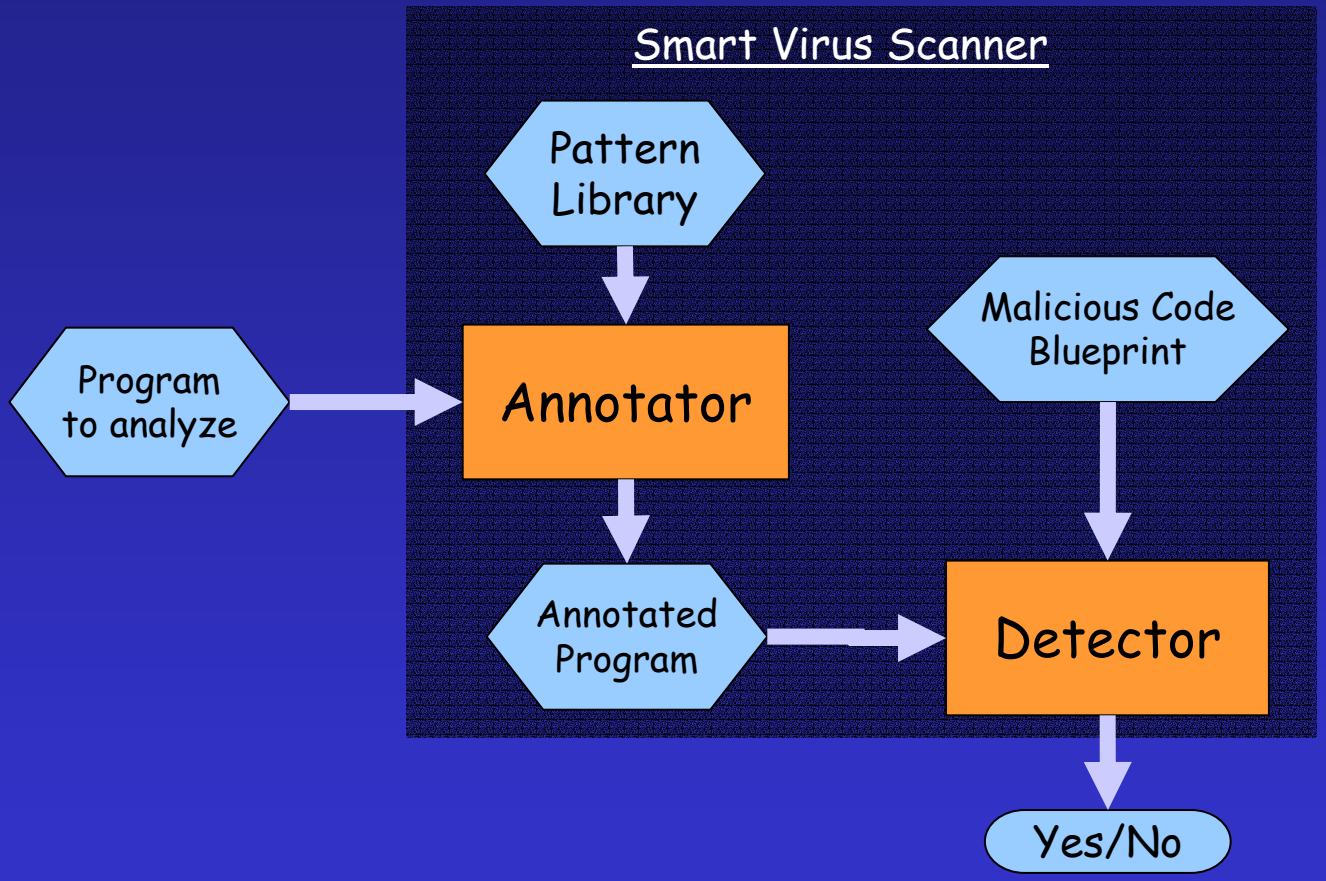
- Explore more obfuscations

  Depends on results of ongoing tests
- Future idea # 1:

  Self-guided test tool that finds the minimal test cases for false negatives
- Future idea # 2:

  Develop tests to check for detection of malicious behavior, not just code sequences

Seeing Through the Obfuscations

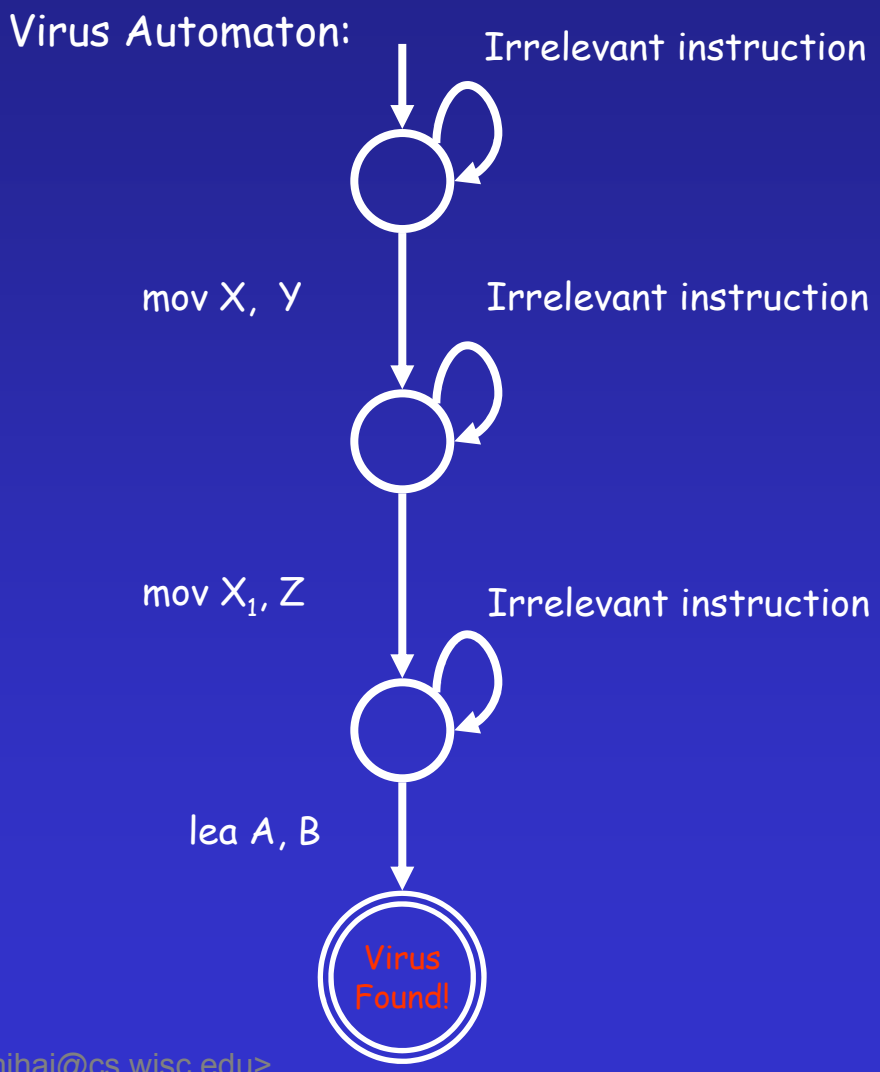


Detection Example

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)



References

Mihai Christodorescu, Somesh Jha "Static analysis of executables to detect malicious patterns". USENIX Security'03, August 2003, Washington DC.

Andreas Marx "Retrospective testing - how good heuristics really work". Virus Bulletin Conference, November 2002, New Orleans, LA.

Sarah Gordon "Antivirus software testing for the year 2000 and beyond". 3rd NISSC Proceedings, October 16-19, 2000, Baltimore, MD.

Binary Code Rewriting