GRAMMATECH

Tiffin and MGen: An Expressive Policy Language with Multiple Runtime Monitoring Tools

Zak Fry, Hajime Inoue, Cameron Swords, Wei-Cheng Wu, Lucja Kot

Outline

- Problem Statement and State of the Art
- Tiffin Policy Language
- MGen: Model Generator
- Deployment Use Cases
- Questions and Discussion

STATEMENT A: APPROVED FOR PUBLIC RELEASE

Problem Statement

- In spite of heavy investment, cyber-attacks are an increasing threat:
  - IP and information theft: $6 Trillion dollars of damage globally in 2021
  - Targeting US defense agencies, and tech companies per month
- Software: increasingly complex, mission-critical; use of inter-connected systems and deployment frameworks
- 0-day exploits, ransomware, insider threat all pervasive
- *Modern security practices can't keep up with current and emergent cyber-attacks*

STATEMENT A: APPROVED FOR PUBLIC RELEASE

State of the Art

Common Limitations and Restrictions

- Pattern matching known-bad patterns
  - Constant updates, limited 0-day capabilities
- Domain specificity, Limited scope
  - Specific system and fault/error/vulnerability types
- Limited Mitigation capabilities
  - Often only provide limited forensic evidence
- Require considerable human effort
  - Manual specification or instrumentation – scalability

STATEMENT A: APPROVED FOR PUBLIC RELEASE

Tiffin: Enforceable Runtime Policies

- **Mechanism**: define correct program behavior and state over critical points in individual programs
- **Autonomics**: both *identifies* and *mitigates* known and unknown attacks
- **Insights**:
  - Policies outlining *correct* behavior – **guard against unknown attacks**
  - *Per-program* models – **generalizable, customizable**
  - *Diverse* specification mechanisms – **flexible, performant**
  - Defines *context-specific* and *deployment-specific* behavior – **specificity**

STATEMENT A: APPROVED FOR PUBLIC RELEASE

Tiffin: Monitor Expressivity

Monitor Type	Preferred Instrumentation	Example Use Case
Invariant	DynamoRIO, Binary Rewriting	Check for valid server response data before transmitting information
State Machine	DynamoRIO, Source Rewriting (future)	Ensure authentication before protected actions occur
Memory Safety	Hypervisor Plugin	Guarantee consistent module write patterns to avoid trojans/overflows
Fuzzing Framework	DynamoRIO	Test for unexpected values in user-supplied data parser

STATEMENT A: APPROVED FOR PUBLIC RELEASE

Tiffin: Monitor Expressivity

Mitigation Type	Example Use Case
Alter Internal State	Change webserver response for malicious request
Skip Instructions	Avoid malicious write mid-execution, continue normal execution
Abort Program	Prevent information exfiltration during active attack
Print Message	Output helpful program state information for manual forensics
Return from Function (future)	Halt an infinite loop without stopping full program execution

STATEMENT A: APPROVED FOR PUBLIC RELEASE

Tiffin: Mitigation Specification

```
instr authenticate:
    loc: function entry “user_authenticate" “auth.c”
    args: cexpr string user_ip

instr protected_access:
    loc: line “serve.c" 522
    args: cexpr int req->ip

action block_request:
    loc: line “serve.c" 522
    update(cexpr int req->ret_code, 403)


state_machine validate_access[ip]:
    Start -> authenticate(id) -> protected_access(ip) -> Start
    Start -> protected_access(ip) { block_request() } -> Start
```

Instrumentation: function

Instrumentation: source line

Action: state change

Policy: state machine

GRAMMATECH

STATEMENT A: APPROVED FOR PUBLIC RELEASE

MGen: Deployable Runtime Checkers

- **Mechanism**: translate high-level specification to low-level, deployable runtime checkers

- Insights:
  - Push-button automation – **usability, rapid deployment**
  - Integrated binary analysis – **generalizable to arbitrary programs**
  - Multiple back-ends – **flexible, wider deployability**
  - Optional preliminary policy generation – **ease initial user burden, handle code evolution**

STATEMENT A: APPROVED FOR PUBLIC RELEASE

Monitor Generation

- **Mgen**: translate high-level spec to low-level checker
- Back-ends target various instrumentation frameworks
- Big challenge: how to convert source-level locations into runtime binary locations for instrumentation? *“Value of* `node->value` *at line* 67 *of* `stack.c`*”* is non-trivial.
  - DWARF parser and analysis

STATEMENT A: APPROVED FOR PUBLIC RELEASE

MGen: DWARF Deep Dive

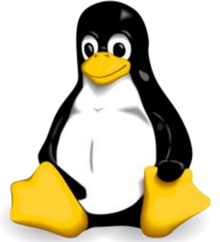
- GrammaTech python library, **Dwalin**, to extract variable locations from DWARF debug symbols
- Handles C/C++ binaries, using monitor source locations to **resolve instrumentation info** needs
- **Future**: direct binary analysis and/or source instrumentation to lift DWARF requirement

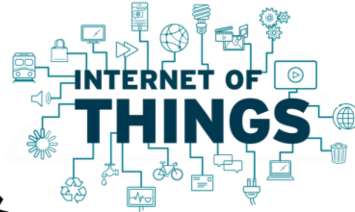
```
python> dwarf_info = make_dwarf_info(
      'nginx_x64', {}, IsaEnum.X64, debug=0)

python> dwarf_info.lookup_c_expr(
      'req->ip', 'nginx_64',
      'serve.c', '0x4f1a40')

FieldOffset(
    base=Location(
        location='%rbp + 16 - 176',
        type_addr='0x429',
        type_size='0x80',
    ),
    offset=Constant(value=0),
    type_addr='0x419',
    type_size='0x80',
    deref=False
)
```


STATEMENT A: APPROVED FOR PUBLIC RELEASE

Use Cases and Deployments

- Monitoring and mitigation for:
  - General purpose Linux apps
  - UEFI/firmware
  - Deployed IoT devices
- Automated support for fuzz testing





STATEMENT A: APPROVED FOR PUBLIC RELEASE

Use Cases: Considerations

- Linux apps: wide range of functionality
- UEFI: limited monitoring options
- IoT: resource constraints, connectivity
- Fuzz testing: additional information needs

STATEMENT A: APPROVED FOR PUBLIC RELEASE

Use Cases: Solutions

- Linux apps: diverse, flexible specification mechanisms
- UEFI: bespoke hypervisor clients, memory introspection
- IoT: binary rewriting, focused policies
- Fuzz testing: language additions, specialized clients

STATEMENT A: APPROVED FOR PUBLIC RELEASE

Deployment: Multi-Layer Security Solution

Deploy ***local monitor policies*** to ***running applications***. Policies watch for malicious behavior and carry out local ***reflex responses***.

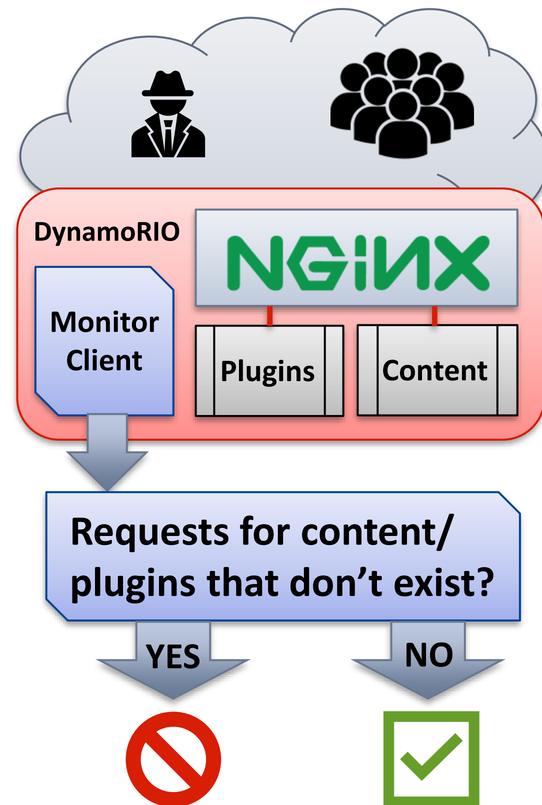
Report monitor events to ***"big picture" reasoning engine*** to track overall system health; detect additional and multi-program attacks. Engine carries ***secondary responses***.

Long-term and recurrent problems result in **longer-term responses**, e.g., ***automated patch generation***, ***manual remediation***.

STATEMENT A: APPROVED FOR PUBLIC RELEASE

NGINX Webserver Example

- **Problem**: Bots probe public servers looking for known-vulnerable modules and secure content
- **Autonomic Solution**:
  - Monitor: Use Tiffin-internal variable per-IP to count accesses to non-existent pages/content
  - Mitigation: Block individual or ranges of IPs from initiating requests entirely



STATEMENT A: APPROVED FOR PUBLIC RELEASE

Protecting Firmware

Initial Semantic Modeling and Checking technology is tailored to firmware:

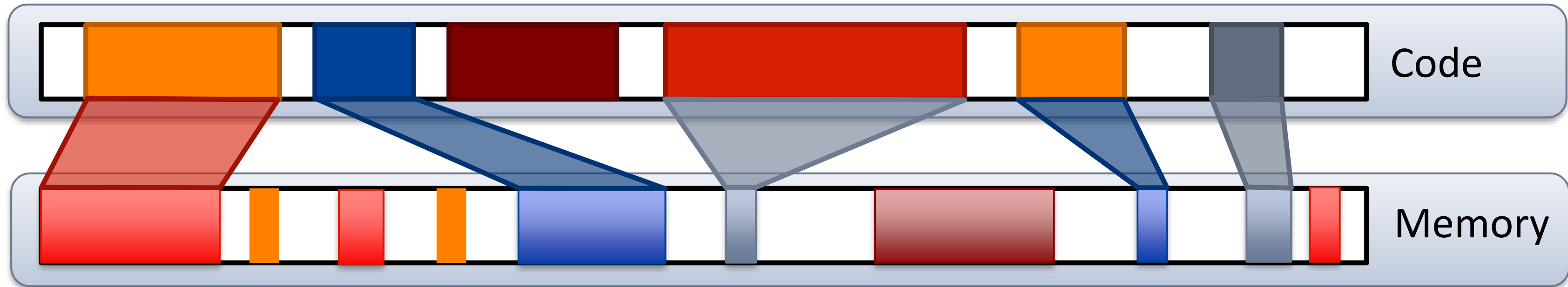
- No operating system
- Single address space
- Initially inspired by Region-Based Write Access Control (RBWAC), by Shapiro (Dartmouth)

UEFI is firmware:

- Used on PCs, but also: printers, routers, switches, storage devices, phones
- Interface between OS and bare hardware
- BIOS: Basic Input/Output System
- UEFI: Unified Extensible Firmware Interface
  - IA-32, ARM32, x64, AArch64

STATEMENT A: APPROVED FOR PUBLIC RELEASE

Memory Policies



Policy Strategy

- Policy breaks down into tuple of <what, where>
  - **What** – What substage (code) is doing the writing
  - **Where** – Which region (memory) is written to
- Policy is:
  - Each substage policy is a set of tuples: <code, memory>
  - File accesses
  - Allowed transitions between substages
  - **Prevent driver loads unless they match expected metadata**

STATEMENT A: APPROVED FOR PUBLIC RELEASE

IoT Security in the News

Watch A Tesla Have Its Doors Hacked Open By A Drone

Thomas Brewster Forbes Staff
Cybersecurity
Associate editor at Forbes, covering cybercrime, privacy, security and surveillance.

Hackers leave Finnish residents cold after DDoS attack knocks out heating systems

- The attack is believed to have lasted for a week, starting in late October and ending on 3 November.

Cybersecurity

Hackers Breach Thousands of Security Cameras, Exposing Tesla, Jails, Hospitals

By William Turton
March 9, 2021, 4:32 PM EST *Updated on March 9, 2021, 8:22 PM EST*

FDA confirms that St. Jude's cardiac devices can be hacked
by Selena Larson @selenalarson

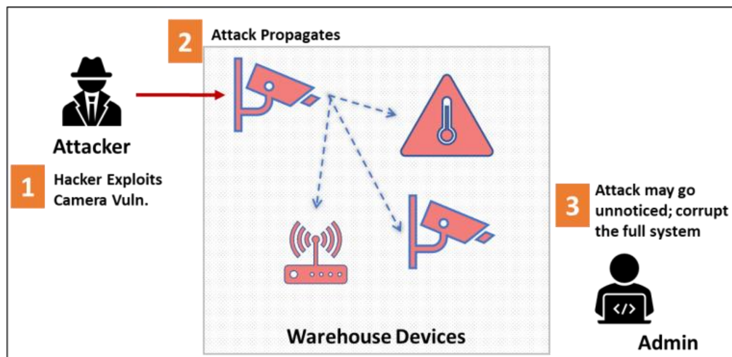
Hacker terrorizes family by hijacking baby monitor

Hackers Remotely Kill a Jeep on the Highway—With Me in It

Ripple20

STATEMENT A: APPROVED FOR PUBLIC RELEASE

IoT Protection



Unsecured

- Attack propagates to all nodes
- Potential for information leak or catastrophic failure
- May go unnoticed, if disguised
- No record of events, infection

IoT-specific deployment:

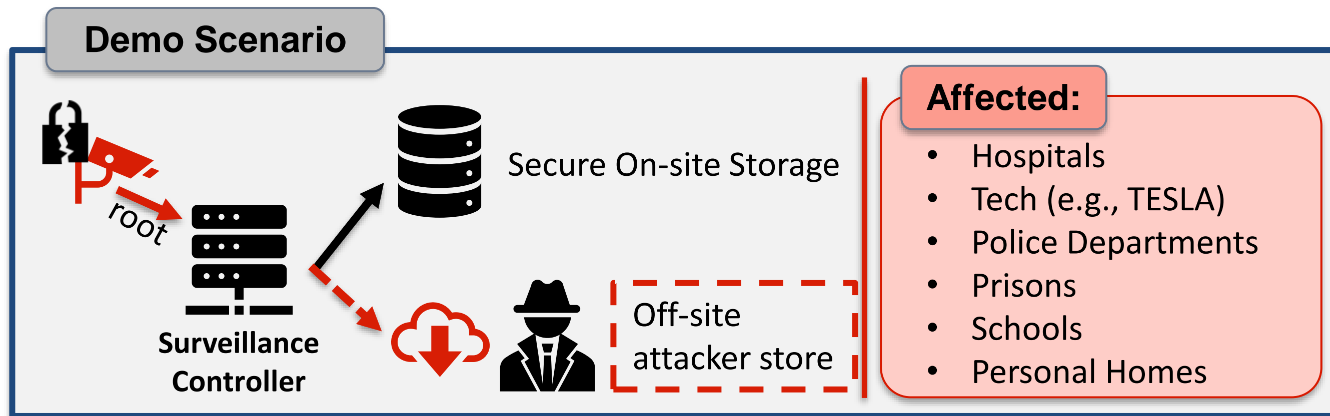
- **Binary instrumentation** for size, weight, and power constrained devices
- Secured, distributed **storage and communication** for forensics

Insights:

- Devices with limited connectivity and on the network edge often require **reflex responses** for effective threat mitigation
- IoT devices typically have a **narrow effective functionality**, well suited for our policies

STATEMENT A: APPROVED FOR PUBLIC RELEASE

Evaluation



- Attack: rewrite surveillance config to *additionally* store footage offsite
- IoT protection: Identify “normal” behavior
  - Correctly identifies the single IP the controller sends feeds in practice
  - Notices immediately when the attack leaks feeds to new IP; block’s traffic
  - Allows users to immediately shut down node and reconfigure

Policy-Based Component Fuzzer

So far, policies protect at runtime – better to discover bugs, malware, or misconfiguration during testing

- Use Tiffin policy to guide fuzzing with policies, instead of just checking correctness
- Infer metric against policy to inform mutator engine: attempt to descend toward a policy violation (instead of using coverage, etc.)

STATEMENT A: APPROVED FOR PUBLIC RELEASE

Example: ArduPilot

ArduPilot bug*:

- Speed variable should not drop below a constant (inferred from reading documentation)
- Straight-forward to express this in policy and detect

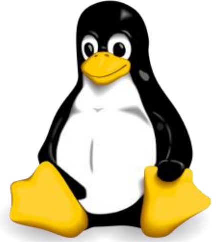
Other applications: malicious implants from supply chain vulnerabilities

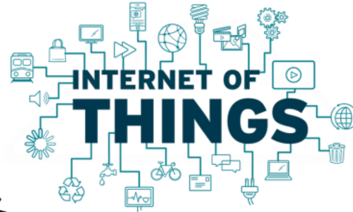
*Kim et al. *RVFuzzer: Finding Input Validation Bugs in Robotic Vehicles through Control-Guided Testing.* USENIX Security 2019

```
instr fuzz_speed:
        loc: line "AC_WPNav.cpp" 204
        fuzz_mode: GRADIENT
        # fuzz speed_cms parameter (index 0),
        # from 0 to 300
        fuzz_params: int speed_cms 0 0 300

instr speed_result:
        # At the end of set_speed_xy function
        loc: line "AC_WPNav.cpp" 20
        args: cexpr float speed_cms,
              cexpr float _wp_speed_cms,
              cexpr int WPNAV_WP_SPEED_MIN

    invariant speed_update:
        speed_cms > WPNAV_WP_SPEED_MIN
```

Questions?

- Monitoring and mitigation for general purpose Linux apps
- UEFI/firmware monitoring and response
- Autonomic protections for IoT
- Automated support for fuzz testing





STATEMENT A: APPROVED FOR PUBLIC RELEASE

State of the Art

- Domain specificity
  - MaC (Viswanathan & Kim 2004) – reactive systems in terms of ω-languages
  - Volz et al. (2011) – Monitoring semantics for distributed complex event processing
- Limited Scope
  - MOSAICO (Muccini 2007) – Monitoring adherence to architectural specifications
  - PROPEL and exceptions (Phan et al. 2008) – Specifying and monitoring exceptional behavior
- Underlying assumptions about target systems
  - mlCCL (Baresi & Guinea 2013) – Multi-layered software-as-a-service architectures
  - Gan et al. (2007) – Runtime Monitoring of Web Service Conversations in IBM's WebSphere Integration Developer
- Human Effort required
  - ReMinds (Vierhauser et al. 2016) – manual specification of instrumentation and hooks

STATEMENT A: APPROVED FOR PUBLIC RELEASE

ARTCAT Toolchain

C/C++ Source Code
Compiler
Application
Instrument & Deploy
mgen
Programmer
Monitor Policy Generation
Manual Editing
Dynamic Analysis
Static Analysis
Runtime Policy

Develop Monitors & Responses
Offline

Running System
Operating System
Deployment Daemon
Stealthy Monitor
DynamoRIO
(model + reflex responses)
Application

Events
Reasoned Responses

Instrumentation Deployment
Online, Synchronous & Asynchronous Monitoring and Responses

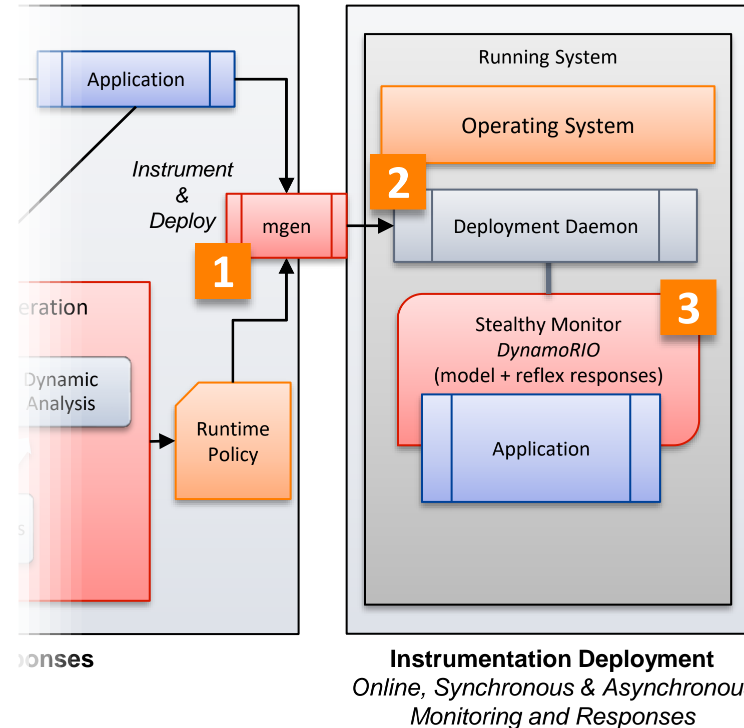
Messaging Broker And Coordinator
Events
High-level response tasks
System Reasoner
Status and events
Web UI
Repair requests
Repairs
Software Repair

Reasoning Engine
Online, Asynchronous Responses

STATEMENT A: APPROVED FOR PUBLIC RELEASE

ARTCAT Deployment

1. The mgen monitor generation tool converts instrumentation and monitors into DynamoRIO clients
2. ARTCAT deploys clients to perform binary instrumentation at runtime
3. Deployment occurs through deployment daemon and DynamoRIO runner
   - Inspects program state
   - Runs monitors
   - Reports events
   - Carries out reflex responses

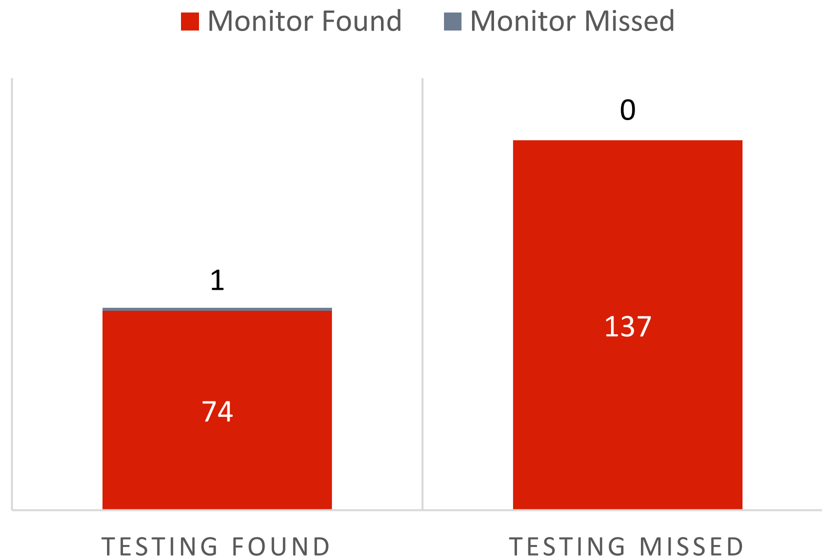


STATEMENT A: APPROVED FOR PUBLIC RELEASE

Accuracy

Monitors caught 99.5% of errors; tests caught 35%

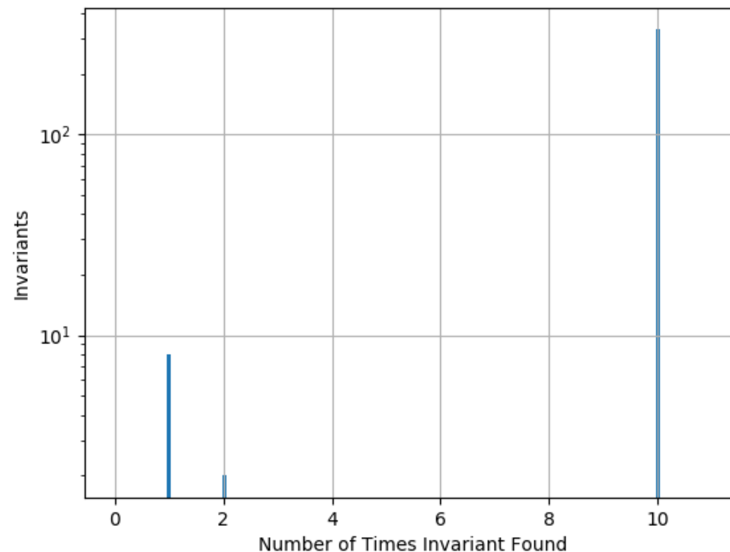
- How effective is ARTCAT at finding policy violations?
- Experiment: CGC water tank controller program. Generated policy and generated variants with errors run on a test suite, plus under monitoring
- Conclusion: domain-focused monitors cover many cases that testing may miss



STATEMENT A: APPROVED FOR PUBLIC RELEASE

Policy Generation

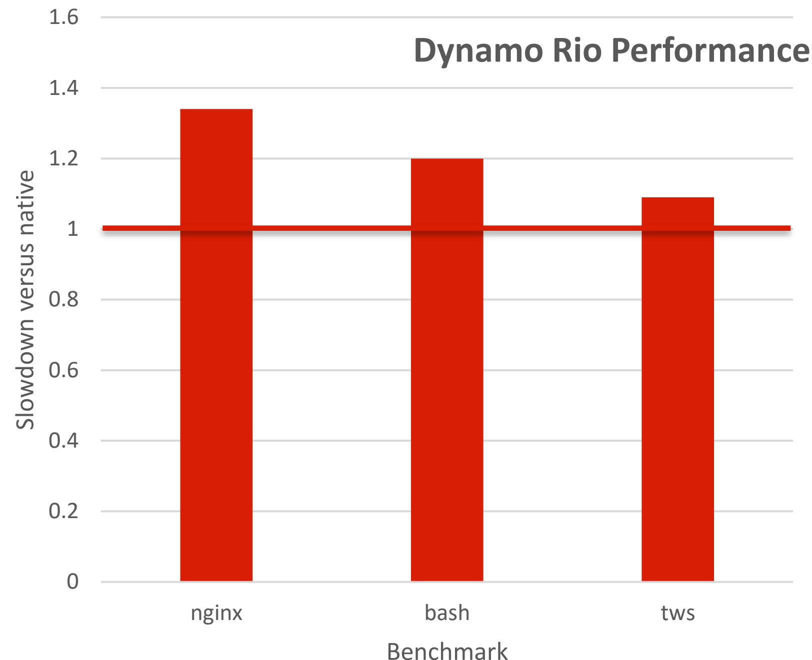
- Invariant generation produces realistic fixpoints.
- Experiment run on tar zip utility.
  - 16k input pool, 1024 per run
  - "Diverse" inputs: create compressed files, unzip compressed files, etc.
  - 341 invariants found, 331 in every run.
- Automated filtering automatically removes low-quality invariants
- Interactive UI allows for fast human review and re-testing to lower performance impact, improve accuracy.
- Performance on the scale of minutes/hours, not days/weeks.



STATEMENT A: APPROVED FOR PUBLIC RELEASE

Runtime Overhead

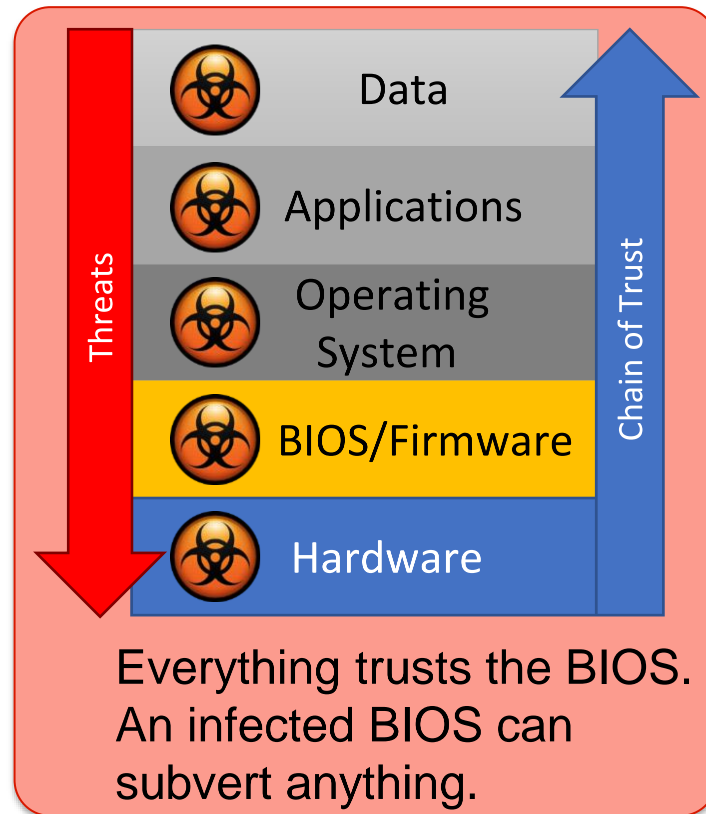
- Experiment: run applications under monitoring
- Conclusion: viable for long-running applications (bash, servers, etc.)



STATEMENT A: APPROVED FOR PUBLIC RELEASE

Protecting BIOS Firmware from Malware

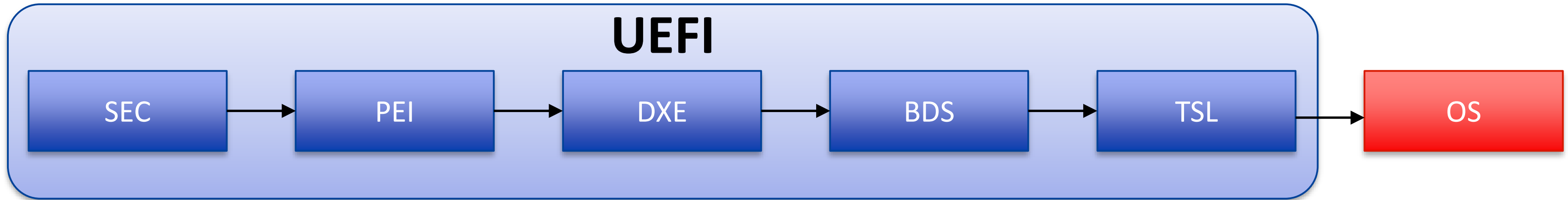
- Goal: Protect UEFI, which is the BIOS type on almost all COTS PC systems
- UEFI BIOS initializes hardware before operating system loads
- Composed of hundreds of components
  - May come from multiple third parties
  - Source is not available
  - Devices may inject their own drivers ("Option ROMs")
- Monolithic – each component can access the entire system
- Malware in BIOS can subvert any operating system, VM, or container, running above it



STATEMENT A: APPROVED FOR PUBLIC RELEASE

UEFI Boot

- **SEC**: Security (code signature checking)
- **PEI**: Pre-EFI Initialization (CPU and Memory Initialization)
- **DXE**: Driver Execution Environment (Device initialization)



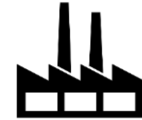
- **BDS**: Boot Device Select (Select operating system)
- **TSL**: Transient System Load (Load operating system)

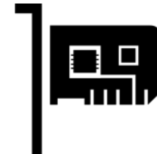
STATEMENT A: APPROVED FOR PUBLIC RELEASE

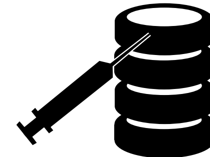
UEFI Threats and Vulnerabilities

- Threats may come from
  - **Supply chain – Malicious code inserted into legitimate drivers**
  - Insider attacks – Additional drivers or misconfiguration
  - Code injection – Exploitation of bugs at UEFI level
- Survive OS and disk replacement
- UEFI Malware Examples
  - ThunderStrike (deceived insider attack)
    - Attacked Mac laptops
    - Spread by “evil” cable
    - Option ROM attack
  - LoJax (code injection)
    - From Fancy Bear/Sednit/Apt28 group (affiliated with GRU)
    - Mimics LoJack for laptops
    - Installs OS backdoor

LOJACK







STATEMENT A: APPROVED FOR PUBLIC RELEASE

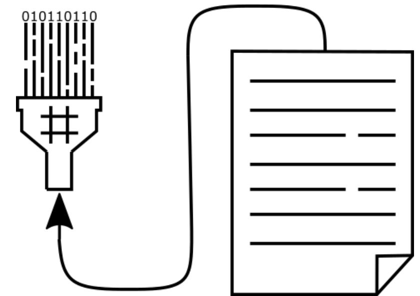
LoJack, LoJax, and MosaicRegressor

	UEFI	Windows Early Boot	Windows OS Running	
LOJACK Absolute Computrace (2006)	UEFI Drivers →	autochk.exe →	rcpnetp.exe →	Remote Shell
FANCY BEAR LoJax (2018)			User-space Exploit (→ UEFI Drivers)	
	UEFI Drivers →	autochk.exe & rcpnetp.exe →	Remote Shell and Other Payload	
MosaicRegressor (2020)			User-space Exploit (→ UEFI Drivers)	
	UEFI Drivers →	IntelUpdate.exe →	Remote Shell and Other Payload	
GRAMMATECH	UEFI Drivers →		OMS Startup →	Greater Mimic

STATEMENT A: APPROVED FOR PUBLIC RELEASE

State of the Art

- **Current solutions are focused on code signing**
  - Secure Boot – no unsigned code allowed
    - Signature checks for UEFI, boot loader, OS
  - Intel Trusted Execution Technology (TXT)
    - Can dynamically check local configuration
    - Provides remote attestation of boot
- **Current solutions do not**
  - Prevent supply chain attacks
  - Provide forensic information
  - Permit responses



STATEMENT A: APPROVED FOR PUBLIC RELEASE

UEFI Boot with SySense and Aperture

Detection & Response

Monitoring

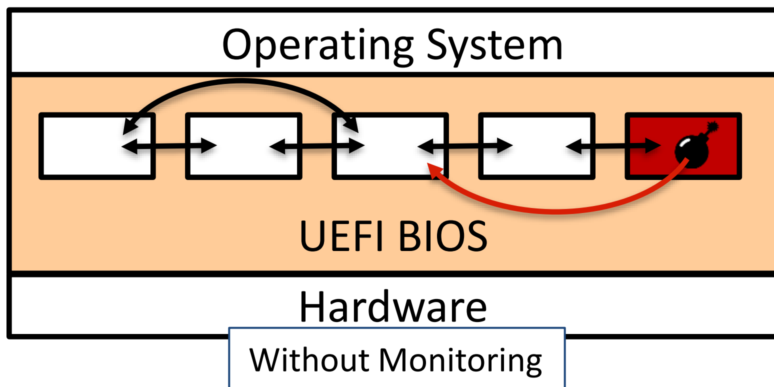
SEC → PEI → SySense & Aperture → DXE → BDS → TSL → OS

Training

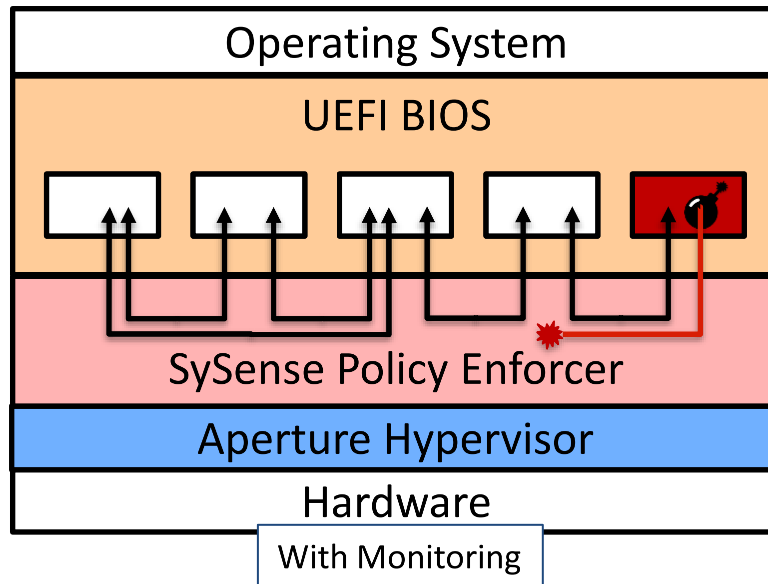
LOGS → Policy Inferrer → SySense & Aperture

STATEMENT A: APPROVED FOR PUBLIC RELEASE

Apply Principle of Least Privilege to UEFI



Without Monitoring



With Monitoring

- Divide UEFI into separate logical **substages**
- Use **Aperture** hypervisor from Clear Hat Consulting to monitor and protect instrumentation
- Monitor interactions between substages to construct **policy**
- **Enforce** policy to prevent substages from interfering with each other

STATEMENT A: APPROVED FOR PUBLIC RELEASE