Resilience of Cyber-Physical Systems and Technologies

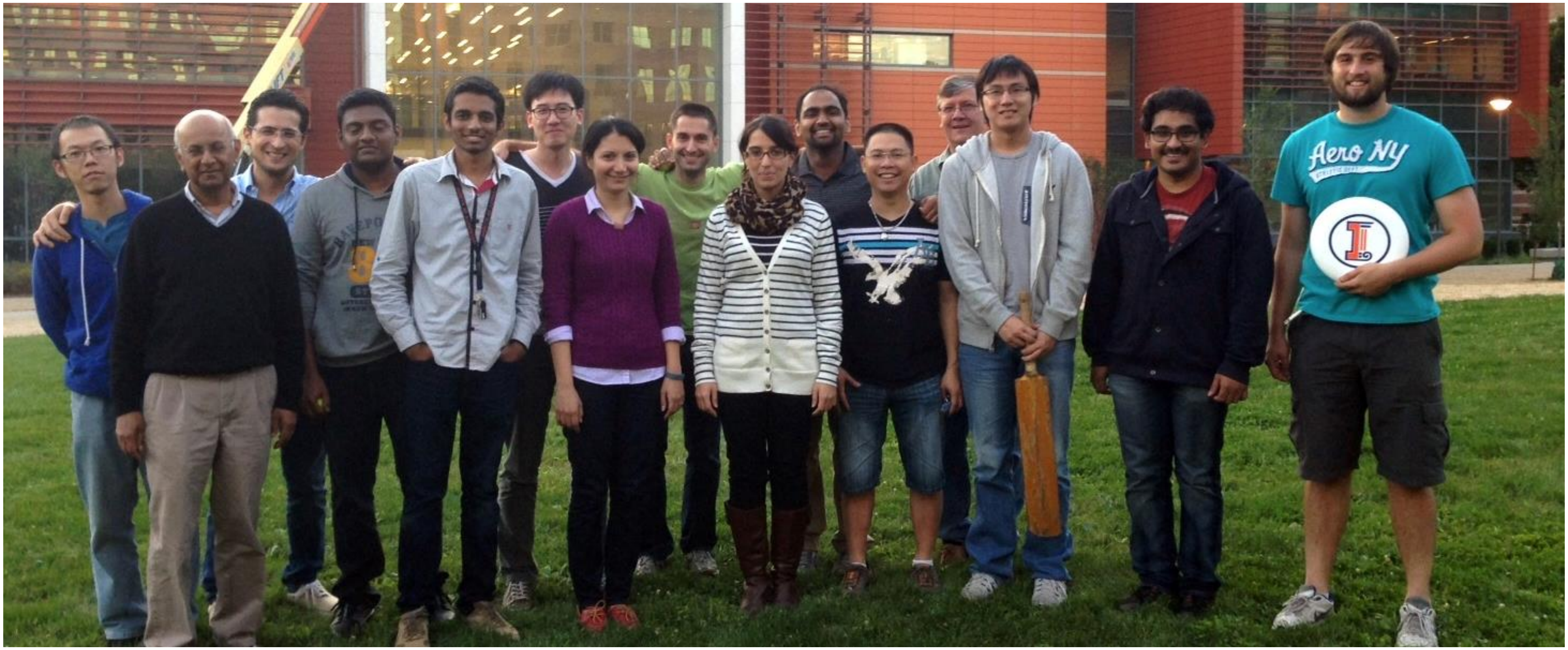
Zbigniew Kalbarczyk and Ravishankar K. Iyer

University of Illinois at Urbana-Champaign
Email: {kalbarcz, rkiyer}@Illinois.edu

ECE ILLINOIS

ILLINOIS

Depend Research Group

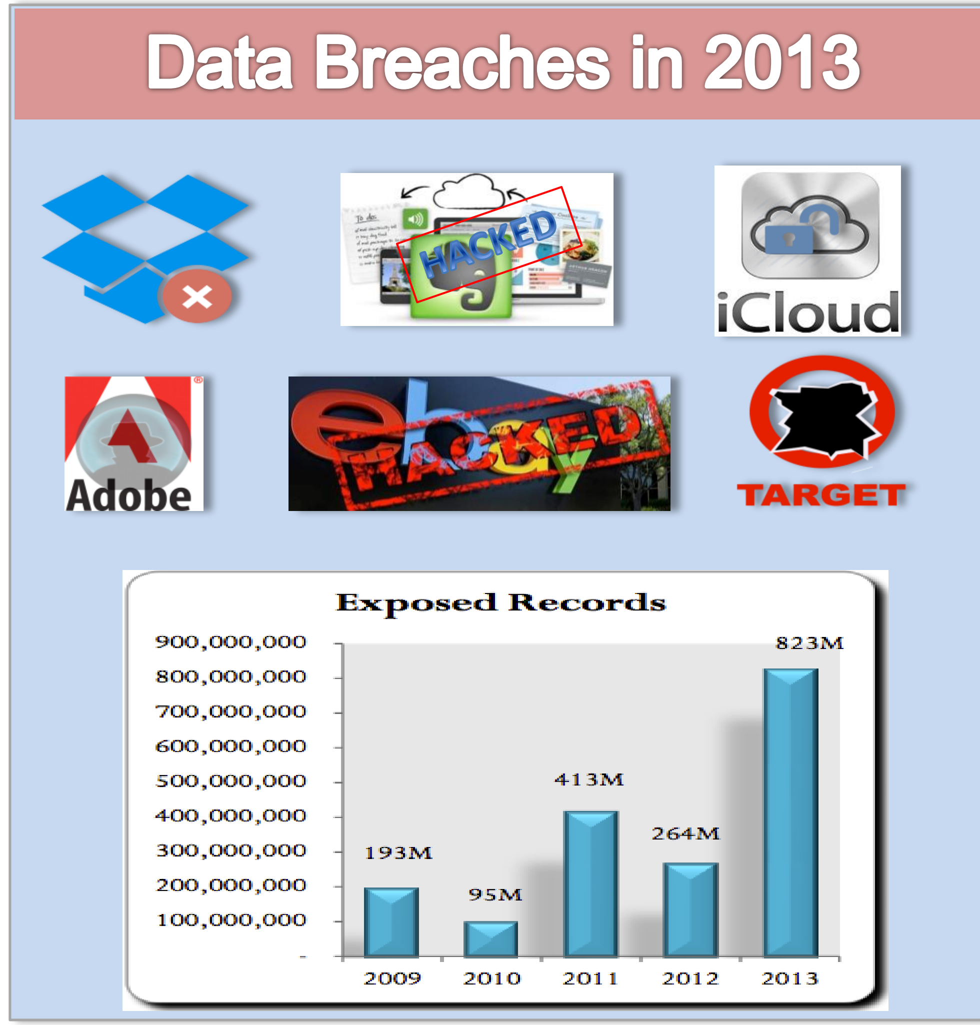


Group Retreat Fall 2014

Building a system is hard…

But maintaining *Reliability & Security* is even… HARDER

Failures and Attacks are Inevitable



Source: An Executive's Guide to 2013 Data Breach Trends, by Risk Based Security

Design for Resiliency

- A ***resilient system*** is expected to maintain an acceptable level of service in presence of internal and external disturbances

- ***Design for resiliency*** is a multi-disciplinary task that brings together experts in security, fault tolerance, human factors, and others

- ***Achieving resiliency*** requires mechanisms for ***efficient monitoring, detection, and recovery*** from failures due to malicious attacks and accidental faults with minimum negative impact on the delivered service

While is this hard?

- ***Design and assessment***
  - systems become untrustworthy due to a combination of: human failures, hardware faults, software bugs, network problems, and inadequate balance between the cyber and the physical systems e.g. the network and control infrastructures
- ***Delivery of critical services***
  - cyber-physical systems (e.g., energy delivery, transportation, communications, Heath Care) are expected to provide uninterruptable services
- ***Interdependencies among systems***
  - resiliency of one system may be conditioned on availability of another system, e.g.,
    - resiliency of the transportation system may heavily depend on the robust operation of energy delivery infrastructure,
    - human-in-the-decision-loop – role of human intelligence in system remediation, service restoration and recovery

Our Approach: Continuous Monitoring

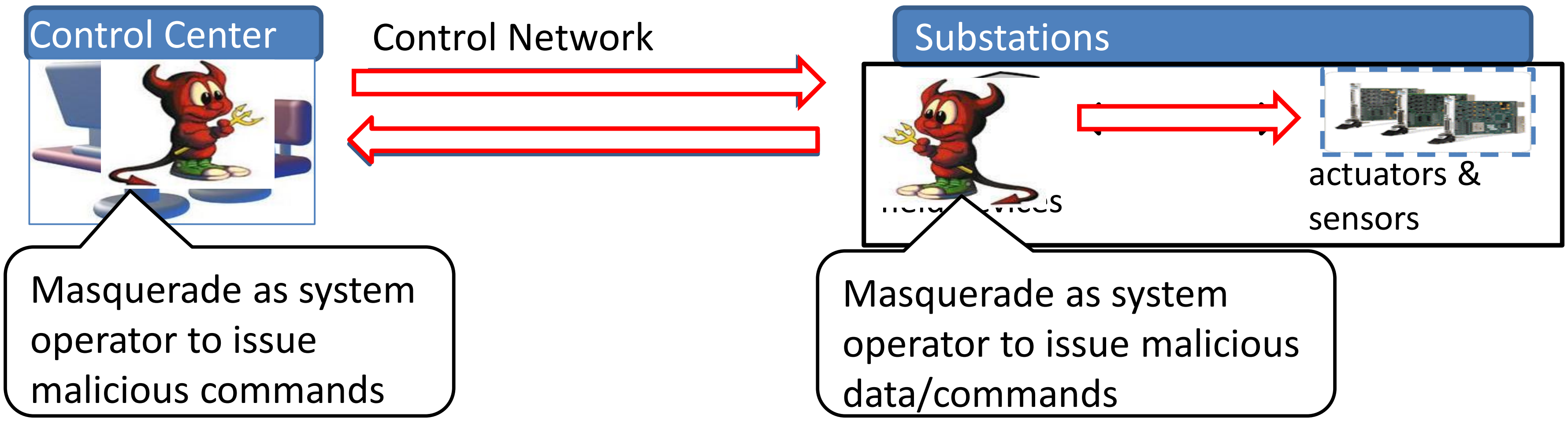
- **Coverage vs. Cost tradeoffs**
  - Detectability/Latency/Root of trust
  - Human/Resources
- **Methods**
  - Active vs. passive monitoring
  - Monitoring coordination
  - Automated reasoning
  - Domain aware techniques

Agenda

- **Leveraging power grid semantic**
  - Integrate power system analysis into network monitoring
- **Virtual machine monitoring**
  - Active vs. Passive
- **Probabilistic inference on security logs**
  - Monitor coordination
  - Automated reasoning

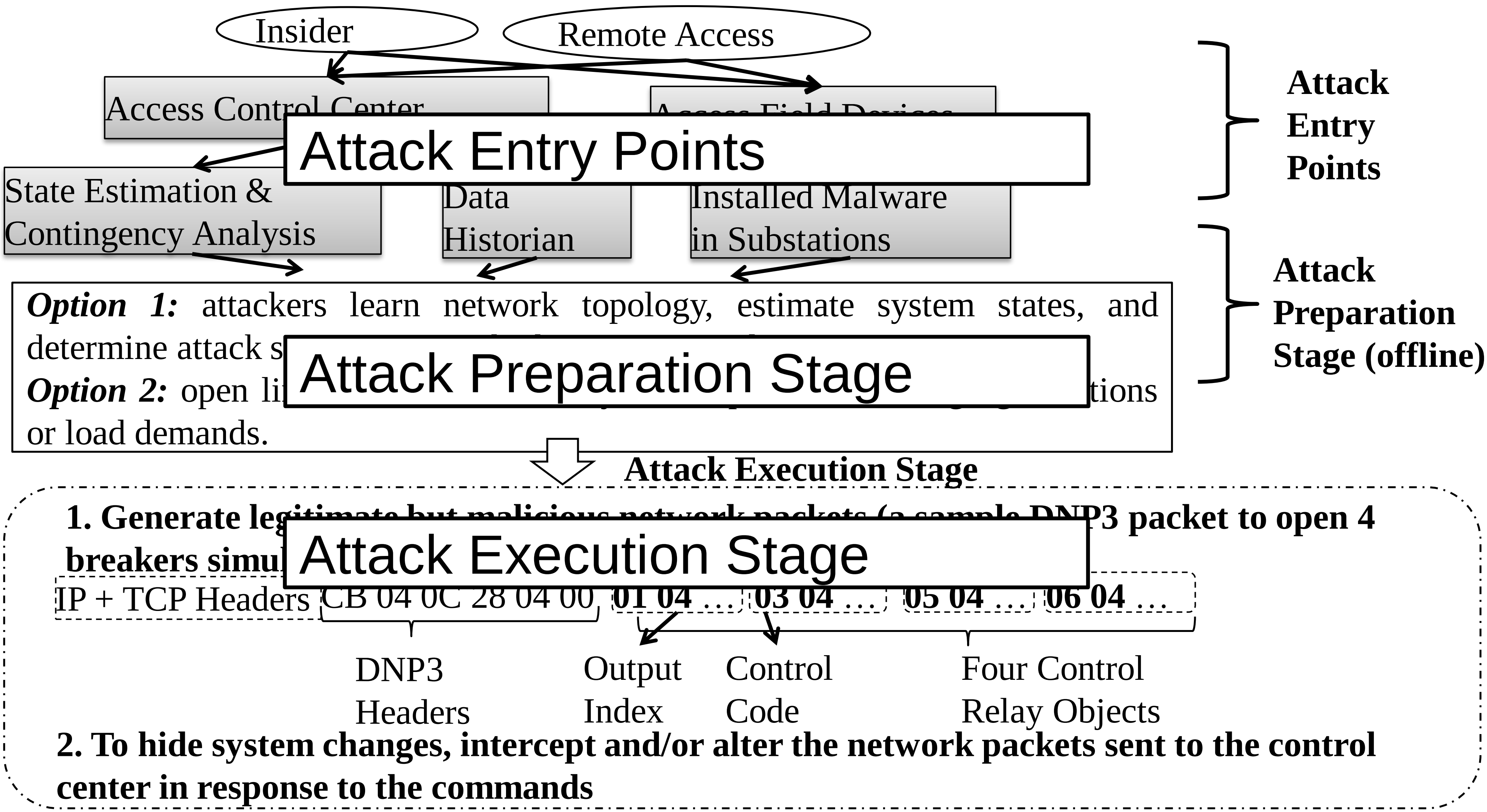
LEVERAGING POWER GRID SEMANTIC

Cyber Threats in Power Systems



- **SCADA (*Supervisory Control And Data Acquisition*)** system
  - Monitor and control geographically distributed assets in industrial control environment, e.g., power grid, gas pipeline, etc.
- Modern SCADA systems integrate commercial computer systems and network
  - Compromise in control center, e.g., stolen credentials and software vulnerability
  - Compromise in substation, e.g., vulnerability in intelligent devices

Example Scenario of Control-related Attack



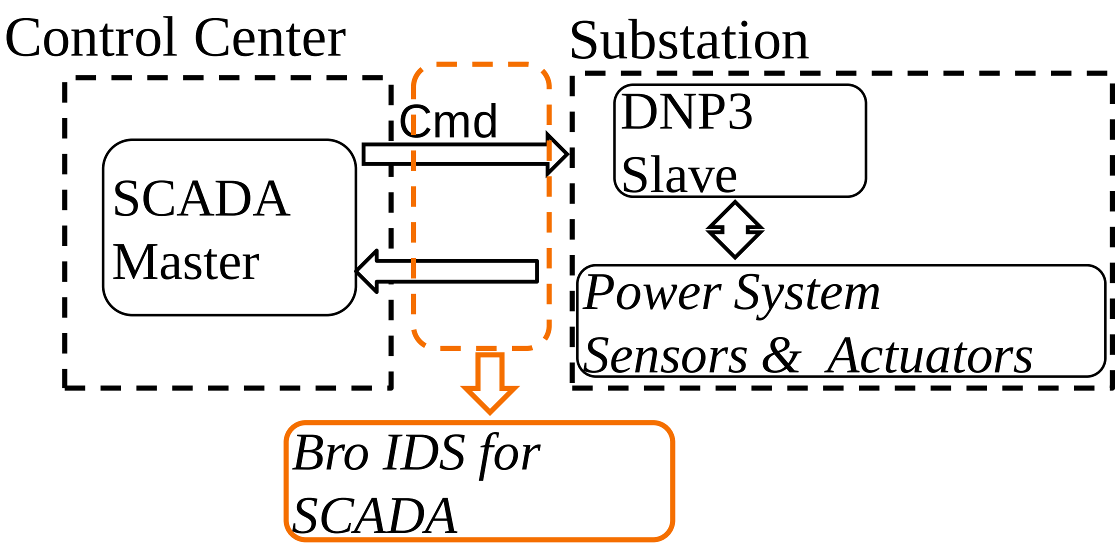
Why Is Detection of Control-related Attacks a Challenge?

- Hard to detect based solely on power systems' electrical states
  - Traditional contingency analysis considers low-order incidents, i.e., the "*N-1*" contingency
  - Traditional state estimation is performed periodically, detecting attacks after physical damage
  - Measurements may be compromised
- Hard to detect based solely on the network intrusion detection systems
  - Commands can be encoded in correct syntax
  - Not detectable by traditional network intrusion detection systems (IDS)

Detection Mechanism

- Combine system knowledge on both cyber and physical infrastructure in the power grid
  - Integrate network monitoring with look-ahead power flow analysis
- Detect malicious commands at their *first appearances*, instead of identifying power system's physical damage after the fact

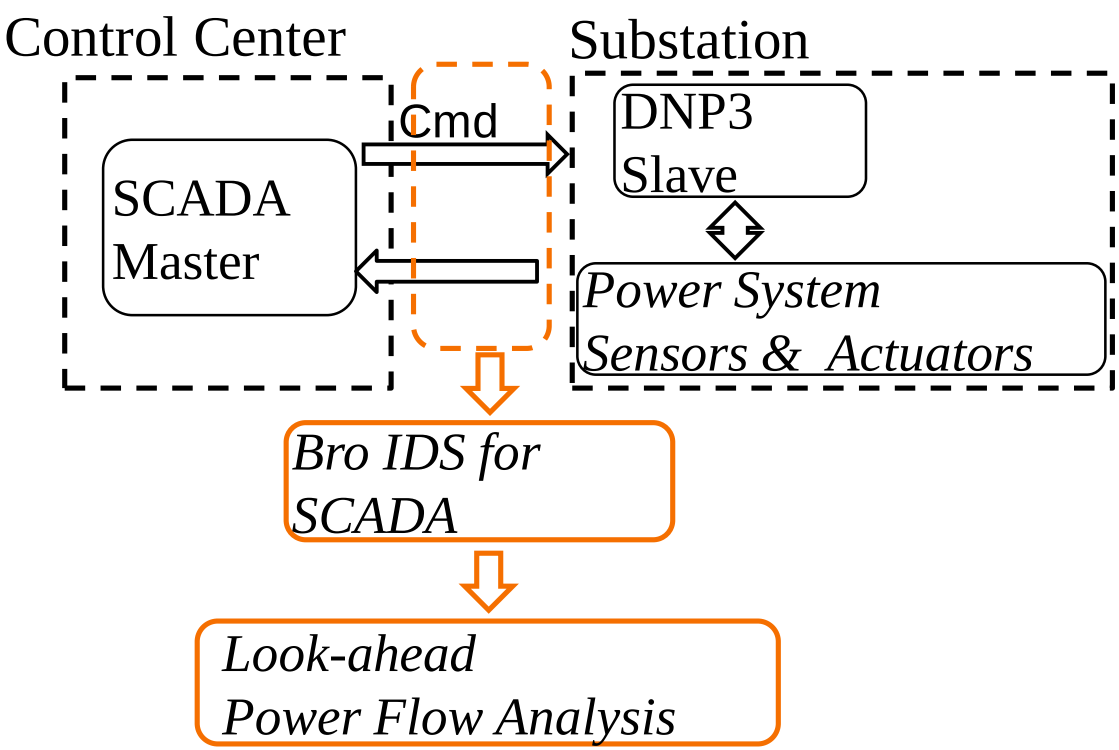
Example Approach: Adapting IDS for SCADA



Cyber Infrastructure

- Adapt specification-based IDS (e.g., Bro) for SCADA systems
  - Detect unexpected network activities based on deviation from security specifications, e.g., protocol definition
- Develop SCADA protocol (e.g., DNP3) analyzer and integrate with IDS system
  - Intercept SCADA commands at runtime

Example Approach: Bring Semantic Analysis



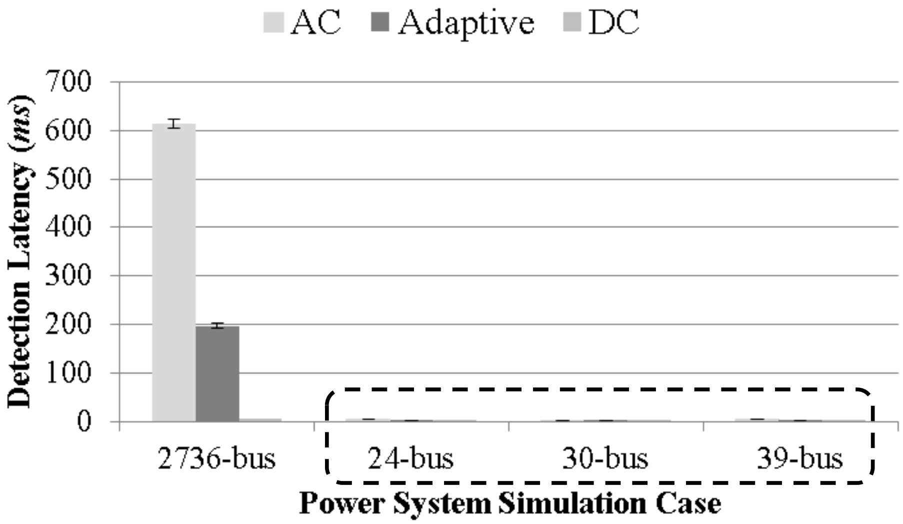
Physical Infrastructure

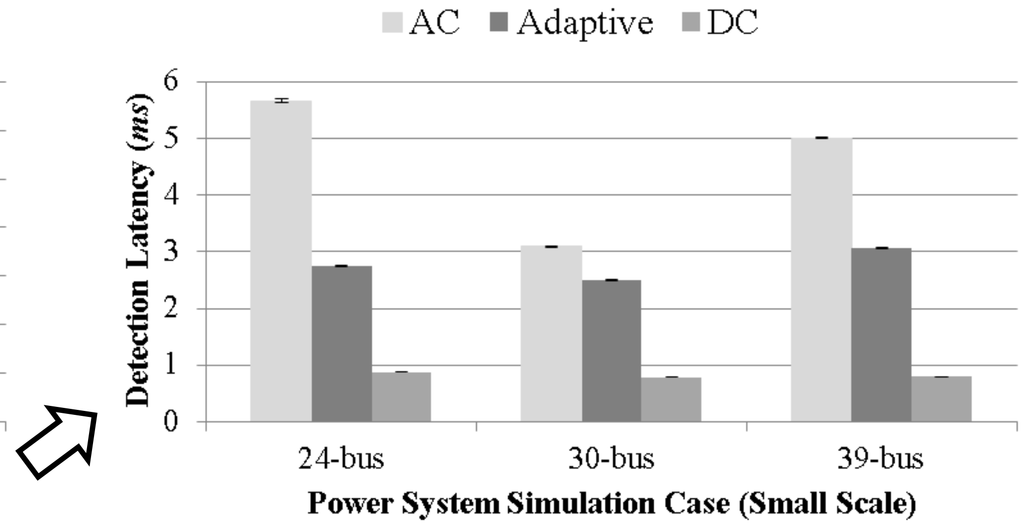
- Identify control commands from the network
- Invoke look-ahead power flow analysis
- Adapt power flow analysis algorithm for quick (low latency) detection

Evaluation: Detection Accuracy and Latency

- Very high detection accuracy: low false positive and false negative rate << 1%

- Low detection latency





Running on a PC with Intel i3 (3.07 GHz) quad-core and 8 GB memory and Ubuntu 12.04

Summary

Attack Model	• Control-related attack in the context of power grid

Detection	• Intercept commands
	  • Use network analyzer for SCADA protocols (DNP3) and integrate it with the IDS
	• *Proactively* estimate commands' execution consequences
	  • Invoke rapid adaptive power flow analysis

Response	• Intrusion response:
	  • use *reclosing logic* in modern relays
	  • use software-defined networking technology (SDN) to allow flexible responses to attacks

Evaluation	• Simulation of power systems with different scales • Detection performance, i.e., latency and accuracy • Integrated simulation of SDN network and power system

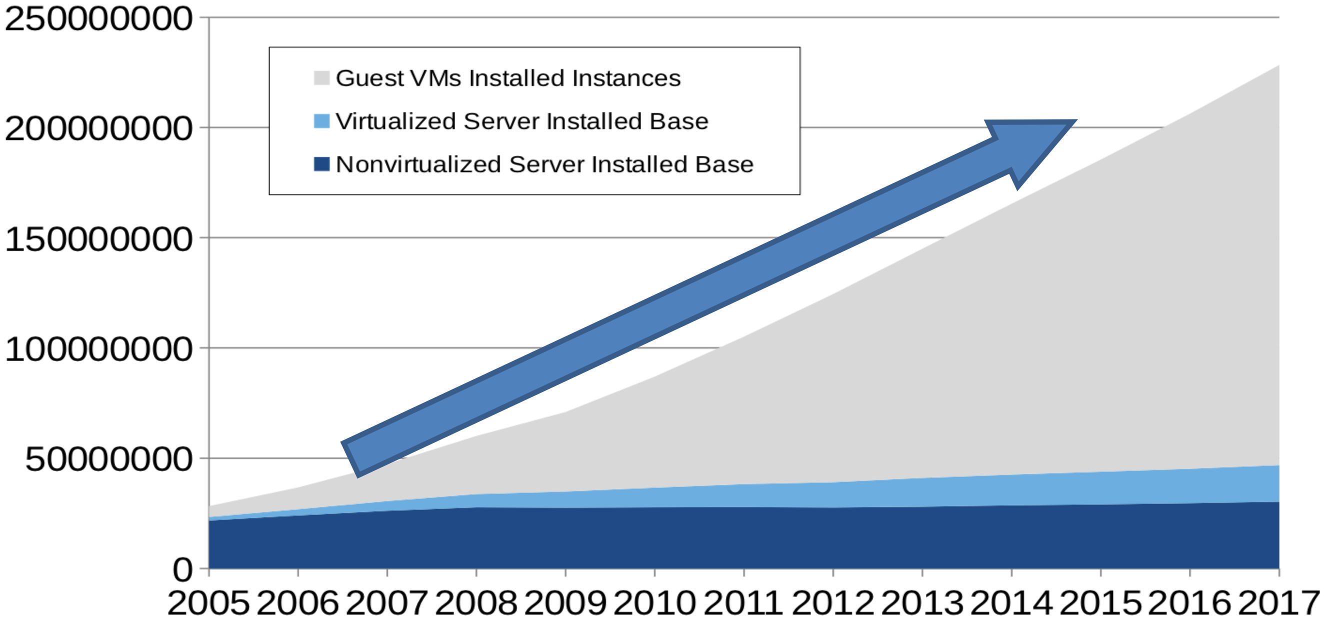
VIRTUAL MACHINE MONITORING

Server Virtualization Trend

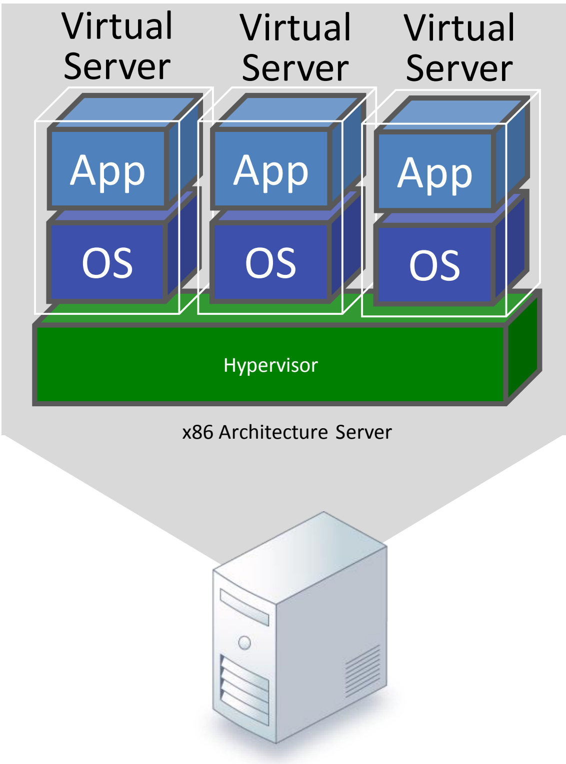
51%

x86 servers were **virtualized** in 2012

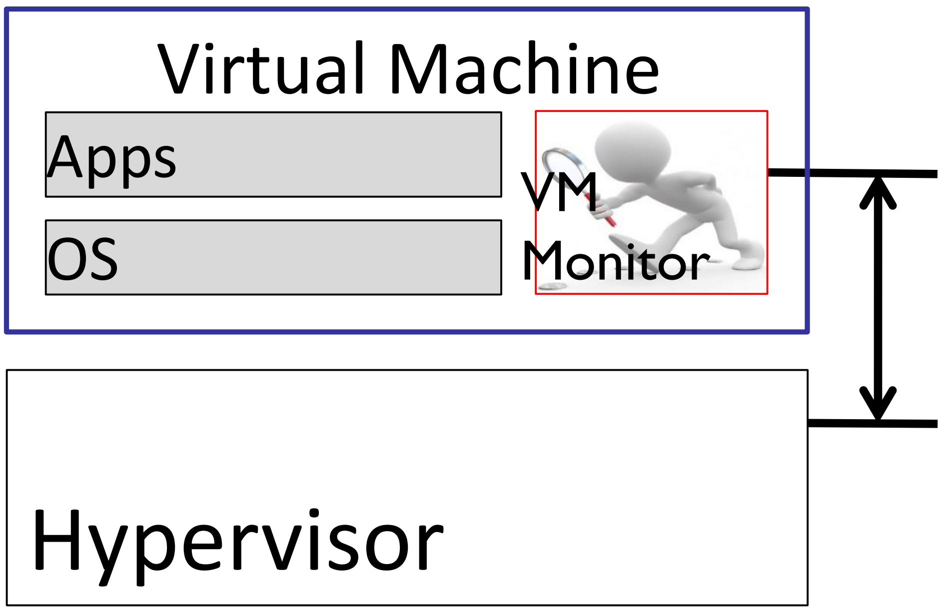
Source: 451 Research's TheInfoPro service reports



Source: Derivative analysis based on Worldwide Virtual Machine 2013–2017 Forecast: Virtualization Buildout Continues Strong IDC #242762 / Aug 2013



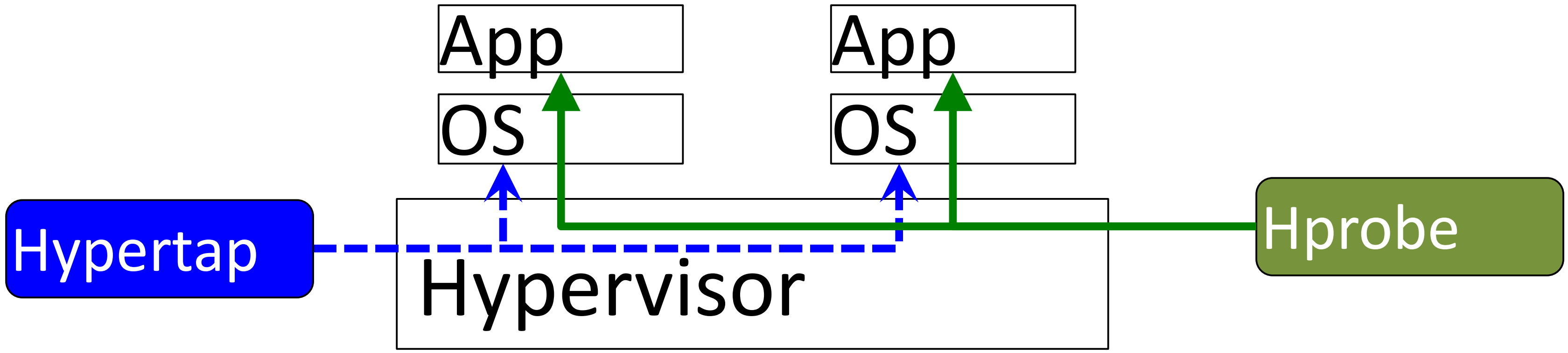
VM Monitoring Overview



+ STRONG ISOLATION

- SEMATIC GAP

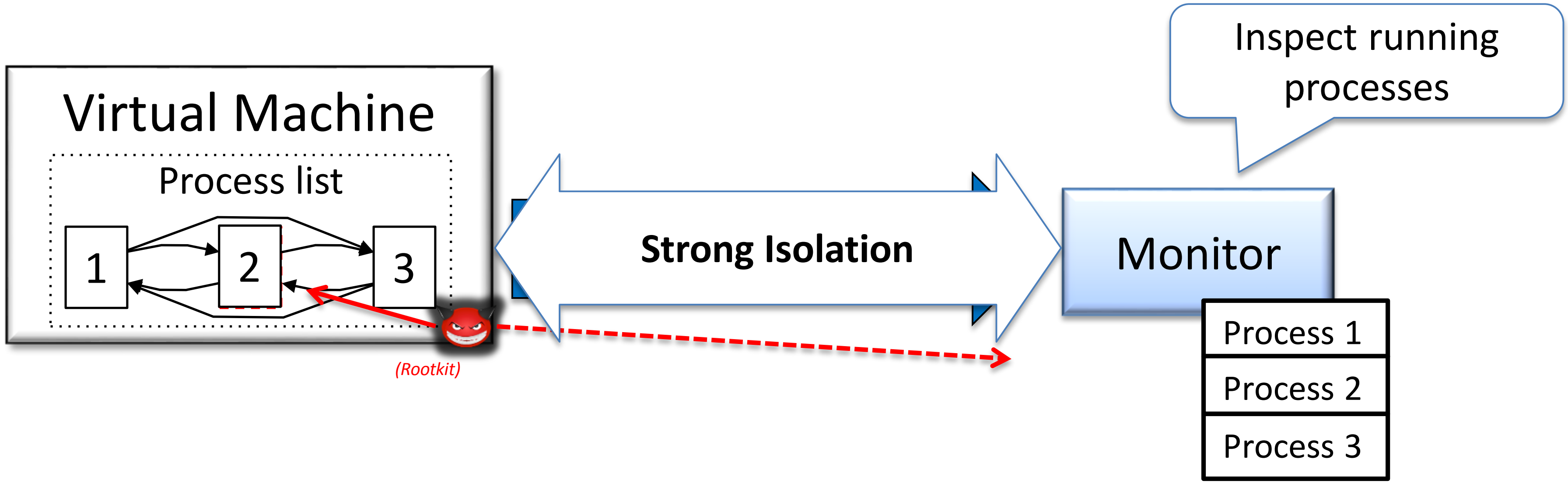
Continuous VM Monitoring



- ✓ Root of trust: HW invariants
- ✓ Tamper-proofed
- ✓ Low runtime overhead

- ✓ Dynamic
- ✓ Supports both VM applications and OS
- ✓ Simple interface
- ✓ Flexible usage

Traditional VM Monitoring

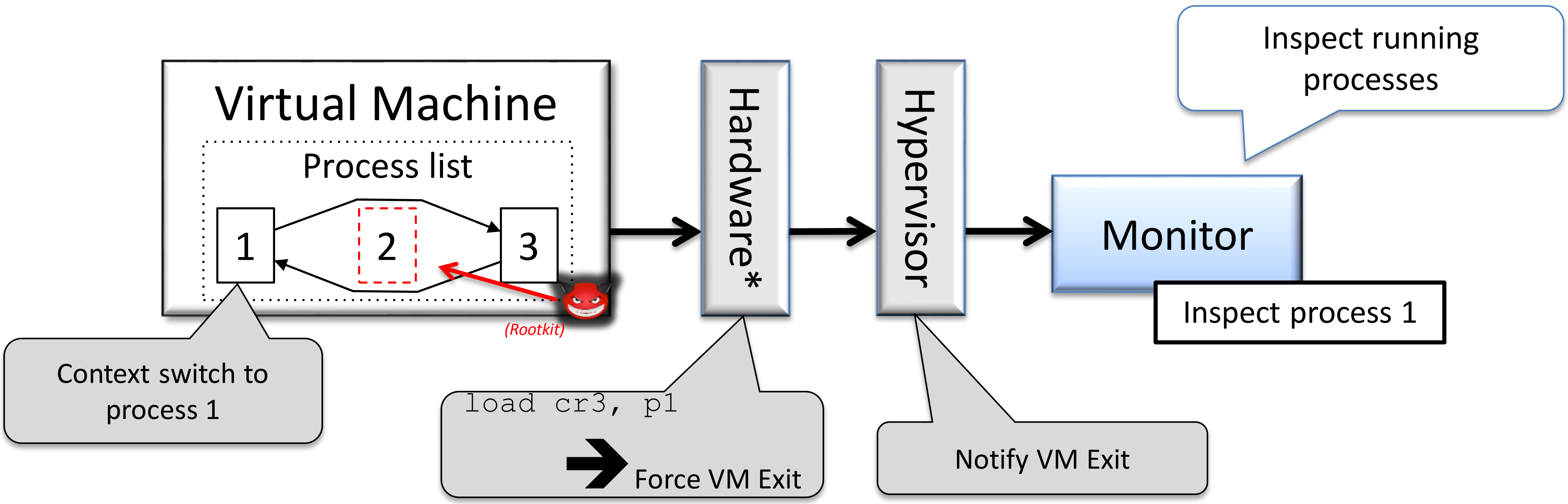


Out-of-VM monitor is manipulated by in-VM attacker!

☹ Places trust on guest *Operating System Invariants*

☹ Polling monitoring - cannot capture VM's operations

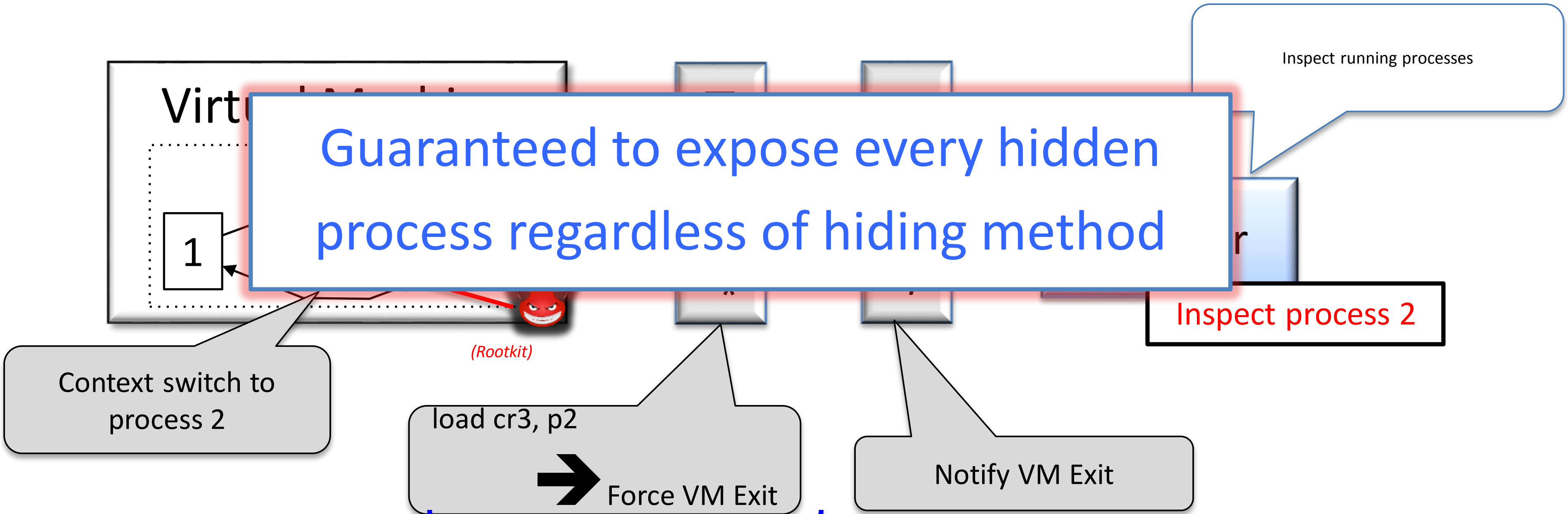
Hardware Invariant Approach



- ✓ Places trust on *Hardware Architectural Invariants*
- ✓ Event-driven monitoring

* x86 with Hardware Assisted Virtualization (HAV) enabled. CR3 holds the Page Directory Base Pointer (PDBP) of running process.

Hardware Invariant Approach



- ✓ Places trust on *Hardware Architectural Invariants*
- ✓ Event-driven monitoring

* x86 with Hardware Assisted Virtualization (HAV) enabled. CR3 holds the Page Directory Base Pointer (PDBP) of running process.

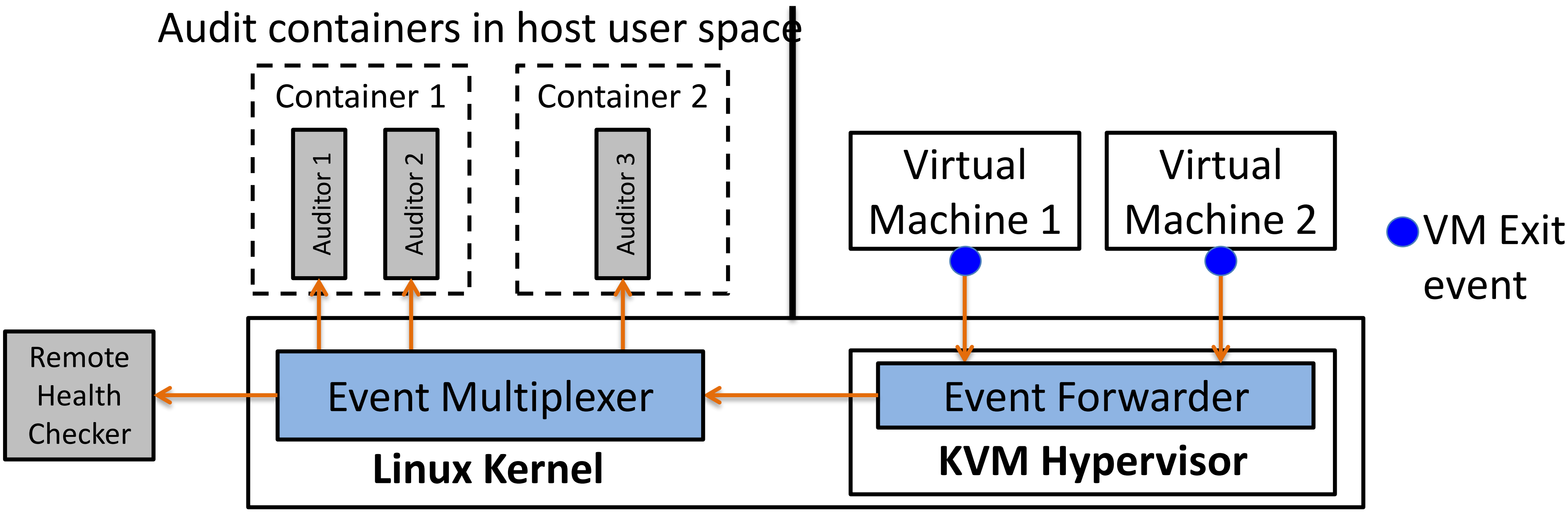
VM Monitoring via HW Invariants

Event	Hardware* Invariants (x86)
Context switch	MMU, CR3 access
Thread/task switch	Page protection, TSS
System call	MSR, Exception
IO access	IO instructions, Interrupts
Memory access	Page protection, Exception

Basis to support a wide range of failure & attack detections

* x86 with Hardware Assisted Virtualization (HAV) enabled.

HyperTap Framework



- Prototyped in KVM
  - Small modification to KVM
- Auditors
  - Implement monitoring policies
  - Run as user processes on host user space
  - Grouped in a container (LXC) per VM

Evaluation of HRKD (Hidden Rootkit Detection)

- Evaluated against real world rootkits on Windows and Linux
- All rootkits successfully detected

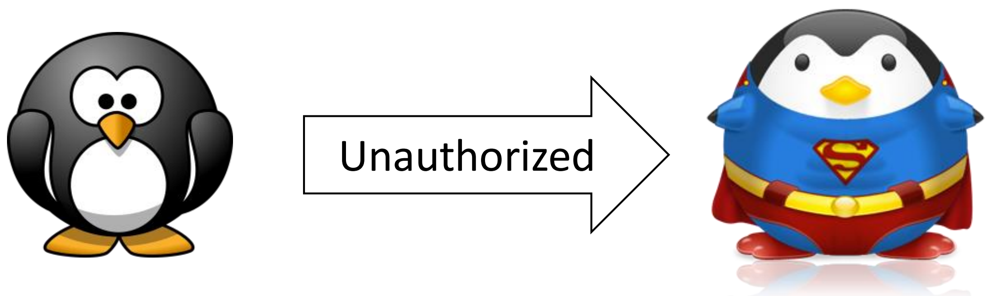
Rootkit	*Target OS*	*Hiding techniques*
FU	Windows XP, Vista	DKOM
HideProc	Windows XP, Vista	..
AFX	Windows XP	Hijack system calls
HideToolz	Windows Vista, 7	Hijack system calls
HE4Hook	Windows XP	Hijack system calls
BH	Windows XP	Hijack system calls
Enyelkm 1.2	Linux kernel 2.6	…
SucKIT	Linux kernel 2.6	Kmem, dkom
PhalanX	Linux kernel 2.6	DKOM

- Detection capability not affected by implementation or hiding techniques of the rootkits
- HRKD can detect future hidden rootkits regardless of their newly invented hiding mechanism

Evaluation of

Privilege Escalation Detection (PED)

- Privilege Escalation Attack



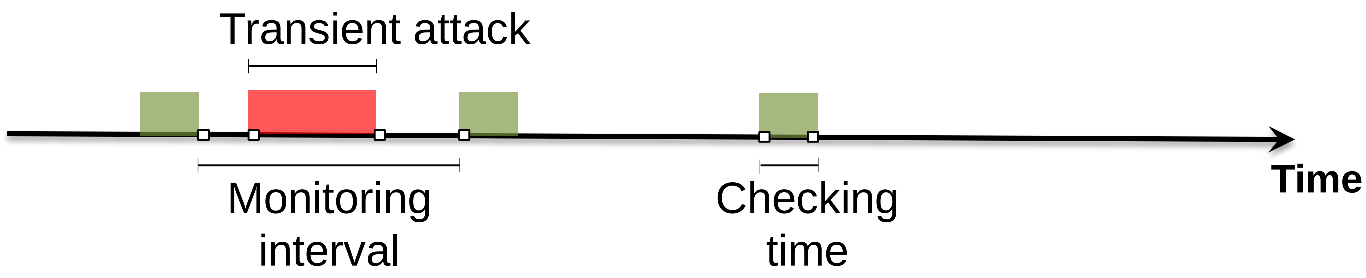
- Detection

Privilege Escalation Detection (PED)

Ninja	Location	Description	Monitoring
O-Ninja	In-VM	Original Ninja	***Polling***
H-Ninja	Out-of-VM	Uses OS invariants	***Polling***
HT-Ninja	**Out-of-VM**	**Uses HW invariants (HyperTap)**	***Event-driven***

HT-Ninja checks a process at *context switches* and *IO system calls*

Three Ninjas against transient attacks



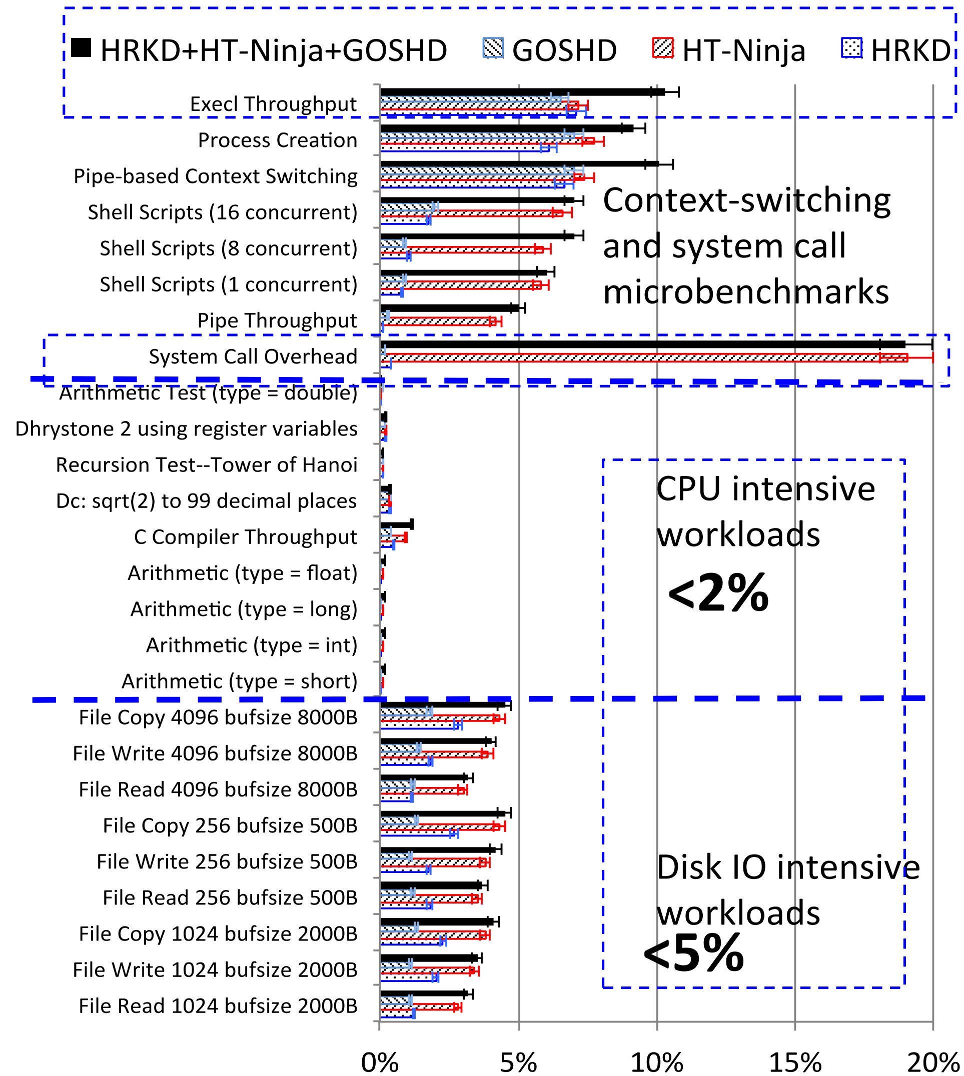
- Transient attacks against polling monitoring

☹ O-Ninja and H-Ninja are highly vulnerable to transient attacks

☺ HT-Ninja uses event-driven monitoring and is not vulnerable to transient attacks

Performance Overhead

- Combined overhead < sum of individual overheads
- **<2%** overhead for CPU workloads
- **<5%** overhead for IO workloads
- Micro-benchmark:
  – Highest performance loss for NOOP system call (~19%)

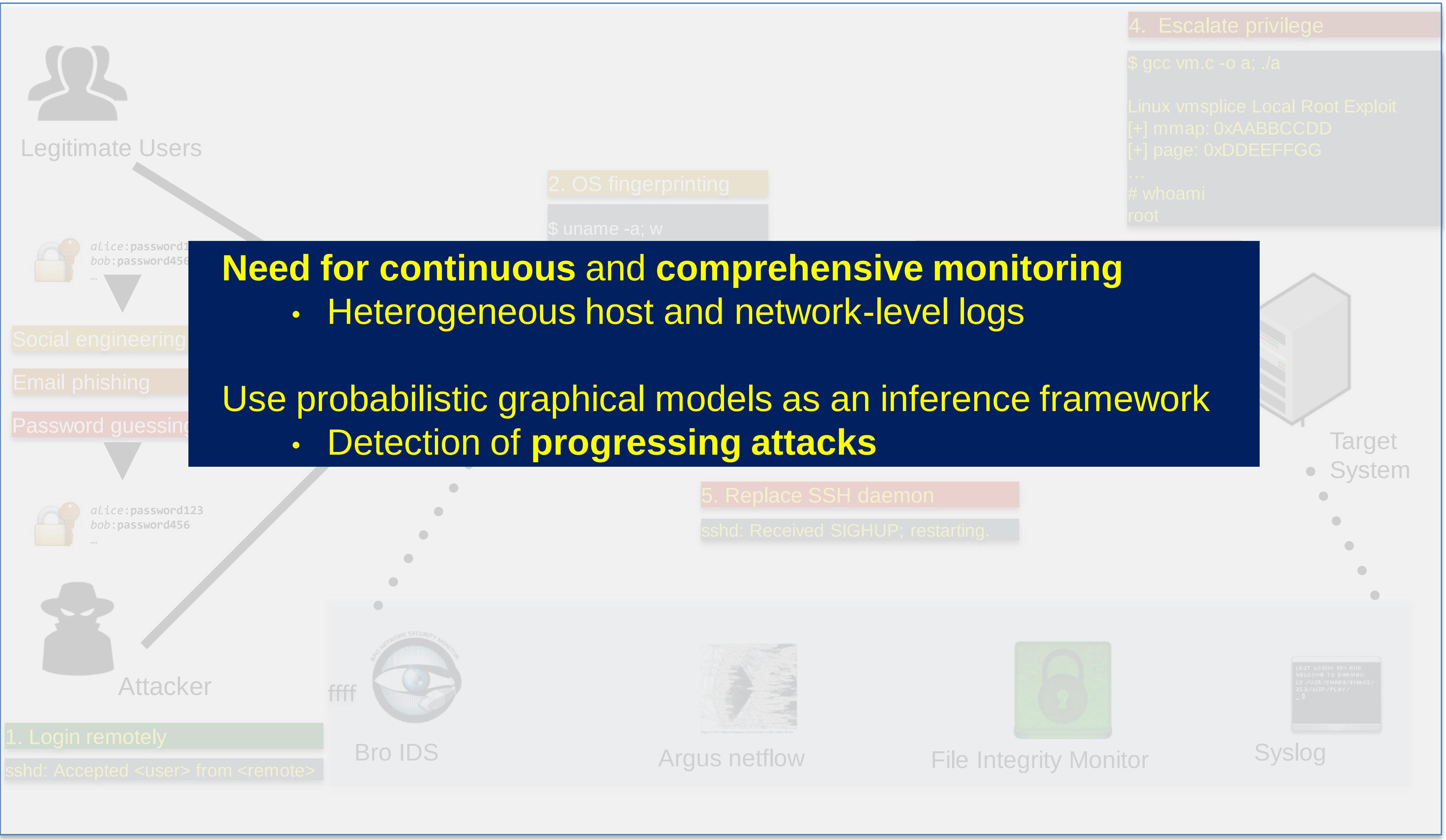


VM Monitoring Overview

	HyperTap (DSN'14)	HProbes (TBD)	LiveWire (NDSS'03)	VMWatcher (CCS'07)	LibVMI (ACSAC'07)	SIM (CCS'09) / Lares (SP'08)	Lycosid (VEE'08) / Antfarm (ATC'06)	Nitro (AICS'11) / Ether (CCS'08)	Osck (ASPLOS'11)	Virtuoso (SP'11)	VMST (SP'12)	TxIntro (HPCA'14)
Root-of-trust (invariant)	HW	OS	OS	OS	OS	OS	HW	HW	OS	OS	OS	OS
Continuous/**P**olling Mon.	C	C	P	P	P	C	P	P	P	P	P	P
Changes to VM	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
Custom Auditors	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓
Online Detection	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Auto-generate Invariants	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗
Userspace Monitoring	✗	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗

PROBABILISTIC INFERENCE ON SECURITY LOGS

Example Attack Scenario



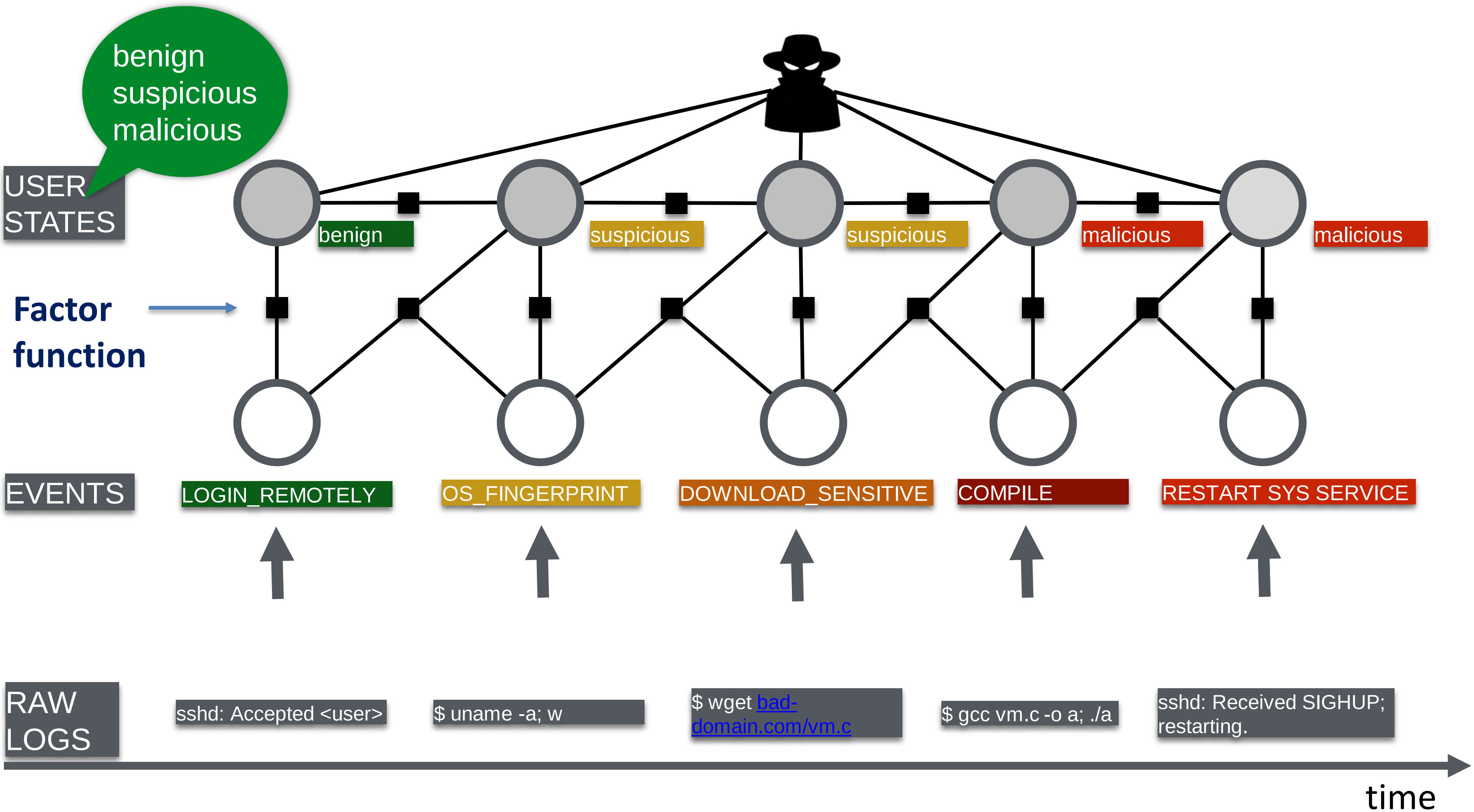
Need for continuous and **comprehensive monitoring**

- Heterogeneous host and network-level logs

Use probabilistic graphical models as an inference framework

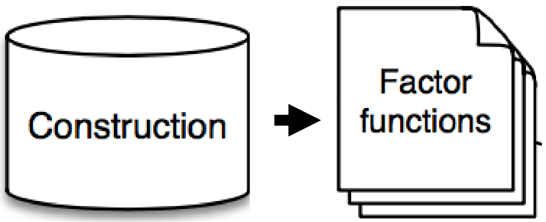
- Detection of **progressing attacks**

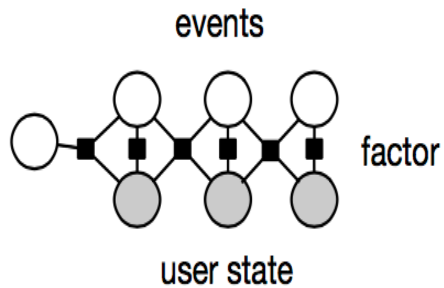
Integrating Heterogeneous Monitoring Data Using Probabilistic Graphical Models

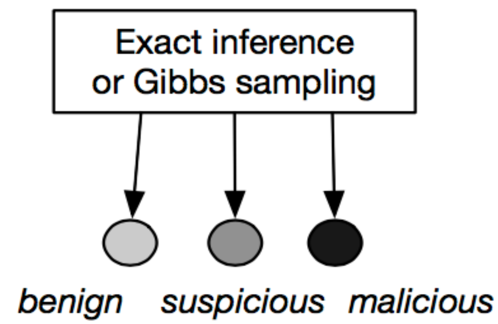


AttackTagger Workflow

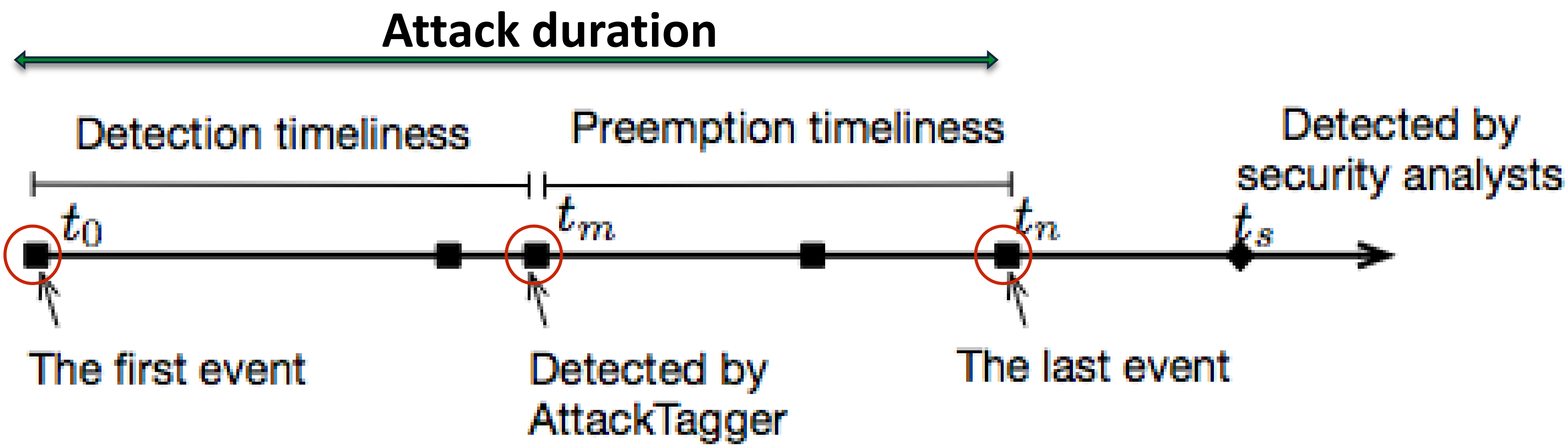
Construct factor functions based on past incidents → Extract events corresponding to an incident → Construct per-user factor graph → Infer the user states



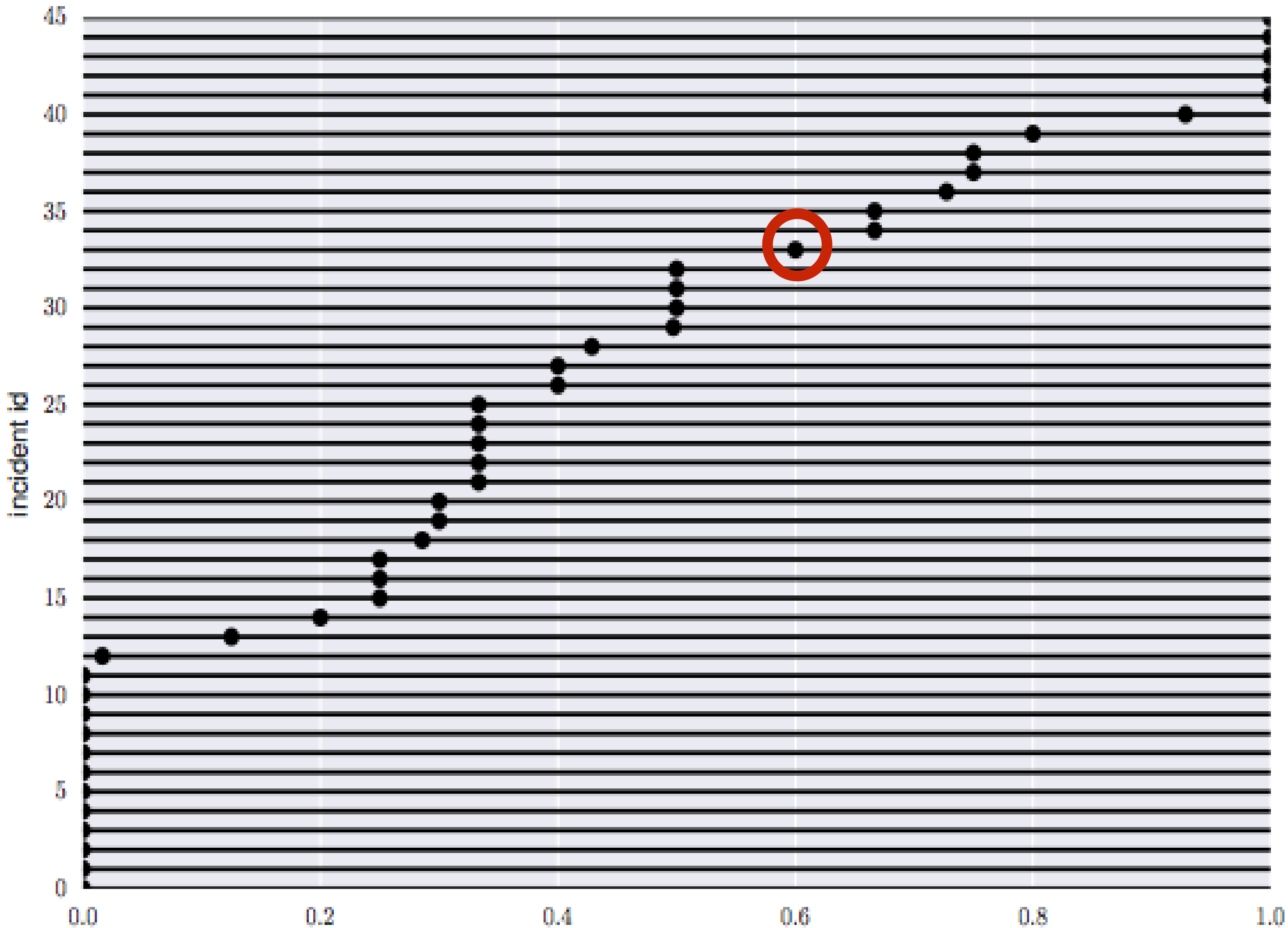




Metrics: Detection timeliness & Preemption timeliness



Detection timeliness & Preemption Timeliness



first event

last event

Percentage of events observed until attack detection

46 of 62 malicious users were detected in tested incidents (74%)

41 of 46 identified malicious users were identified before the system misuse

Conclusions

- ***Design for resiliency*** needs multi-disciplinary experts in security, fault tolerance, human factors
- ***Achieving resiliency needs***
  - Application driven continuous monitoring and response to intrusions
  - Combination of knowledge on cyber and physical aspects of the system to devise protection while preserving system performance
  - Scientifically sound inference methods (and tools) to determine system/application state based on runtime observations on the system behavior