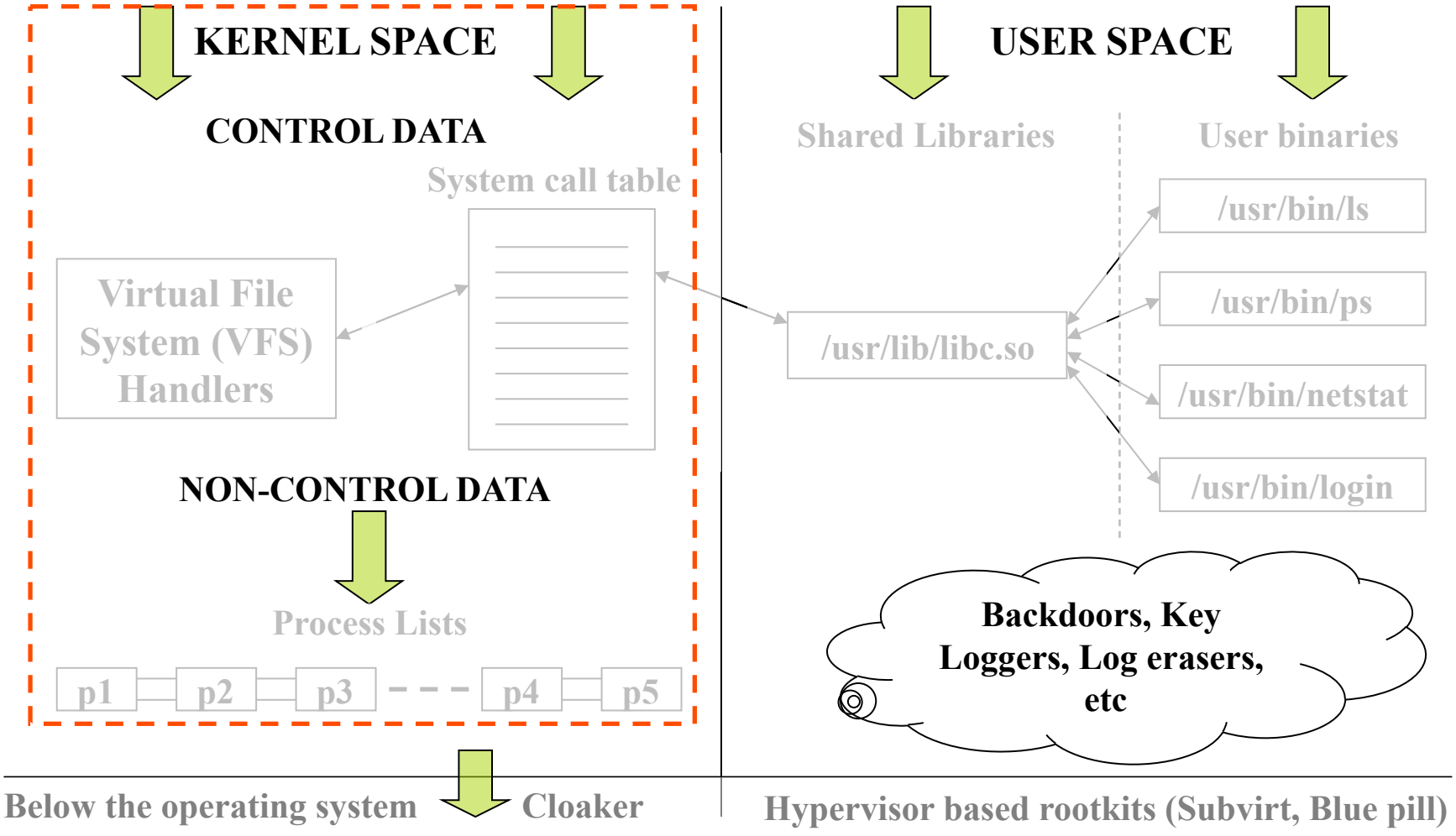
Automatic Inference and Enforcement of Kernel Data Structure Invariants

Arati Baliga, Vinod Ganapathy and Liviu Iftode
Department of Computer Science
Rutgers University

Rootkits, the growing threat !

- Computer systems today face a realistic and growing threat from rootkits.
  - 600% increase from 2004-2006 (McAfee Avert Labs)
  - Over 200 rootkits in first quarter of 2008 (antirootkit.com)
- Collection of tools used by the attacker to conceal his presence on the compromised system.
- **Rootkits allow the attacker to**…
  - Maintain long term control
  - Reuse the system's resources
  - Spy on the system
  - Involve system in malicious activities

Rootkit hiding trends



Current Approaches

- **Automated technique, limited in scope**
  - SBCFI [Petroni et al., CCS 2007]
- **Manual specification based techniques**
  - Copilot [Petroni et al., Usenix Security 2004]
  - Specification based architecture [Petroni et al., Usenix Security 2006]

- **Challenge**

	Location of data		Type of data		Specifications
	Static	Dynamic	Control	Non-control	Automatic
Copilot	✓	x	✓	✓	x
Specification based detection	✓	✓	✓	✓	x
SBCFI	✓	✓	✓	x	✓
Our approach	✓	✓	✓	✓	✓

Outline

- Introduction
- **Approach**
- Attack examples
- Design and implementation
- Experimental evaluation
- Conclusions

Our approach

- A comprehensive technique to detect rootkits based on **automatic invariant inference.**
- **Invariant is a property** that holds over an individual object (e.g. variable or struct) or a collection of objects (e.g. arrays or linked lists).
- Learns invariants over a training phase and enforces them during normal operation.
- Works uniformly across control as well as non-control data.

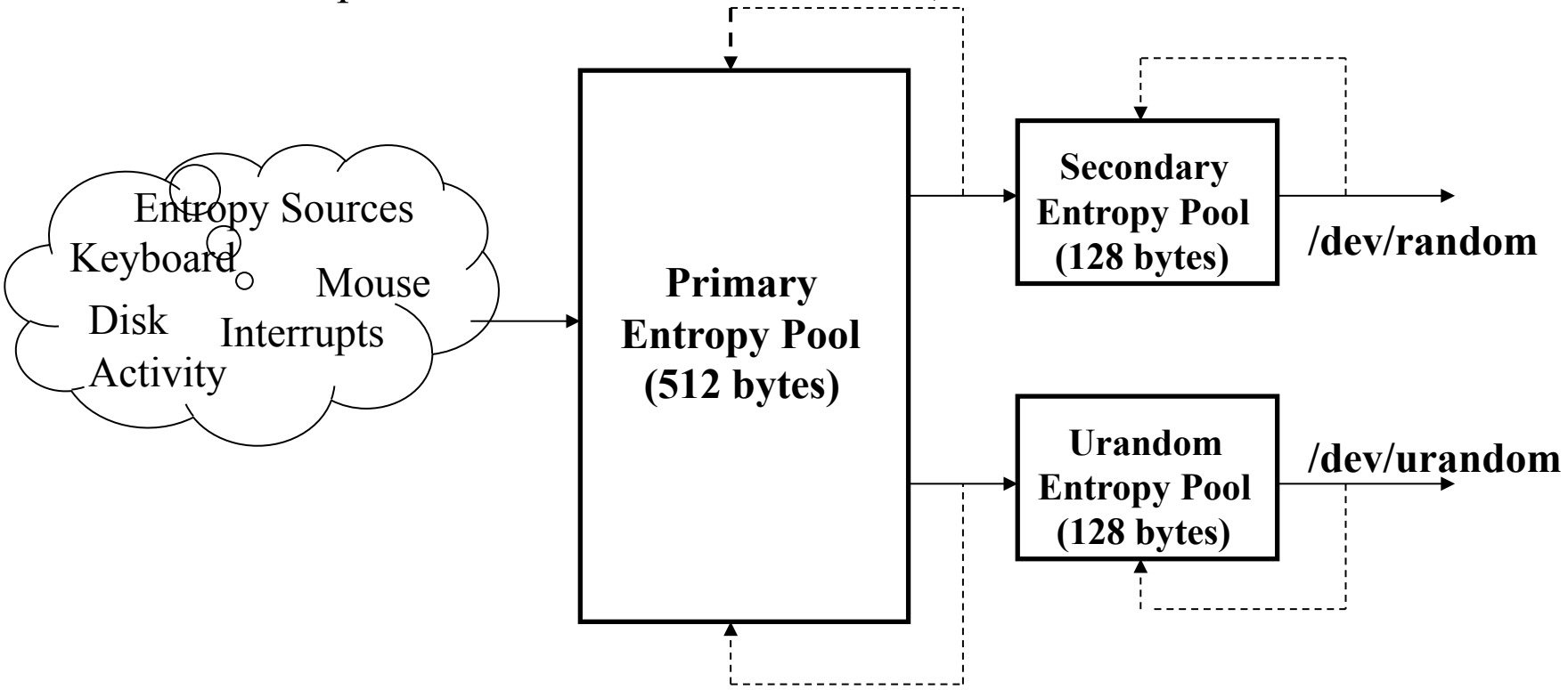
Attacks that violate invariants

- We demonstrate four examples in this talk
  - Two proposed by us [Baliga et al., Oakland 2007]
    - Entropy pool contamination
    - Resource Wastage
  - Two attacks proposed by others
    - Hiding Process (Used by the *fu* rootkit, Butler et al.)
    - Adding binary format (Proposed by Shellcode security research group)

Attack 1 – Entropy pool contamination

Attack Overview:
Attack constantly writes zeroes into all three pools and the polynomials used to stir the pools

Impact:
All applications that rely on the random number generator such as tcp sequence numbers, session ids are affected



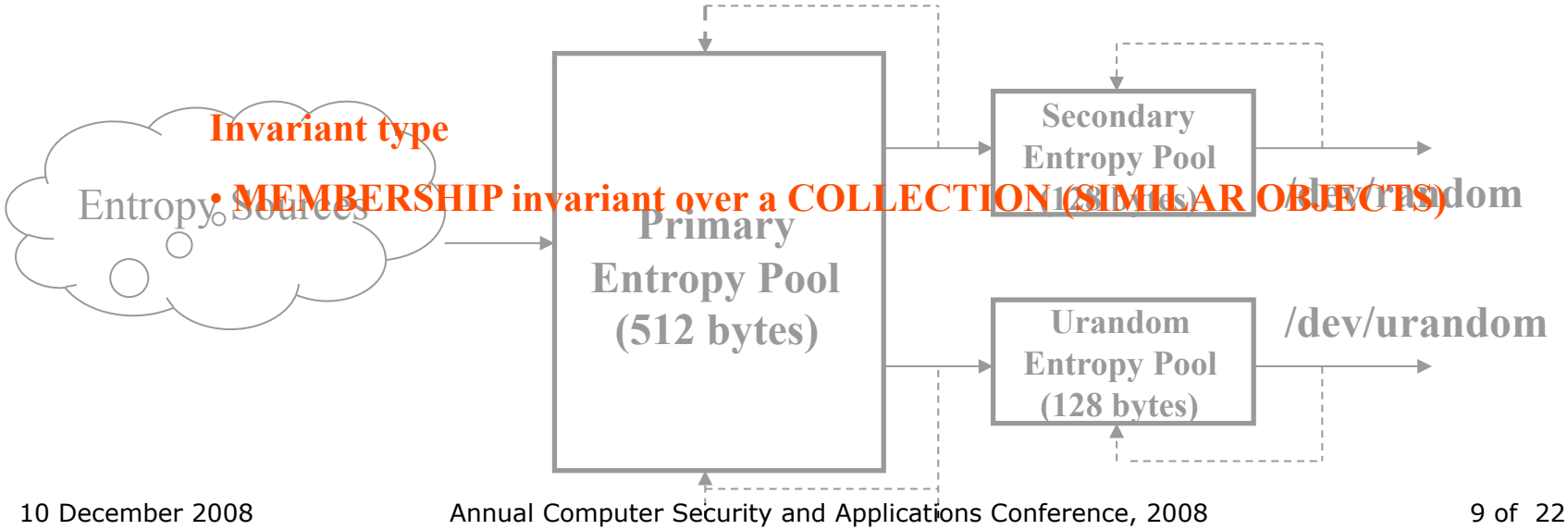
Attack 1 – Invariants violated

Data structures involved.

`struct poolinfo`. This is a member of the entropy pool data structures of type `struct entropy_store`

Invariant violated by attack.

```
poolinfo.tap1 € {26, 103}
poolinfo.tap2 € {20, 76}
poolinfo.tap3 € {14, 51}
poolinfo.tap4 € {7, 25}
poolinfo.tap5 == 1
```

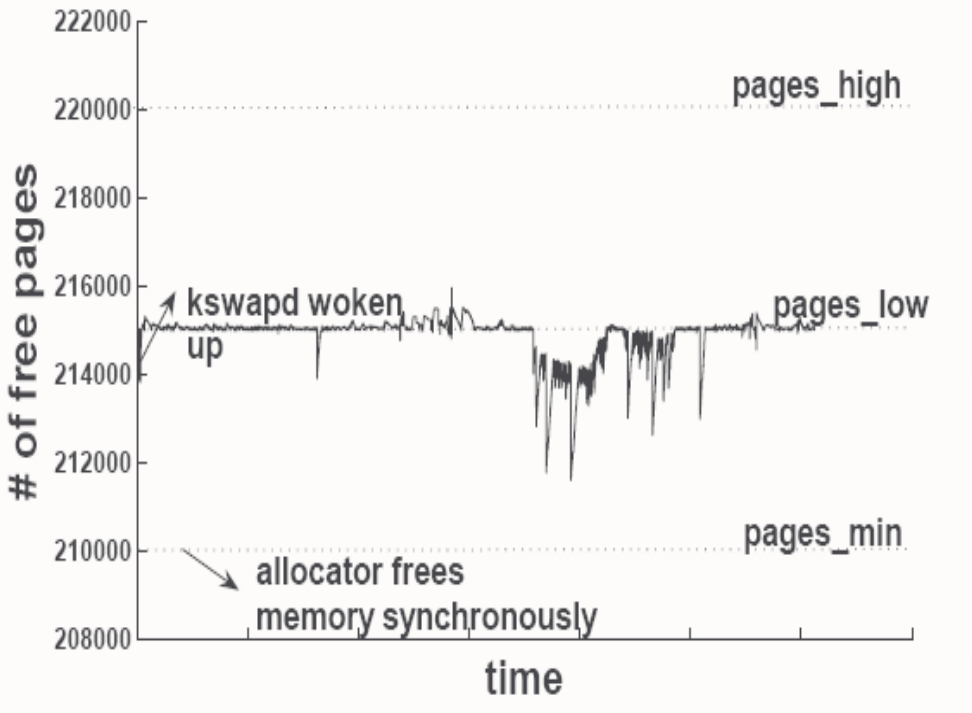


Invariant type

- **MEMBERSHIP invariant over a COLLECTION (SIMILAR OBJECTS)**

Attack 2 – Resource wastage attack

Attack Overview:
Attack manipulates the zone watermarks to create an impression that most of the memory is full



Watermark	Original Value	Modified Value
pages_min	255	210000
pages_low	510	215000
pages_high	765	220000
total free pages	144681	210065
Total number of pages in zone: 225280		

Impact:
Resource wastage and performance degradation

Application	Before Attack	After Attack	Degradation (%)
file copy	49s	1m, 3s	28.57
compilation	2m, 33s	2m, 56s	15.03
file compression	8s	23s	187.5

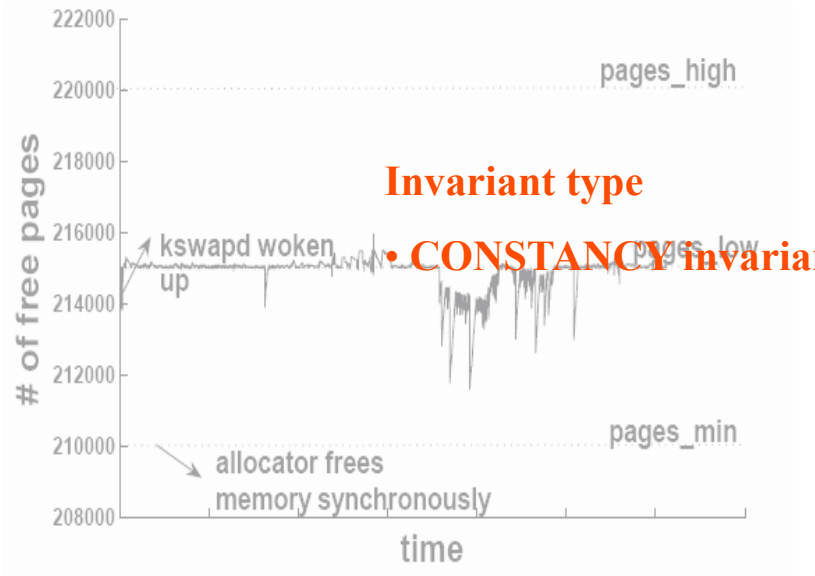
Attack 2 – Invariants violated

Data structures involved.

`zone_table[]` array. Each element of type `struct zone_struct`

Invariant violated by attack.

```
zone_table[1].pages_min == 255
zone_table[1].pages_low == 510
zone_table[1].pages_high == 765
```

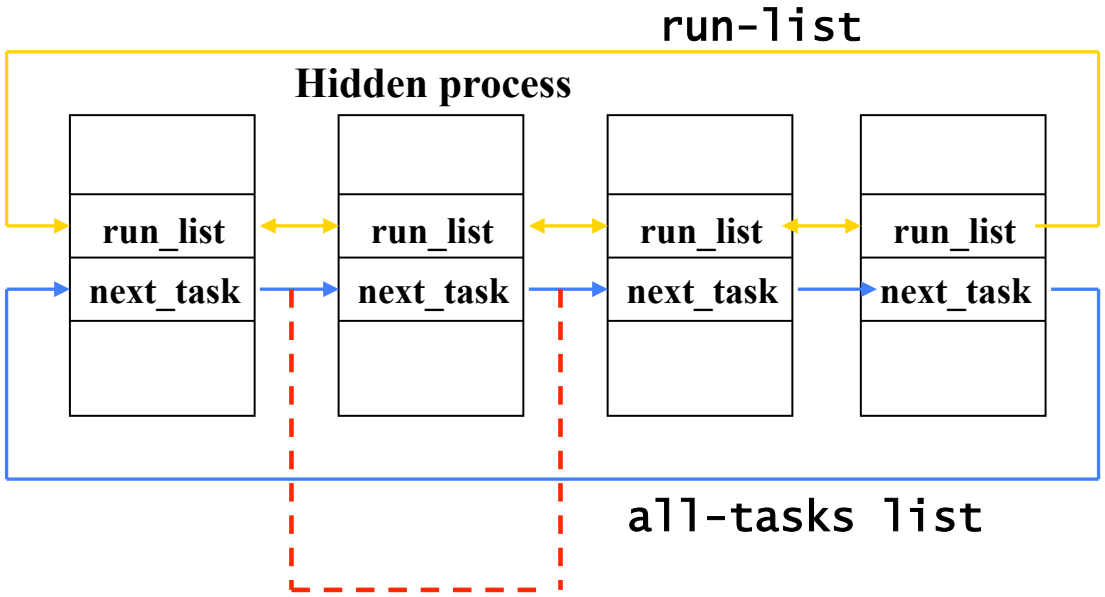


Invariant type

- **CONSTANCY invariants over individual OBJECTS**

Watermark	Original Value	Modified Value
pages_min	255	210000
pages_low	510	215000
pages_high	765	220000
total free pages	144681	210065
Total number of pages in zone: 225280		

Attack 3 - Hidden process attack



Invariant type

- **SUBSET property over a COLLECTION (LINKED LIST)**

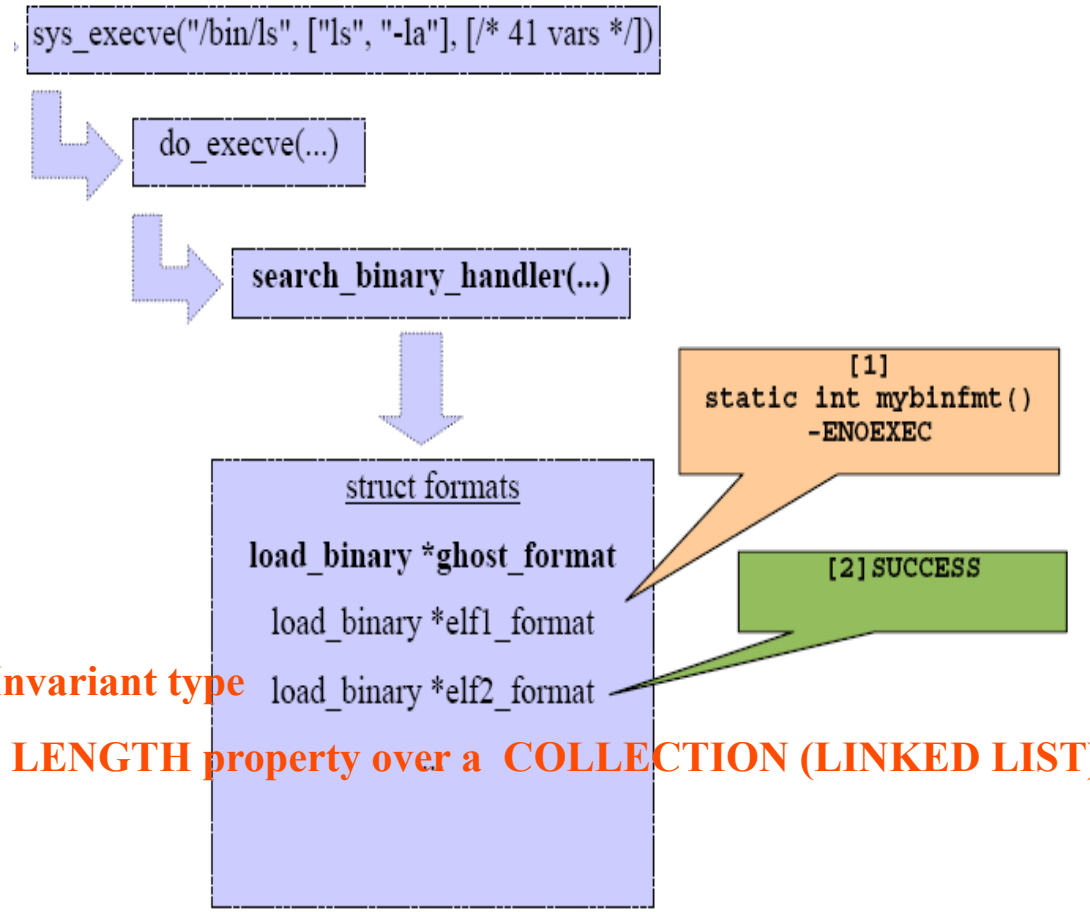
Attack Overview:
Attack removes malicious process entry from `all-tasks` list but retains in `run-list`

Impact:
Malicious process is hidden from accounting tools

Data structures involved.
`Process run-list`
`Process all-tasks list`

Invariant:
`run-list` $\subseteq$ `all-tasks`

Attack 4 – Adding binary format attack



Invariant type

- **LENGTH property over a COLLECTION (LINKED LIST)**

Figure used from Shellcode security research document published at

http://goodfellas.shellcode.com.ar/own/binfmt-en.pdf

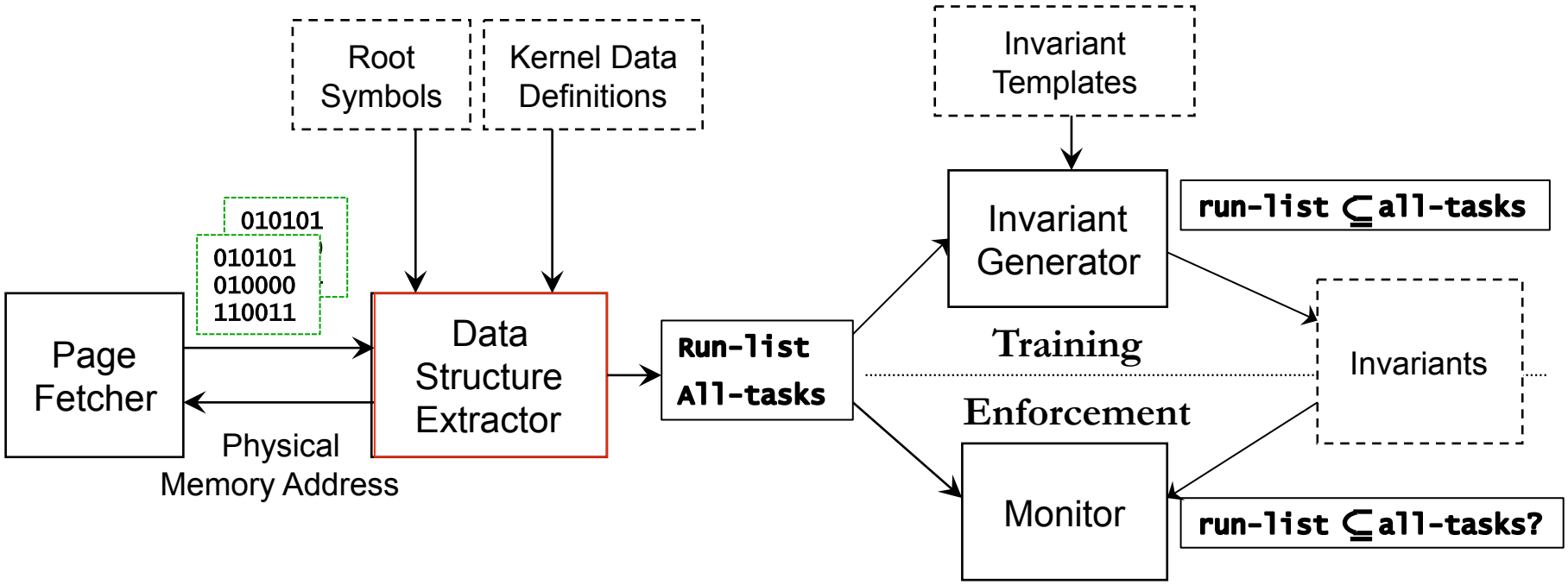
Attack Overview:
Attack adds a new binary format containing a malicious handler.

Impact:
Malicious code invoked each time a new process is created on the system

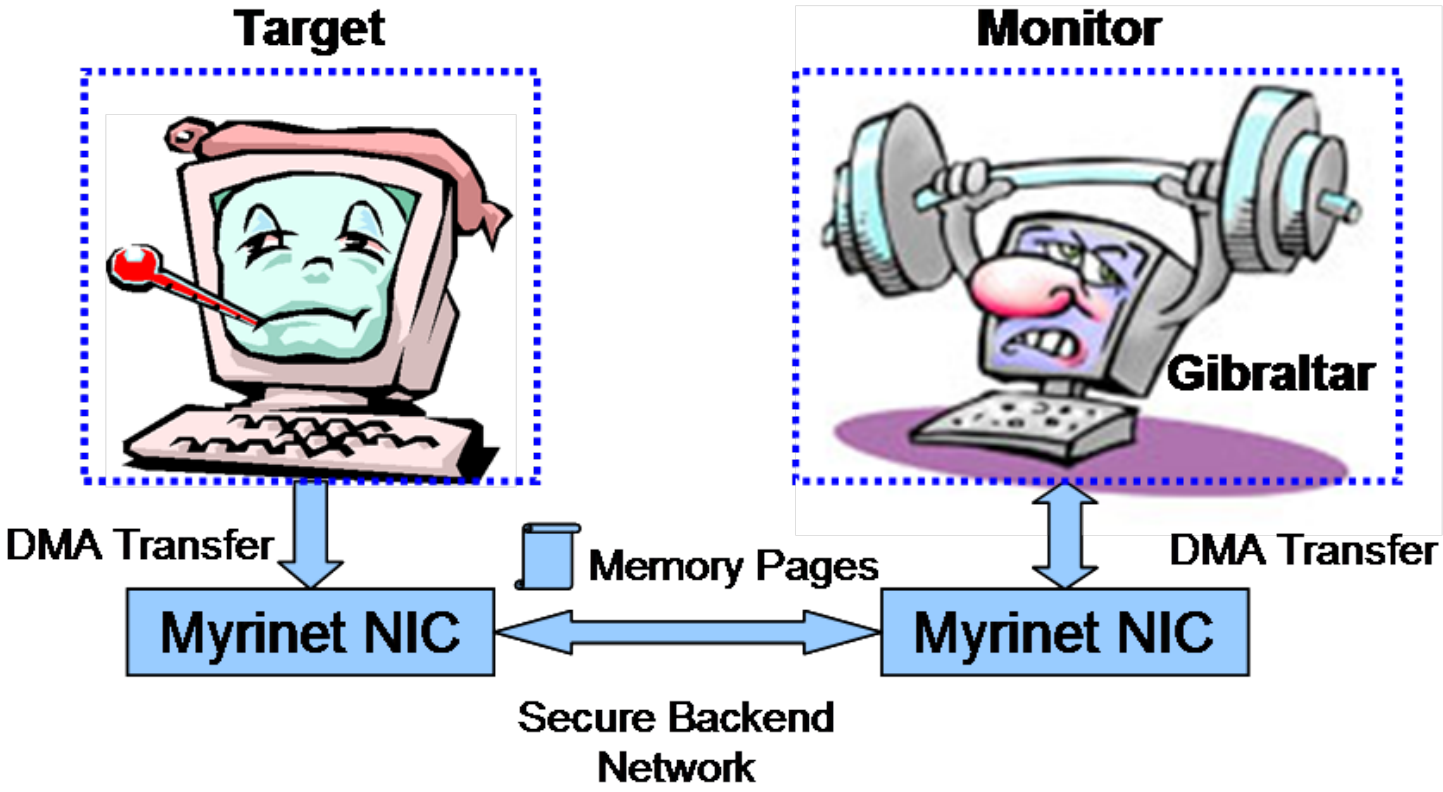
Data structures involved.
`formats` list

Invariant:
`len(formats) == 2`

Gibraltar architecture



Prototype (Gibraltar)



- Fetches remote memory pages from the target continuously

Invariants automatically inferred

Template	Object	Collection
Membership	643,622	422
Non-zero	49,058	266
Bounds	16,696	600
Length	NA	4,696
Subset	NA	3,580
Total	709,376	9,564

Total 718,940 invariants inferred by Gibraltar. These invariants are used as data structure integrity specifications during enforcement.

Detection Accuracy

- Test suite
  - Fourteen publicly available kernel rootkits
  - Six advanced stealth attacks on the kernel (previously discussed)
- Results
  - All of them detected (No false negatives)
- False positive evaluation
  - Benign workload run for half an hour consisting of combination of tasks
  - 0.65% false positive rate

Template	Object	Collection
Membership	0.71%	1.18%
Non-zero	0.17%	2.25%
Bounds	0%	0%
Length	NA	0.66%
Subset	NA	0%
Average false positive rate: 0.65%		

Copying the Linux kernel source code from one folder to another.

Editing a text document

Compiling the Linux kernel

Downloading eight video files from the Internet.

Perform file system operations using the IOZone benchmark

Performance Evaluation

- Training Time
  - 25 mins for snapshot collection, 31 minutes for invariant inference (Total of 56 minutes).
- Detection Time
  - Ranges from 15 seconds up to 132 seconds. Large variance depending on the number of objects found in memory.
  - Number of objects varies depending on the workload running on the system and system uptime.
- PCI Overhead
  - DMA access creates contention for the memory bus.
  - 0.49% (Results of the stream benchmark)

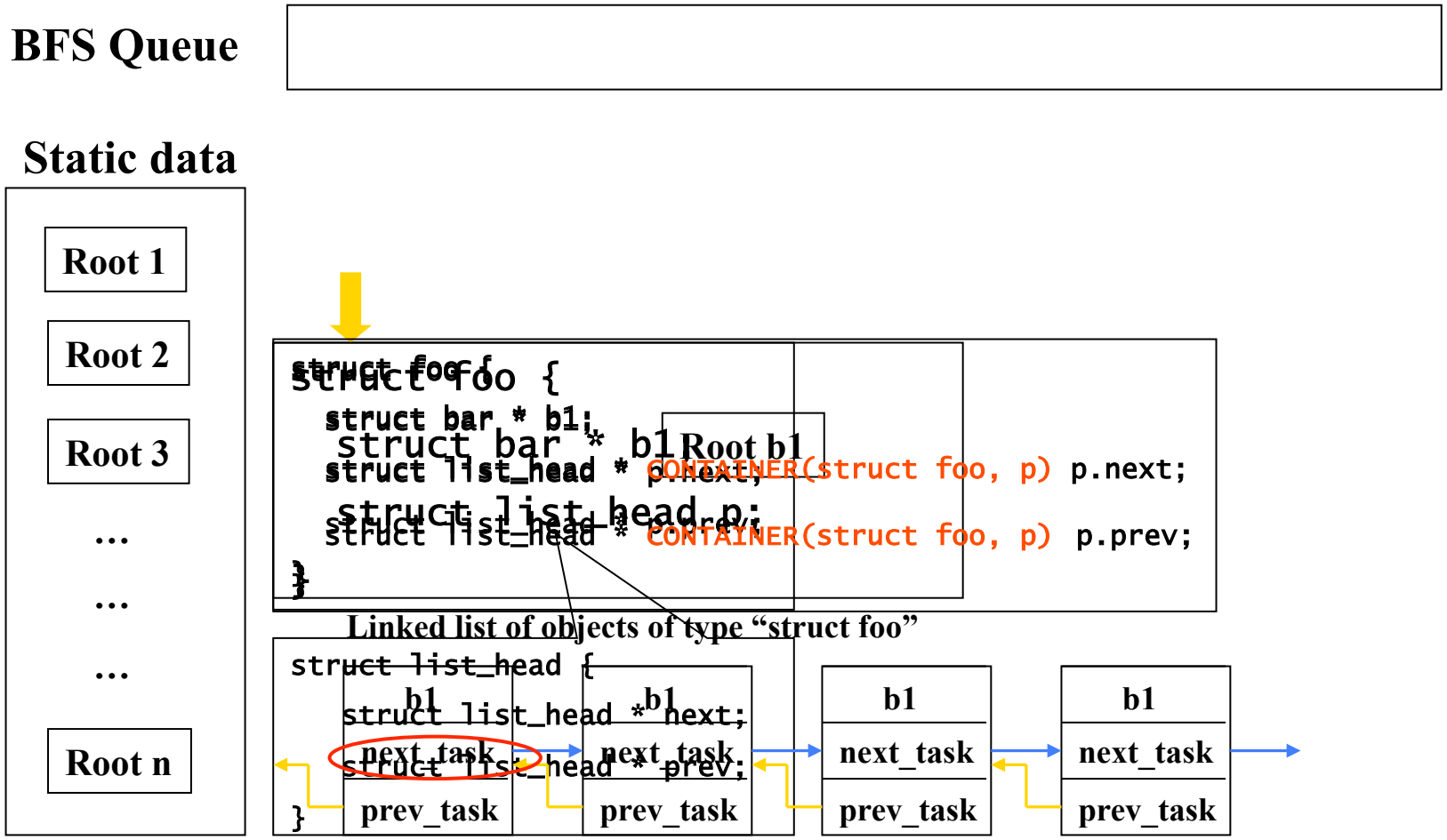
Conclusions and future work

- Our approach automatically infers invariants over kernel control and non-control data.
- Gibraltar could automatically detect publicly available rootkits and advanced stealth attacks using automatically inferred invariants.
- As future work, we plan to investigate
  - Improvement of false positive rate (filtering, feedback)
  - Quality of invariants generated
  - Portability of invariants across reboots.

Questions ?

Thank you !

Data structure extractor



Invariant generator

- We leverage Daikon's invariant inference engine to extract invariants over kernel snapshots.
- Daikon is a tool for dynamic invariant inference over application programs.
- We focus on the following five templates
  - Membership template (var € {a, b, c}).
  - Non-zero template (var != 0).
  - Bounds template (var < const), (var > const).
  - Length template (length(var) == const).
  - Subset template (list1⊆ list2).