Automated Detection and Containment of Rootkit Attacks

Arati Baliga and Liviu Iftode
Computer Science Department
Rutgers University
Piscataway, NJ

Xiaoxin(Mike) Chen
VMware Inc.
Palo Alto, CA

Outline

- Motivation
- Background
  - Rootkits
  - Virtualization
- Our Approach
- Prototype
- Evaluation
- Conclusion

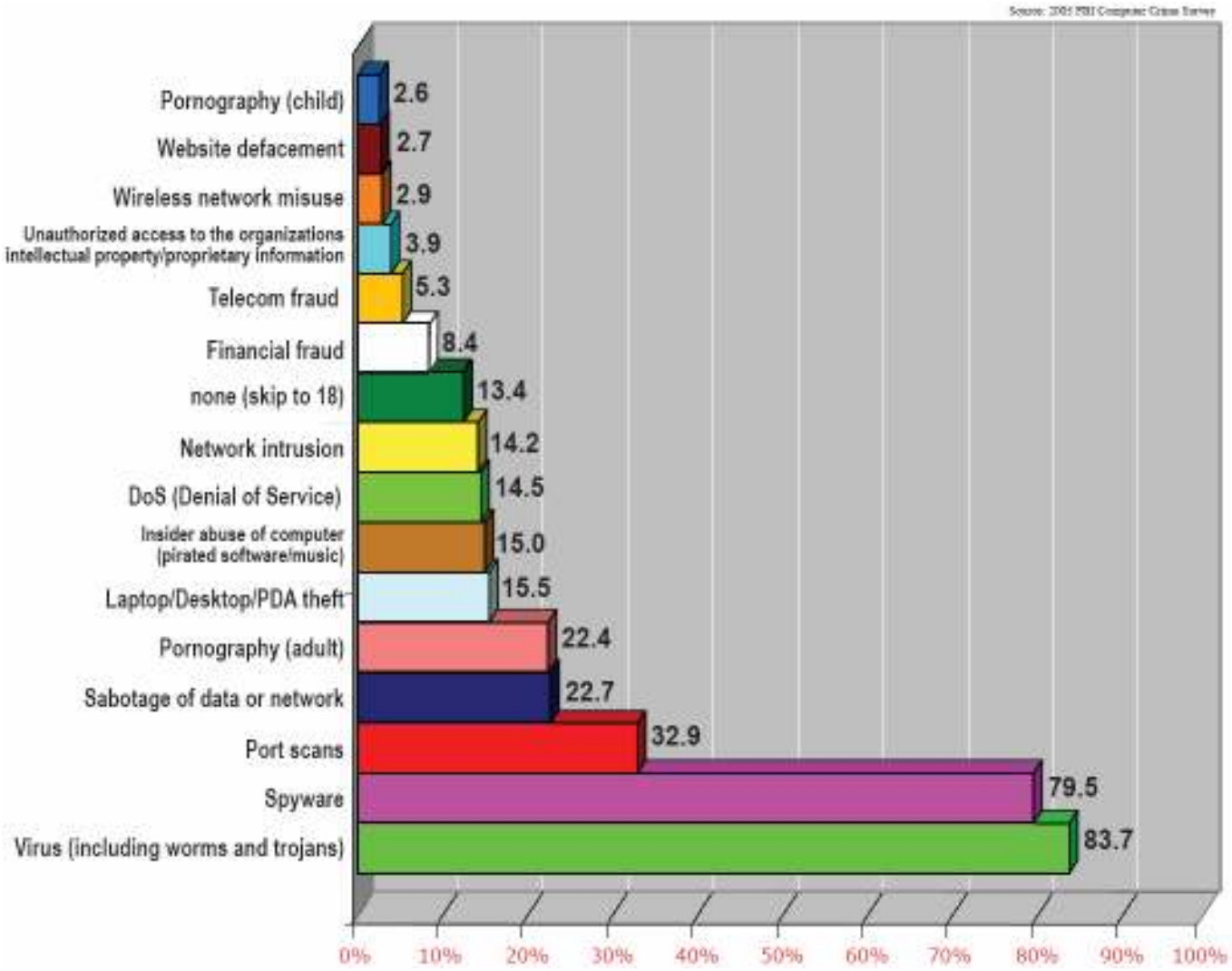
Motivation

- Large percentage of internet traffic is becoming increasingly adversarial.
- Computer systems equipped with inadequate defenses are highly susceptible to attacks from the network.
  - Most computers today have always-on connection to the internet
- Computer crimes are on the rise.
  - Data theft/misuse (ex: identity theft)
  - Disruption of service (ex: DDoS attacks, website defacement)
  - Controlling other PCs on the internet

Shifting Motives

- Attackers have matured from **criminal mischief** to **criminal profiteering**
  - Criminal Mischief
    - Disruption of service (website defacement)
    - Naïve viruses and worms
  - Criminal Profiteering
    - Data theft (stealing passwords, bank information, credit card numbers, identity)
    - Controlling PCs (pay-per-click, Google Adsense, zombie PCs, botnets)
    - Disruption of service (DDoS attacks - extortion)

FBI Computer Crime Survey (2005)



Shifting attack trends

- Shifting motives have led to shifting attack trends
  - Most common attacks are automated (ex: SQL Slammer, Code Red)
- Attackers want to maintain long-term control of compromised PCs
  - Such attacks require remote access and stealth (ability to stay undetected), which are readily provided by ***rootkits***.
- **Automated attacks need automated solutions**
  - Human response is often inefficient and almost always too slow !

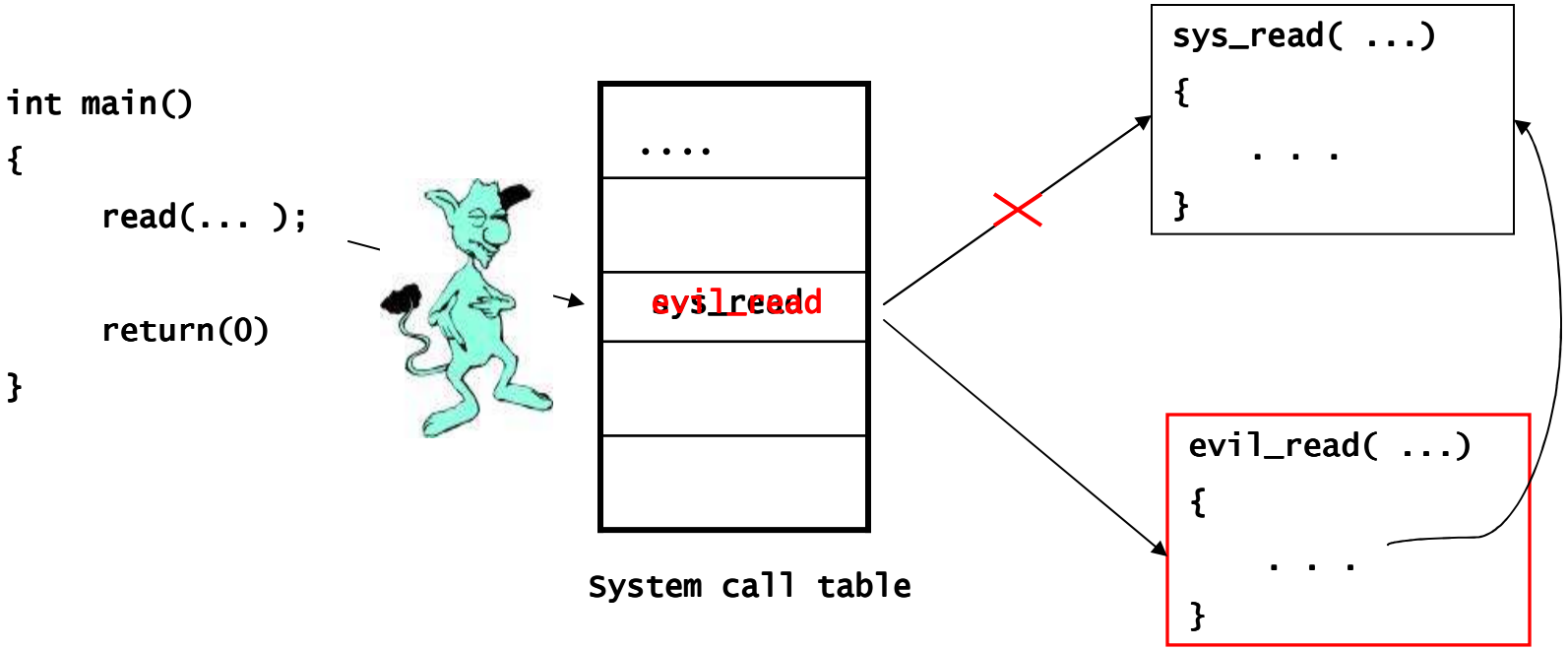
Rootkits

- Collection of tools used by the attacker to hold root privileges on the compromised system.

- Rootkit hiding mechanism:
  - User-level rootkits
    - Replace system binaries like *ps* and *netstat*
    - Replace shared libraries
  - Kernel-level rootkits
    - Replace entries in system call table
    - Replace entries in interrupt descriptor table (IDT)
    - Replace kernel text.

Rootkit Example

- System call hijacking



Research in Countering Rootkits

- Tools used
  - Chkrootkit, Rootkit Revealer (signature based approach)
- Copilot
  - Hardware based independent auditor
- Strider Ghosbuster
  - Comparison of clean view v/s infected view

All approaches limited to detection !

They do not address containment.

Why is containment important ?

- Containment is critical to stop immediate damage
  - Containment can stop a worm from spreading or spyware from leaking your credit card information
- A coarse reaction would be to shutdown the machine
  - Frequency of attacks makes this infeasible
- We need a more fine-grained reaction to rootkits

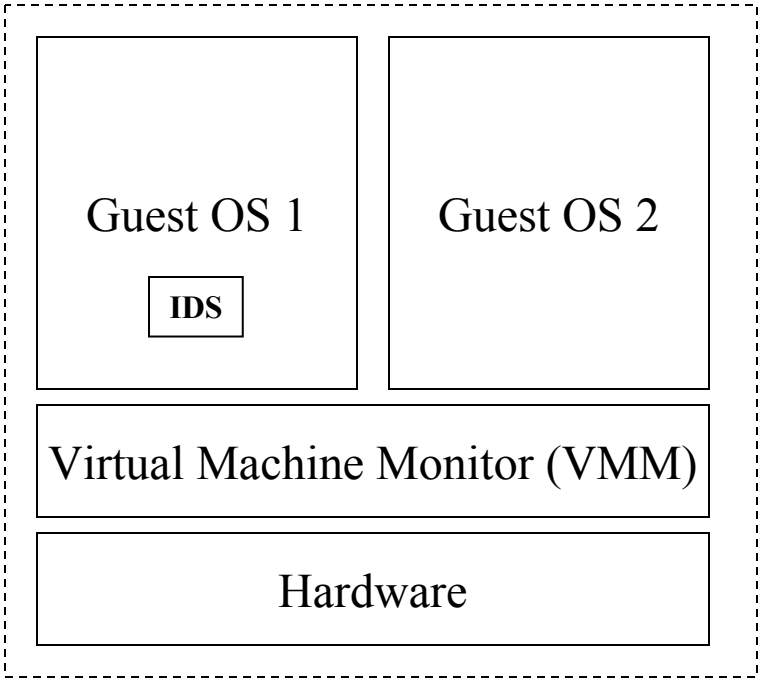
IDS Location

- IDS needs to be isolated from the system that it is monitoring
  - Network Based IDS (NIDS)
    - Poor view of the system
  - Independent hardware on host
- Virtualization provides this isolation and still a better view of the system being monitored.

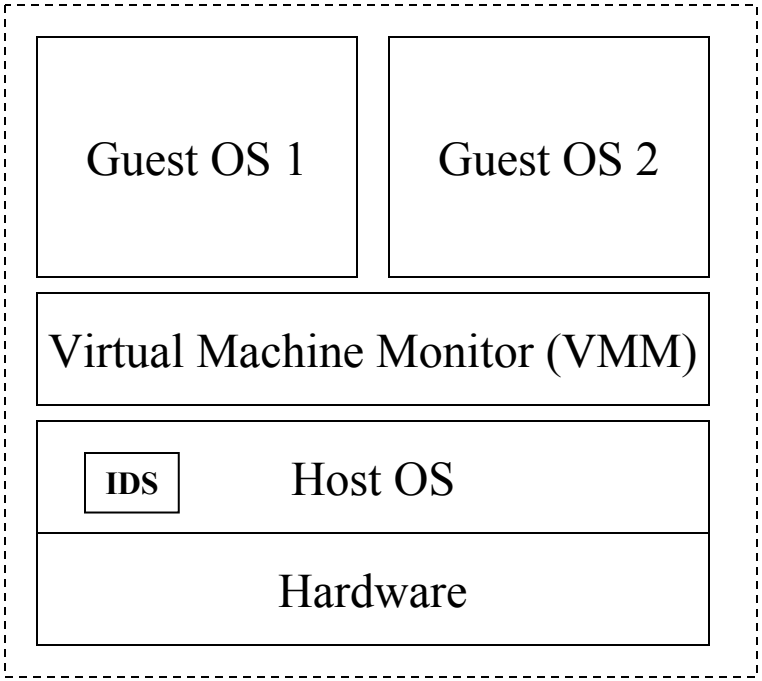
Virtualization Technology

- Have been around since 1960's on mainframes (VM/370)
  - Multiple operating systems on single machine
- Have resurfaced on commodity platforms in a new way
  - Provide better management
  - Resource consolidation
  - Mobility
  - Intrusion detection and secure logging
- Rapidly adopted by data centers and enterprises for hosting services

Virtual Machine Types



Type 1 VM



Type 2 VM

Our Approach

- Leverages VM technology and it's security model
- Define notion of ***protected zones***
  - In memory
  - On file system
- **Prevent** illegal access to protected zones
- **Detect** attempted illegal access to protected zones.
- **Track** dependencies between processes and between files and processes
- **Contain** damage in progress using dependency information

Protected Zones

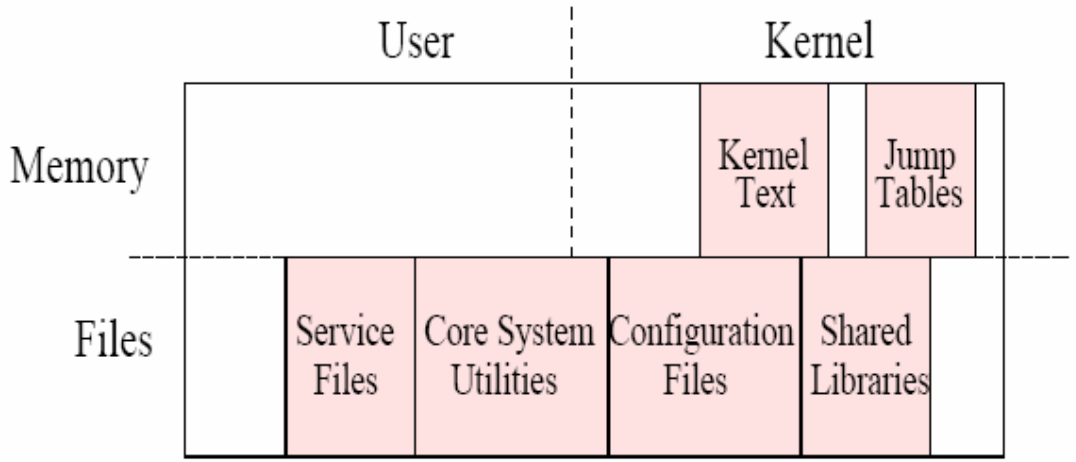


Fig 1: Protected Zones

```
/bin          RD_X
/sbin         RD_X
/boot         RD_X
/usr/bin      RD_X
/usr/sbin     RD_X
/etc/passwd   RD_ONLY
```

Fig 2: Sample policy file (protected files)

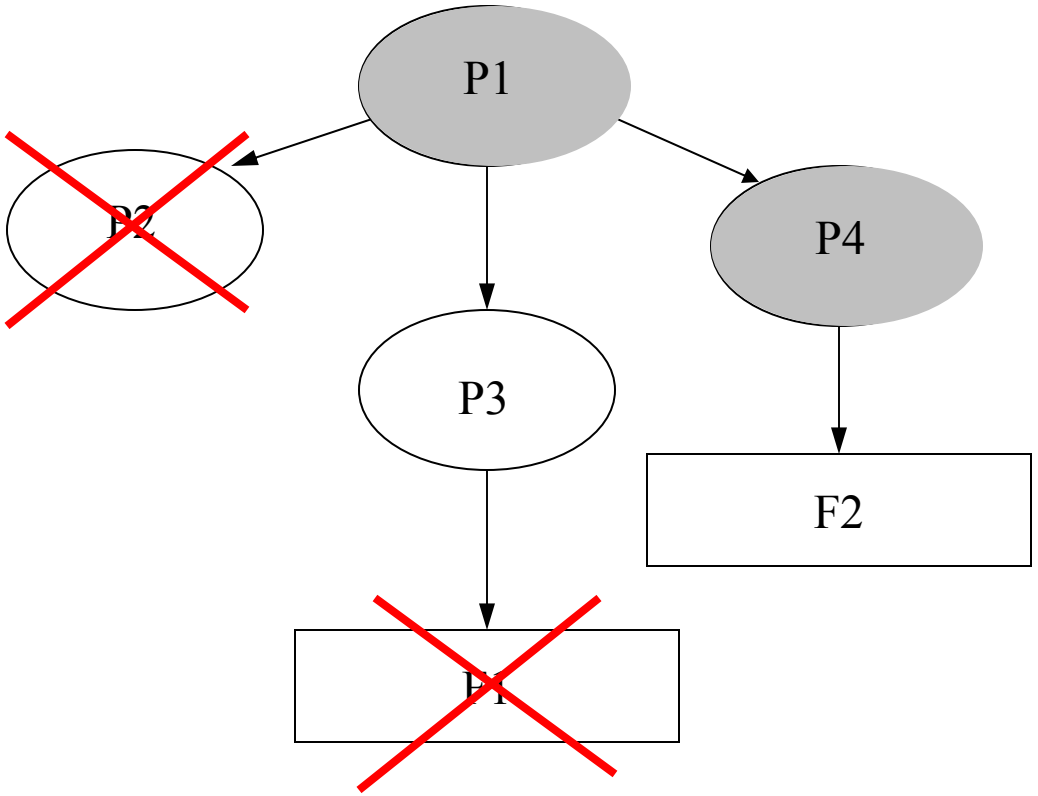
```
System call table
Interrupt descriptor table
Kernel text
```

Fig 3: Protected Memory Zones

Track Dependencies

- **Infer dependencies**
  - Parent-child relationships between processes.
  - Dependencies between files and processes.
  - Dependency rules ensure dependency tree is continuously pruned
- **Store dependencies**
  - Dependency tree is stored in a database.
  - Dependency tree size has to be small enough to provide fast response.

Dependency Rules



P1 creates P2

P2 exits

P1 creates P3

P1 creates P4

P3 creates F1

P4 creates F2

F1 is deleted

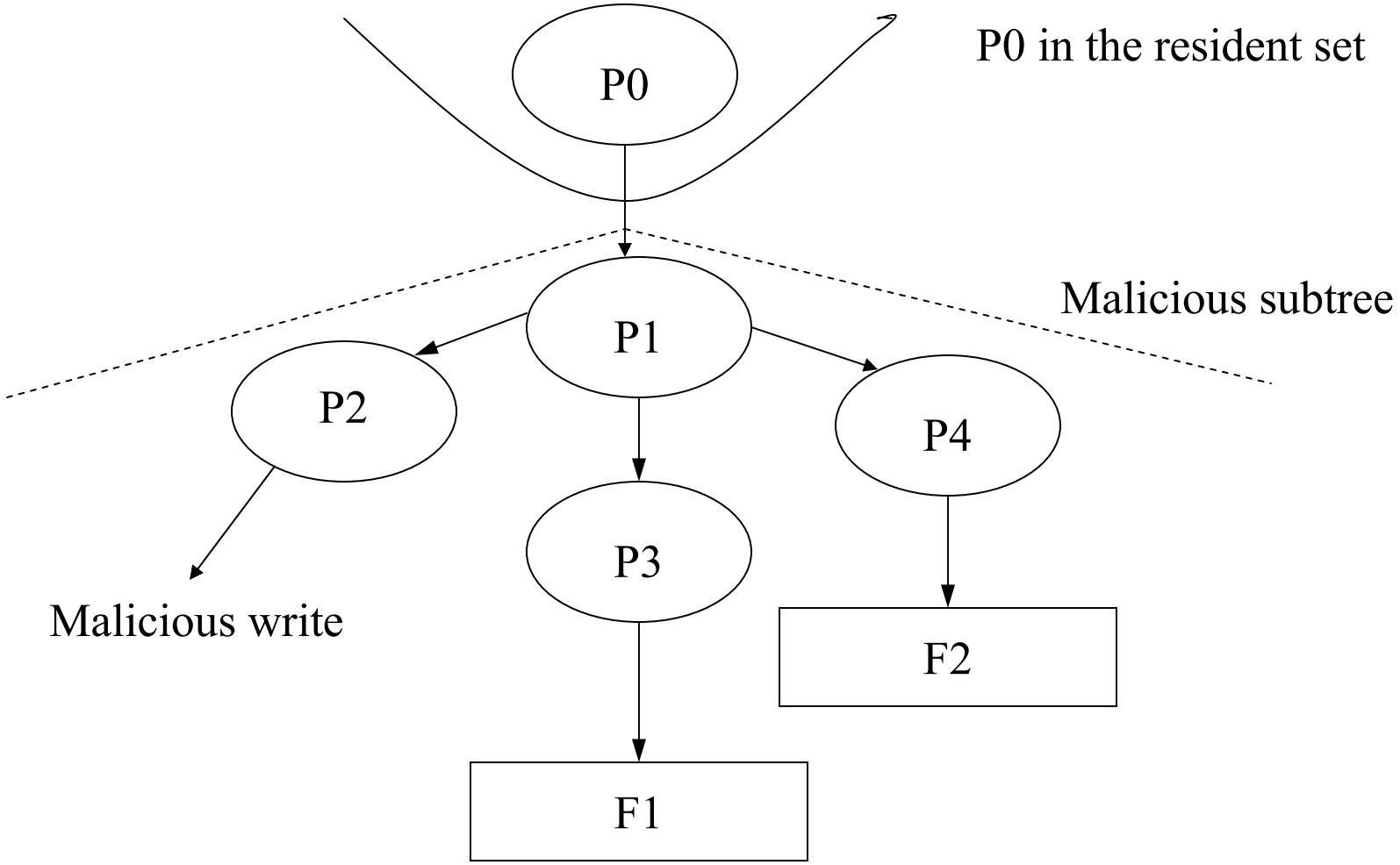
P4 exits

P1 exits

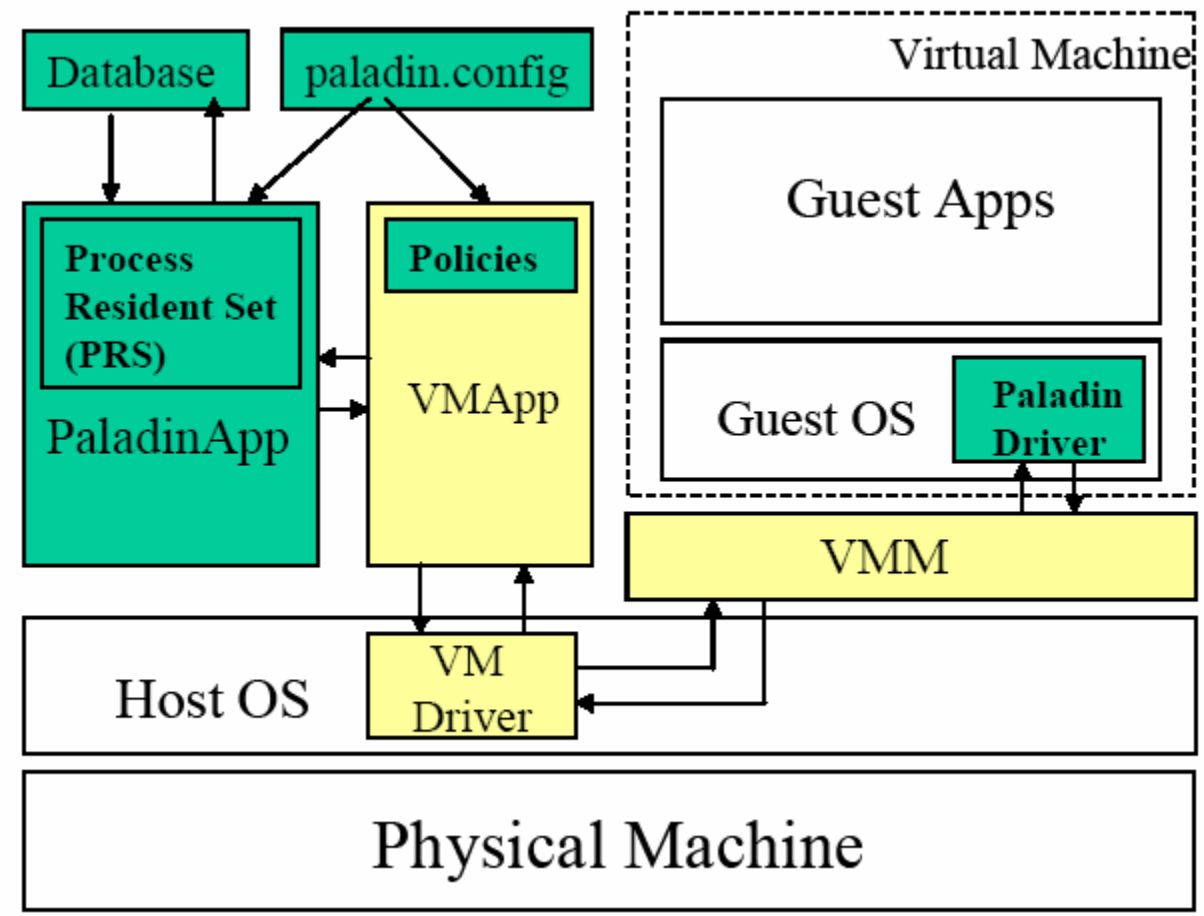
Containment

- Refers to dependency tree and locates malicious subtree
- Kills all possible malicious processes
- Stops ongoing damage
- Challenges
  - How to find the controlling parent ?
    - Traverse the dependency tree and assume the top most process is malicious
  - Which processes should be killed without killing the system ?
    - Assumes a **process resident set** that always exists in the system. (System daemons, processes like login, init)

Containment Algorithm



Prototype (Paladin*)



* A paladin is the prototypical ”knight in shining armour,” a hero of sterling character and courage, who rights wrongs and defends the weak and oppressed. (http://en.wikipedia.org/wiki/Paladin)

Performance Overhead

	Without Paladin	**With Paladin**
File Copy	**7 mins and 29 secs**	**8 mins 30 secs**
Compilation	**53 mins and 3 secs**	**56 mins and 7 secs**

VMware also implements a fast system call optimization that takes about 50% less time. Paladin currently is not implemented using this optimization.

Evaluation

Category 1	
Rootkit	Test Set
balaur 2.0	✓
lrk5 and variants	-
troier v 1.0	✓
cbr00tkit	✓
ark 1.0.1	✓
tl0gin	✓
flea	✓
torn 6.66	✓
tnet-tools v1.55	-
dica	✓
devNull v0.9	-
fk v0.4	✓
trNkit v1.0	✓
0x333openssh-3.7.1	-
sm4ck	✓

Category 2	
Rootkit	Test Set
linspy2	✓
rial	✓
rkit 1.01	✓
knark 2.4.3	✓
phide	✓
adore-0.42	✓
kis 0.9	✓
taskigt	✓
maxty	-
synapsys	✓
override	-
phalanx-b6	-
kbd v3	✓
all-root	✓
modhide	✓

Category 3	
Rootkit	Test Set
enyelkm v1.1	-
suckit	✓
superkit	✓
suckit2priv	✓
phantasmagoria	✓
adore-ng	-

Category 4	
Rootkit	Test Set
backdoor-caca	-

✓	Included in test set
-	Not included in test set

Fig. 6. Linux rootkits in the wild categorized by hiding techniques

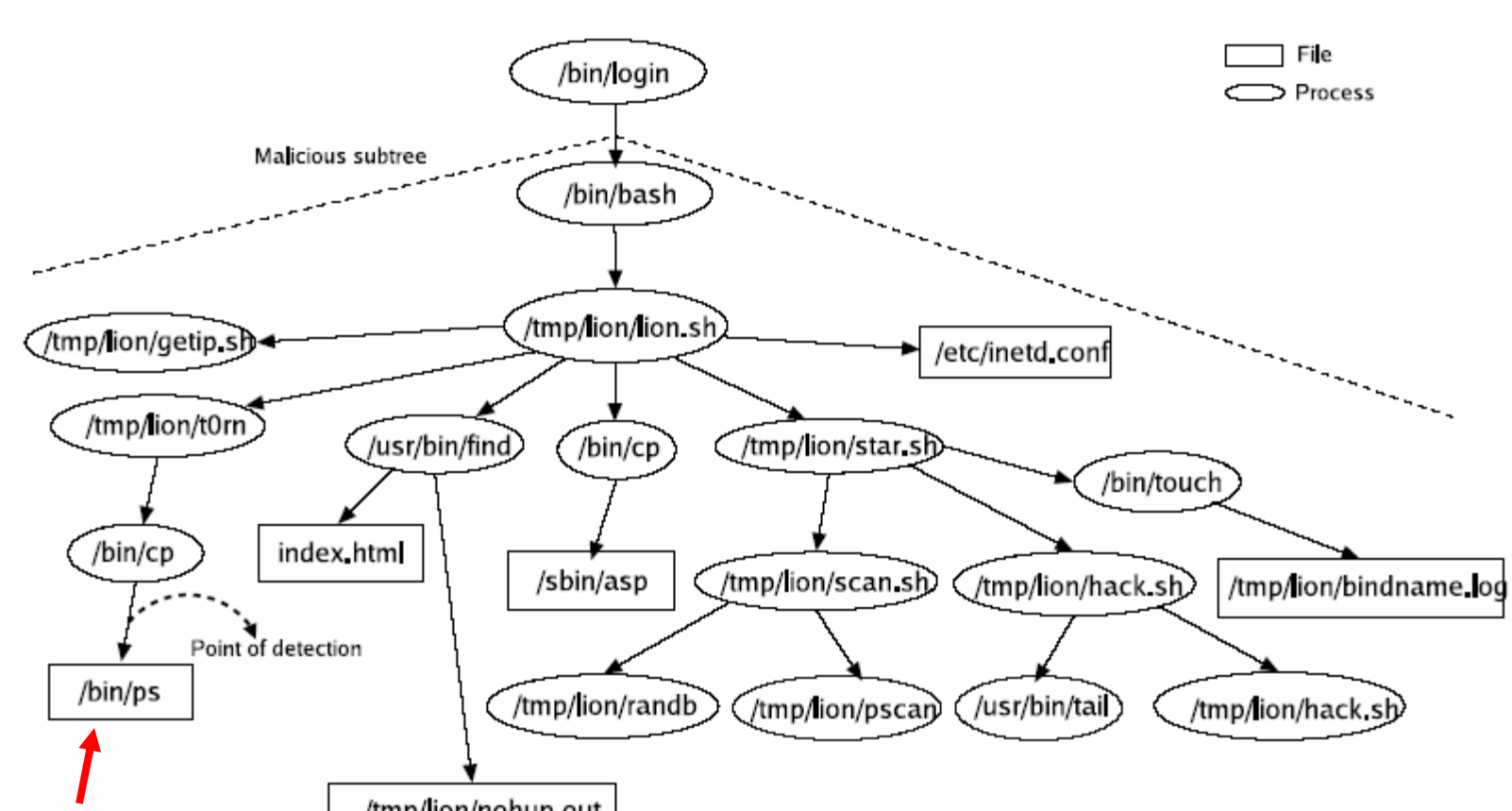
category 1: User-level - Install trojaned system binaries

category 2: Kernel-level - Modify the system call table

category 3: Kernel-level - Modify kernel text

category 4: Kernel-level - Modify interrupt descriptor table (IDT)

Lion Worm



Detection point

Conclusion

- We have designed an approach to detect and contain rootkit attacks and malware that use rookits to hide.

- We developed a prototype that demonstrated our approach and was tested against 27 rootkits available for Linux in the wild and a worm armed with a rootkit.

Thank You !

VM Security Model

First proposed by [Garfinkel' 2003].

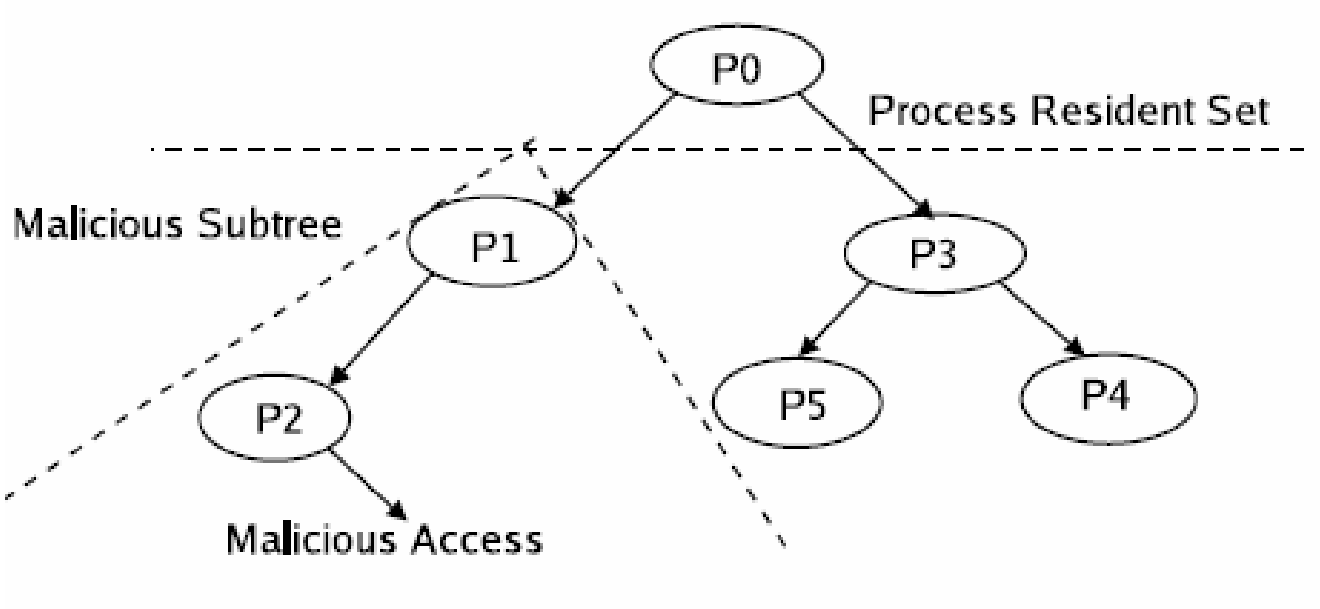
- Isolation
  - Isolates the Intrusion Detection System (IDS) from the Guest OS
  - Difficult for attacker to compromise IDS after gaining control of Guest OS.
- Introspection
  - IDS can inspect Guest OS state
- Interception
  - IDS can intercept events inside the VM

Implementation Details

- VMware workstation software (Type II VM)
  - Was implemented during my summer internship at VMware
- Database: MySQL
- Guest OS and Host OS: Linux 2.4 and 2.2 kernel.
- Paladin driver - Linux kernel module
  - Has knowledge of Guest OS kernel
  - Performs symbol lookup of kernel text and jump tables

Counter Attacks

- Multiple control processes



Counter Attacks (contd.)

- Writing directly to disk blocks
  - Can be overcome by storing the files in a separate data VM
  - This forces access through a well-defined interface.

- In-memory corruption of resident processes
  - Can be overcome by protecting the code pages of these processes.

Limitations

- Killing Legitimate Processes
- Accidental Modifications to protected files
- Rootkits that achieve stealth through modification of kernel data structures
  - FU rootkit

Related Work

- Model for Intrusion Detection
  - VMI
- Automated Detection
  - Copilot, Strider Ghostbuster
- Automated Post-Intrusion Analysis and Repair
  - Repairable File Service
  - Backtracker
- Automated Online Response
  - Introvert, Vigilante

Future Work

- Exploring solutions for Windows specific rootkits.
- Explore feasibility of automated recovery
- Automated fingerprint generation by examining file system access patterns
- Collaborative detection across VMs
  - Sharing attack fingerprints
  - Cross node detection

Future Work (contd)

- Exploring other detection mechanisms like anomaly detection (behavior based detection)
- Enhancing detection mechanisms by examining network packets.
- Exploring multi-core architectures for continuous monitoring performed from the other core.

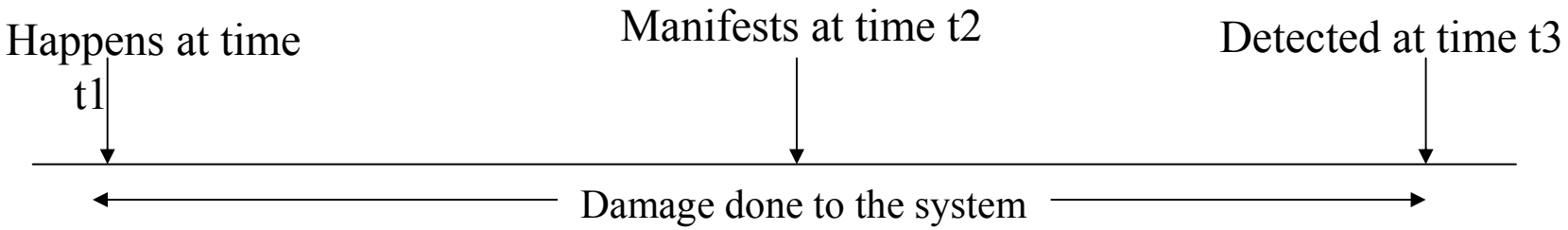
Windows Rootkits

- User-Level Rootkits
  - Hook to Import Address Table (IAT) or Export Adress Table (EAT)
  - Exploit OpenProcess API and extensibility hooks provided by Windows
- Kernel-Level Rootkits
  - System Service Table (SST) or IDT manipulation
  - Direct Kernel Object Manipulation (DKOM)

Virtual Machine Monitor

- Resides as a layer below the operating system
- Runs in the most privileged ring of the processor
- Multiplexes resources between several virtual machines
- Isolates the virtual machines running on it from each other
- Presents interface identical to the hardware that it runs on.

Intrusion Timeline



- Damage done to the system from t1-t3 needs to be discovered and undone – this takes time!
- Ideally intrusion should be detected at t1(Prevention)
  - Easier for known attacks
  - Hard for new/unknown attacks
- Intrusion Detection Systems (IDS)
  - Move t3 closer to t2
  - Ideally move t3 close to t1

Infer Dependencies

- On fork
  - Child has same <**esp,eip**> but different **cr3**
  - When processes executed from the same program, fork at a given point, all children have same <**esp,eip**>
  - Ambiguities are resolved using physical page number of stack page
  - Relies on Linux fork copy-on-write semantics

- Apply dependency rules

Attack methods

- Exploiting software vulnerabilities
  - Buffer overflow still the most common exploit !
- Prying on ignorance of lay users
  - Email viruses, trojaned downloads, phishing attacks
- Lack of security measures taken by users, human configuration errors, insufficient skills/training
  - Firewall misconfiguration, anti-virus with old signature files
- Lack of appropriate tools to verify if system is secure/compromised
- Insider attacks

Detection

- Tools available for rootkit detection
  - Chkrootkit, F-Secure BlackLight, RootkitRevealer
- Copilot
  - Automated detection from an independent PCI device [Petroni ‘ 2003]
- Strider Ghostbuster
  - A cross-view diff-based approach. [Wang ‘2005]

Post-Intrusion Analysis and Repair

Aid to the administrator

- Fix the file system and keep good changes
- Find how the intrusion happened

- RFS, Taser
  - Design, Implementation and Evaluation of Repairable File Service [Zhu ‘ 2003]
  - The Taser Intrusion Recovery System [Goel’ 2005]
- BackTracker
  - Backtracking Intrusions [King ‘ 2003]

Online Response

Provides online defense against different types of intrusions

- Introvert
  - Detecting Past and Present Intrusions Through Vulnerability-Specific Predicates [Joshi' 2005]
- Vigilante
  - Vigilante: End-to End Containment of Internet Worms [Costa ' 2005]