

The Hyperion System: Cybersecurity through Software Behavior Computation

“What does your software do?”

Rick Linger, Stacy Prowell, Stephen Lindberg
Cyber Security and Information Intelligence Research Group
Oak Ridge National Laboratory
202-701-6257, lingerr@ornl.gov

**Customer Need:
Assure Software
Function and Security**

The Value of Assured Software

- Software vulnerability announcements cost vendors **\$860M** in market value.¹
- Average cost of security incident: **\$300K** among surveyed firms.²

1. R. Telang and S. Wattal, "An Empirical Analysis of the Impact of Software Vulnerability Announcements on Firm Stock Prices," *IEEE Transactions on Software Engineering*, 33(8):544-557, August 2007.
 2. Aberdeen Group, "Securing Your Applications: Three Ways to Play," August 31, 2010, quoted on <http://blogs.aberdeen.com/it-security/quantifying-business-value-of-application-security-cost-avoidance-cost-savings/> (retrieved on 10/2/2012).

SUMMARY STATISTICS

WHO IS BEHIND DATA BREACHES?

98% stemmed from external agents (+6%)

4% implicated internal employees (-13%)

<1% committed by business partners (↔)

58% of all data theft tied to activist groups

No big surprise here; outsiders are still dominating the scene of corporate data theft. Organized criminals were up to their typical misdeeds and were behind the majority of breaches in 2011. Activist groups created their fair share of misery and mayhem last year as well—and they stole more data than any other group. Their entrance onto the stage also served to change the landscape somewhat with regard to the motivations behind breaches. While good old-fashioned greed and avarice were still the prime movers, ideological dissent and schadenfreude took a more prominent role across the caseload. As one might expect with such a rise in external attackers, the proportion of insider incidents declined yet again this year to a comparatively scant 4%.

HOW DO BREACHES OCCUR?

Incidents involving hacking and malware were both up considerably last year, with hacking linked to almost all compromised records. This makes sense, as these threat actions remain the favored tools of external agents, who, as described above, were behind most breaches. Many attacks continue to thwart or circumvent authentication by combining stolen or guessed credentials (to gain access) with backdoors (to retain access). Fewer ATM and gas pump skimming cases this year served to lower the ratio of physical attacks in this report. Given the drop in internal agents, the misuse category had no choice but to go down as well. Social tactics fell a little, but were responsible for a large amount of data loss.

81% utilized some form of hacking (+31%)

69% incorporated malware (+20%)

10% involved physical attacks (-19%)

7% employed social tactics (-4%)

5% resulted from privilege misuse (-12%)

WHAT COMMONALITIES EXIST?

79% of victims were targets of opportunity (-4%)

96% of attacks were not highly difficult (+4%)

94% of all data compromised involved servers (+18%)

85% of breaches took weeks or more to discover (+6%)

92% of incidents were discovered by a third party (+6%)

97% of breaches were avoidable through simple or intermediate controls (+1%)

96% of victims subject to PCI DSS had not achieved compliance (+7%)

Findings from the past year continue to show that target selection is based more on opportunity than on choice. Most victims fell prey because they were found to possess an (often easily) exploitable weakness rather than because they were pre-identified for attack.

Whether targeted or not, the great majority of victims succumbed to attacks that cannot be described as highly difficult. Those that were on the more sophisticated side usually exhibited this trait in later stages of the attack after initial access was gained.

Given this, it's not surprising that most breaches were avoidable (at least in hindsight) without difficult or expensive countermeasures. Low levels of PCI DSS adherence highlight a plethora of issues across the board for related organizations.

While at least some evidence of breaches often exists, victims don't usually discover their own incidents. Third parties usually clue them in, and, unfortunately, that typically happens weeks or months down the road.

Did you notice how most of these got worse in 2011?

Attacks Depend on Knowledge of Behavior

- Our adversaries are students of our software
 - Study and execute our code
 - Accumulate and share knowledge
 - Know more than we do
 - Discover and exploit behavior unknown to us
- Behavior discovery is a big business

Software Realities I

- Most software is out of intellectual control
 - Full behavior is not known by us
- Attack and defense
 - All about understanding and exploiting behavior
- Software is complex
 - That's no excuse – our adversaries figure it out

Software Realities II

- Unknown behavior means unknown ...
 - Errors
 - Vulnerabilities
 - Security properties
 - Risks
 - Opportunities for adversaries
- At the core of the cyber security problem

Today's Response

- Set security standards
 - Install security tools
 - Scan code using signatures
 - Train security personnel
 - Conduct risk management
 - Analyze supply chains
-
- All good, but we're losing the battle

Technical Approach: Compute the Behavior of Software

The Idea of Behavior Computation

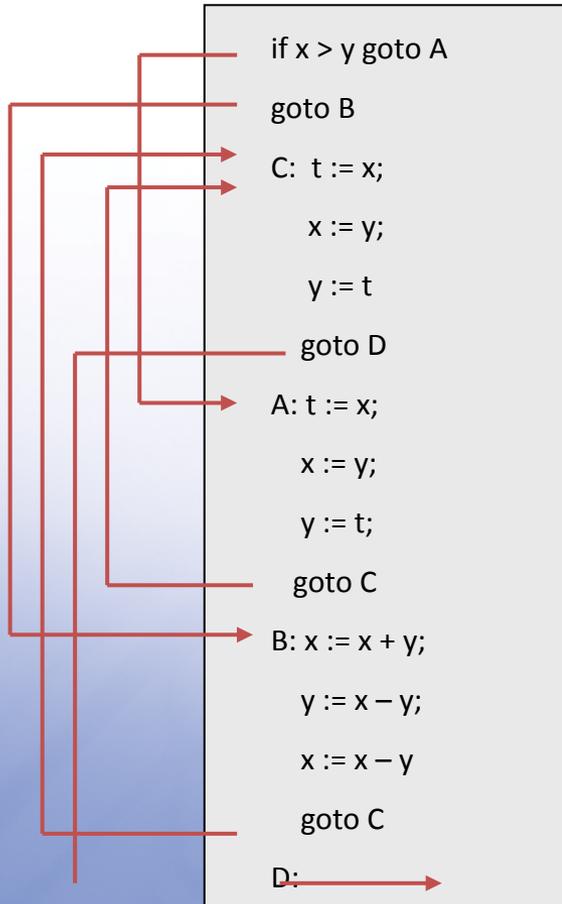
- New technology for understanding software (malware)
 - What is computed behavior?
 - What a program does in all circumstances of use
 - The “as-built” specification
-
- What are key properties?
 - Operates on program semantics, not syntax
 - Analyzes binaries to approach ground truth
 - Mathematical precision, no heuristics
 - Doesn’t look for things in code
 - Zero day makes no difference – just more behavior

Key Concept: Program Structuring

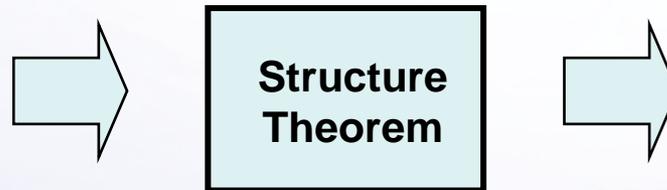
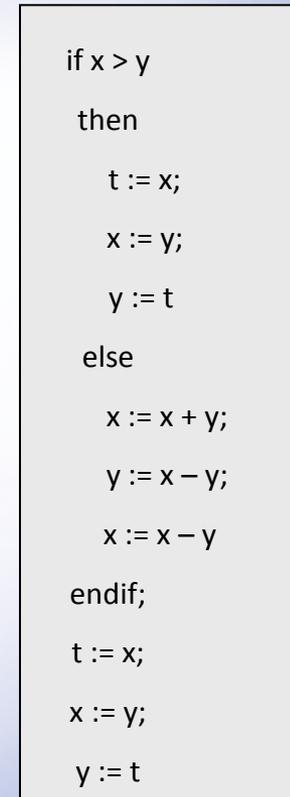
Cyberspace Sciences & Information Intelligence (CSII) Group

Computational Sciences & Engineering Division

Unstructured (obfuscated) spaghetti logic:



Transformation to structured form:

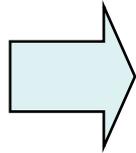


Coelesces and aggregates related code into a systematic structure

Key Concept: Behavior Computation

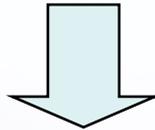
Program:

```
do
  x := x + y;
  y := x - y;
  x := x - y
enddo
```



Correctness Theorem

Defines mathematical transformations from procedural logic expressed in sequence, ifthenelse, and whiledo forms into behaviorally-equivalent, functional forms.



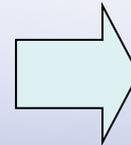
Computation:

assignment	x	y
1 x := x + y	$x_1 = x_0 + y_0$	$y_1 = y_0$
2 y := x - y	$x_2 = x_1$	$y_2 = x_1 - y_1$
3 x := x - y	$x_3 = x_2 - y_2$	$y_3 = y_2$

Derivations:

$$\begin{aligned} x_3 &= x_2 - y_2 \\ &= x_1 - (x_1 - y_1) \\ &= y_1 \\ &= y_0 \end{aligned}$$

$$\begin{aligned} y_3 &= y_2 \\ &= x_1 - y_1 \\ &= x_0 + y_0 - y_0 \\ &= x_0 \end{aligned}$$



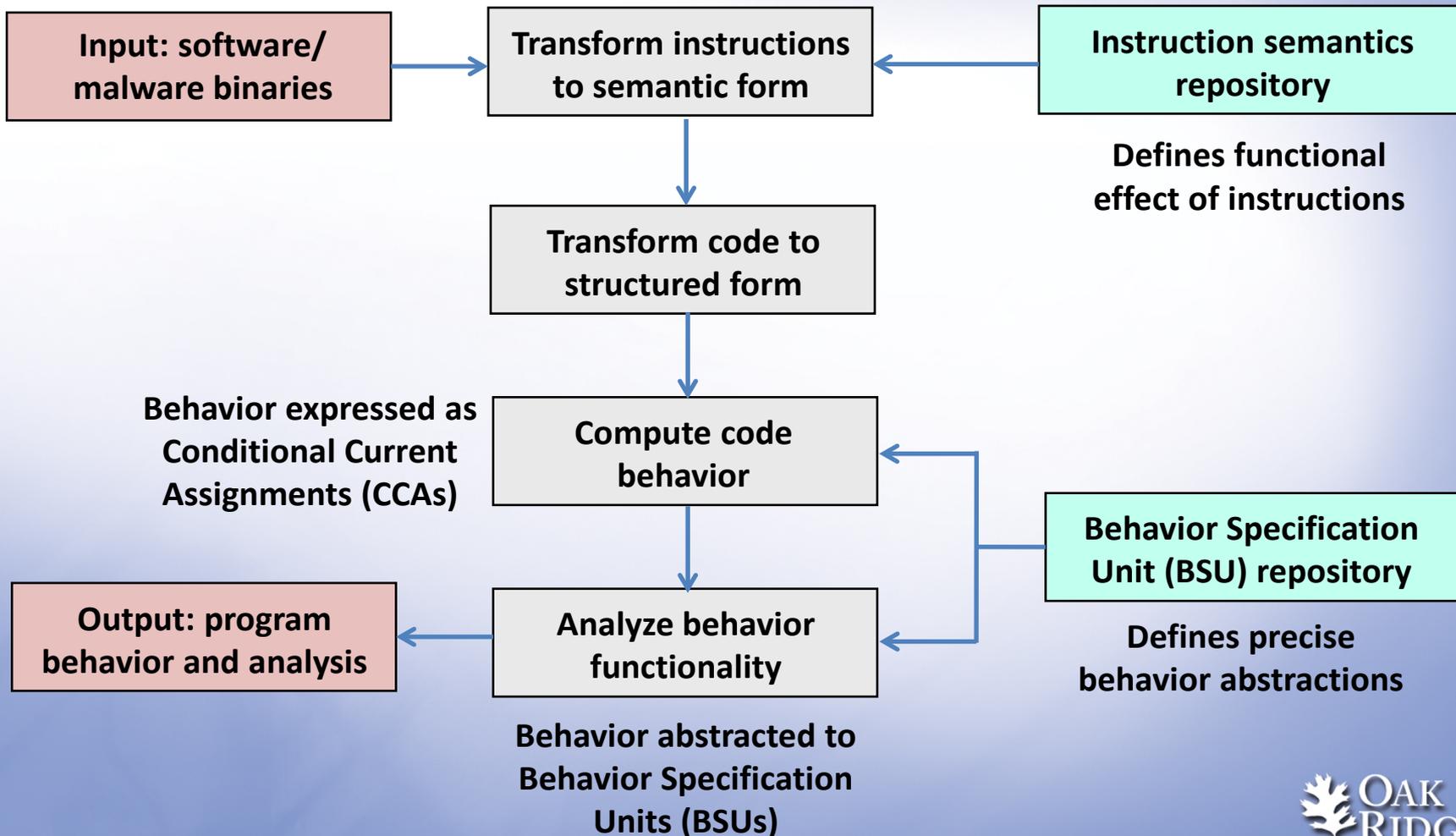
Computed behavior:

$\left. \begin{array}{l} \text{true} \rightarrow \\ x := y \\ y := x \end{array} \right\} \text{Conditional concurrent assignment (CCA)}$
 (swaps values of x and y)

(x, y integers;
machine precision aside)

Transforms procedural logic into non-procedural as-built specification

The Behavior Computation Process



An Example Behavior Computation

Cyberspace Sciences & Information Intelligence (CSIIR) Group

Computational Sciences & Engineering Division

Input code

Structured code

The screenshot displays a software interface with three main panes. The left pane, titled 'Assembly Listing', shows a table of assembly instructions. The middle pane, titled 'Structured Flowgraph', shows a control flow graph with a 'DO' loop. The right pane, titled 'Visual Computed Behavior', shows a hierarchical diagram of behavior. A yellow callout box points to the 'Visual Computed Behavior' diagram, and another yellow callout box points to the 'Structured Flowgraph'.

Address	Entry	Raw
-0x00000010	FX:SET_VALU	
0x00401000	add	01d8
0x00401002	mov	89c1
0x00401004	sub	29d9
0x00401006	mov	89cb
0x00401008	sub	29d8

```
graph TD
    DO[DO] --> S1[-0x00000010 FX:SET_VALUE $EIP = 0x00401000 (clear DF)]
    S1 --> S2[0x00401000 add DWORD EAX, DWORD EBX]
    S2 --> S3[0x00401002 mov DWORD ECX, DWORD EAX]
    S3 --> S4[0x00401004 sub DWORD ECX, DWORD EBX]
    S4 --> S5[0x00401006 mov DWORD EBX, DWORD ECX]
    S5 --> S6[0x00401008 sub DWORD EAX, DWORD EBX]
    S6 --> S7[label = exit]
```

Behavior

- Case 1:
 - Condition: true
 - State Update:
 - \$EAX: DWORD, \$EBX: DWORD
 - \$EBX: DWORD, \$EAX: DWORD
 - \$ECX: DWORD, \$EAX: DWORD

Computed behavior expressed as a Conditional Concurrent Assignment (CCA)

Customer Benefits: Understanding What Software Does

What is the Value Proposition?

- Understand behavior at machine speeds
 - Intellectual control over software
- Cheap, repeated validation
 - Confidence in mission readiness
- Know more than our adversaries
 - We understand behavior before they do

What are the Markets?

Cyberspace Sciences & Information Intelligence (CSII) Group

Computational Sciences & Engineering Division

- Malware detection and analysis
- Vulnerability detection
- Mobile device validation
- Rigorous software development
- Supply chain, anti-tamper analysis
- Forensic investigation
- Hardware analysis
- ...

Competition: The Behavior Computation Advantage

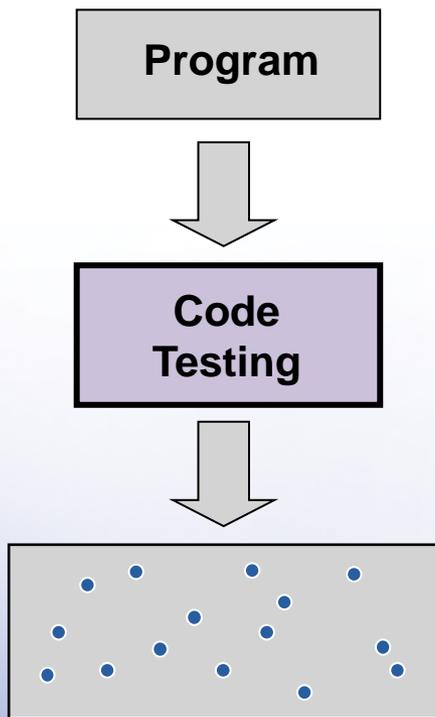
How is Software Analyzed Today?

- Human review:
 - Expensive, fallible
 - Execution testing
 - Expensive, inconclusive
 - Syntactic scanning
 - Cheap, inconclusive
-
- No good means for understanding full behavior

Comparing Technologies

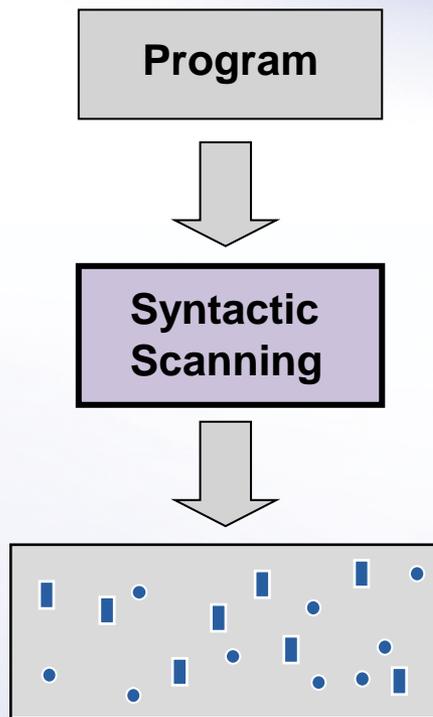
Cyberspace Sciences & Information Intelligence (CSII) Group

Computational Sciences & Engineering Division



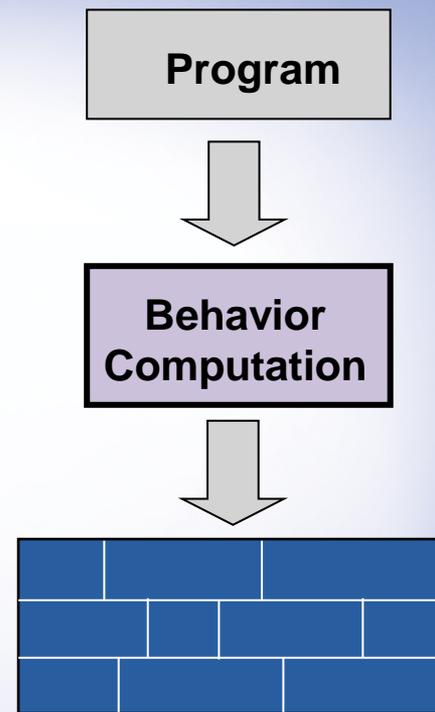
Population of executions:

Dots are test cases, area not covered is untested executions in the population.



Population of instructions:

Dots are recognized signatures, squares are problems that do not have signatures or are obfuscated.



Population of behaviors:

Disjoint partitions are behavior cases that cover the entire population.

Status: Evolving the Technology

- Next-generation Hyperion system under development
 - More powerful semantic processing
 - HPC potential for parallel computation
 - Addressing scale-up for larger programs
 - Currently Intel x86, other languages can be supported
 - Can customize for sponsor requirements

Demonstration:

Behavior computation for malicious code that attacked ORNL