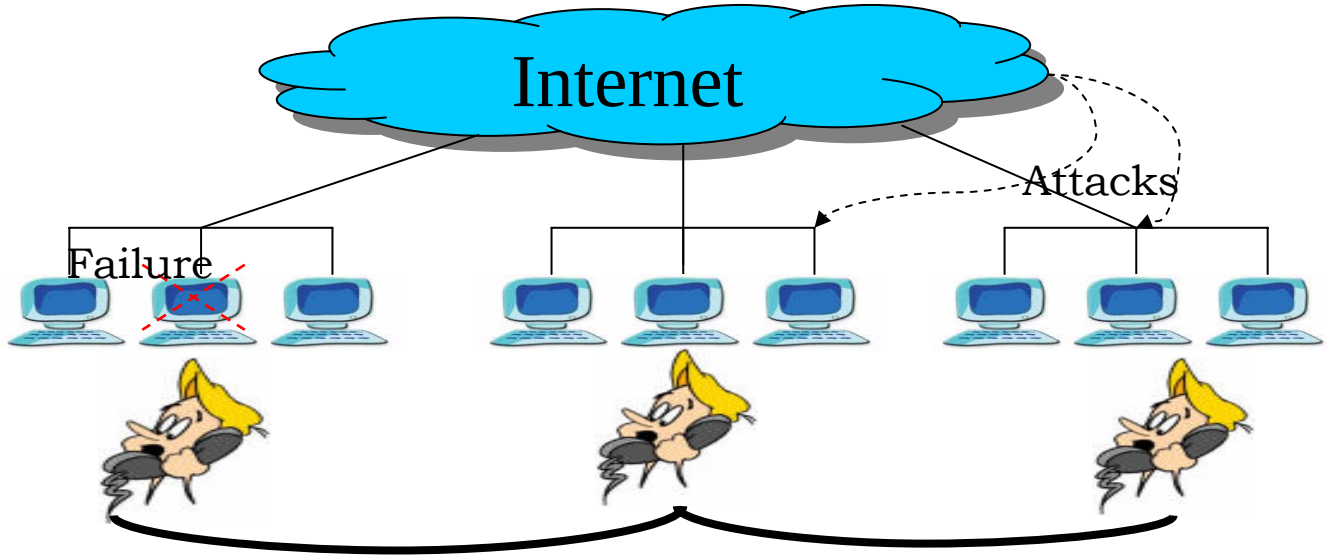
Towards Automated Defense from Rootkit Attacks

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

Motivation

- With the increasing attack trends, human response to intrusions is very slow.
  - SQL Slammer
- We need systems that can automatically detect and recover from intrusions without human intervention.



Viruses/Worms

- Viruses have the ability to replicate by modifying a normal program/file with a copy of itself.
  - Execution in the host program/file results in the execution of the virus
  - Usually needs human action to execute infected progam.
- Worm is a stand-alone program. Spreads itself through the network by exploiting vulnerabilities in services.

Rootkits

Collection of tools used by the attacker to maintain root on the compromised system. Particularly important as they compromise system integrity.

Types of rootkits based on their hiding mechanism:

- User-level rootkits (Shared library rootkits)
  - Replace system binaries like ps and netstat
- Kernel rootkits
  - Replace entries in system call table
  - Replace entries in interrupt descriptor table
  - Replace kernel/module text.

Stealth Malware

- Increasing number of virus/worm writers use rootkits to evade detection from anti-virus software.
- Our approach can contain stealth malware that hide using rootkit techniques.
- Stealth AOL worm

Dealing with Rootkit Attacks

- Security Model
  - The intrusion detection system (IDS) should be independent of the host.
  - Compromising the host should not lead to a compromise in the IDS
- Common Types of Systems
  - Stand-alone Systems
  - Virtualized Environment

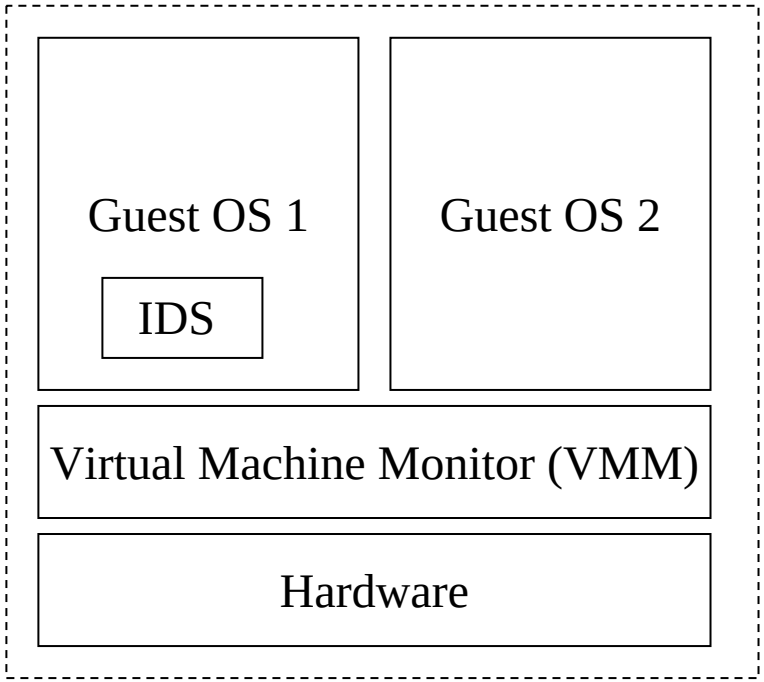
Intrusion Recovery System Location

- Stand-alone systems
  - Independent secure device (secure coprocessor)
    - Polling based approaches
    - Non-intrusive (better performance)
- Virtualized Environment
  - Provides a good security model
  - Near-native performance.

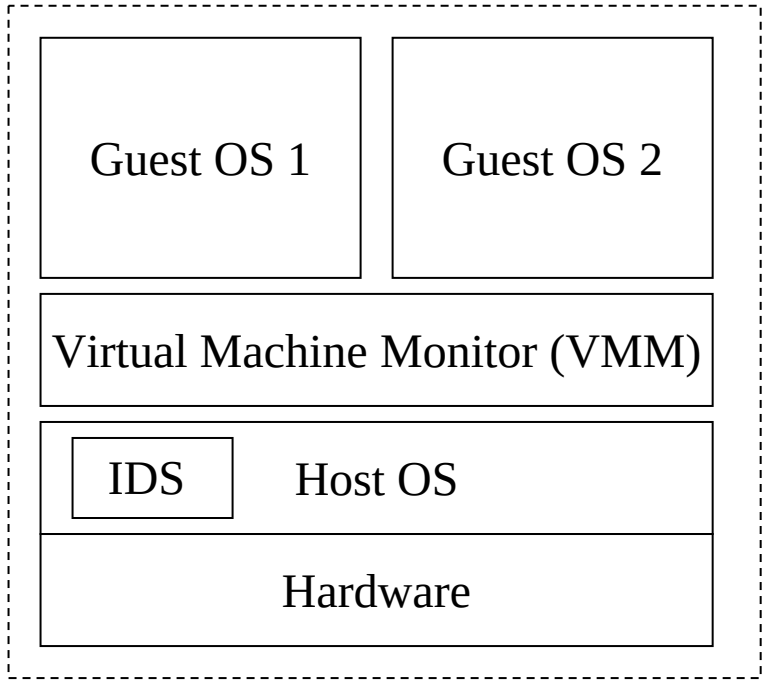
Virtual Machines

- Have regained popularity
- Mainly used for
  - Hosting Web services
    - Lowers costs by consolidating idle system resources
    - Ease of maintenance and operational efficiency
  - Cross-platform development.
    - Supports multiple operating systems on the same machine.
    - Ease of testing and debugging.

Virtual Machines



Type 1 VMM



Type 2 VMM

- Can intercept interesting events in the virtual machine
- Can interpose and change virtual machine (VM) state to perform preventive/healing actions.

Our Approach

- **Detect** malicious process trying to perform illegal access.
  - **Prevent** illegal access to ***protected zones***.
  - **Track** dependencies between files and processes.
- **Contain** the effects of the attack
  - Identify and kill malicious processes.
- **Fingerprint** attacks by tracking simple changes to the filesystem.
  - Manual inspection of affected files for removal/quarantine.
  - Early attack identification and containment.

Automated Detection

- **Detect** malicious process trying to perform illegal access.
  - **Prevent** illegal access to ***protected zones***.
  - **Track** dependencies between files and processes.

Protected Zones

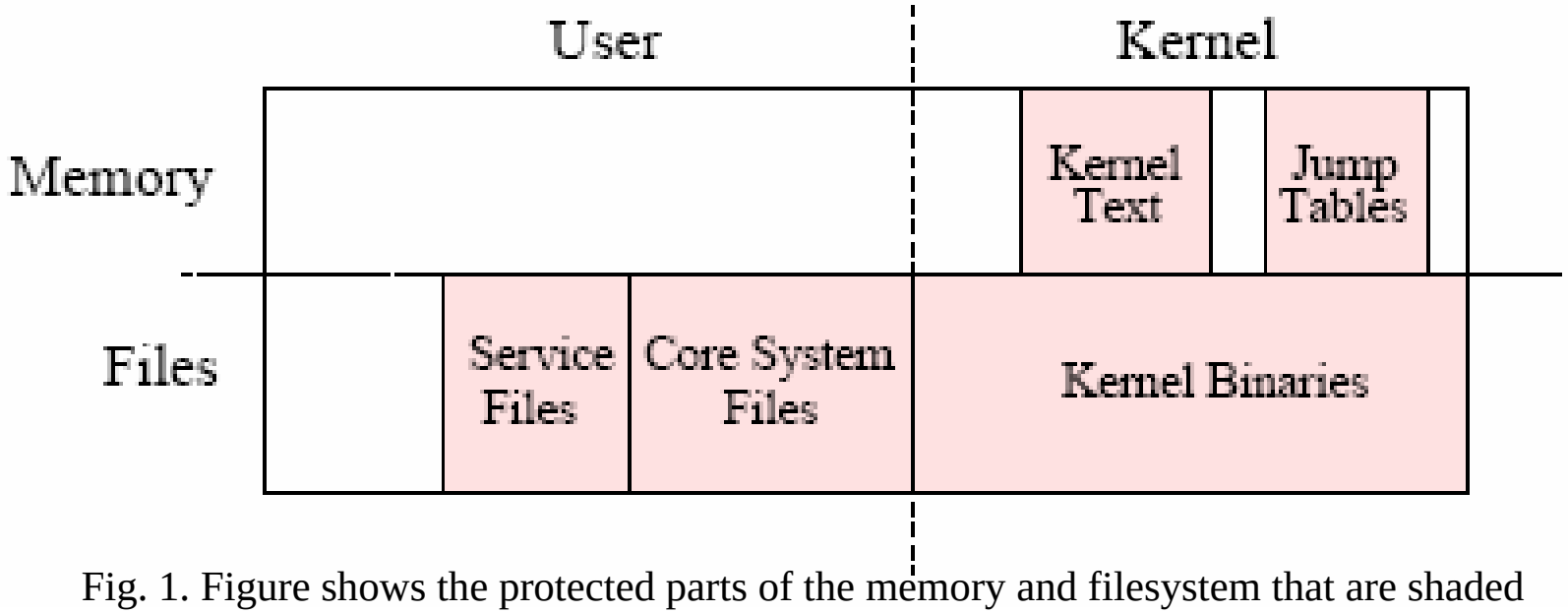


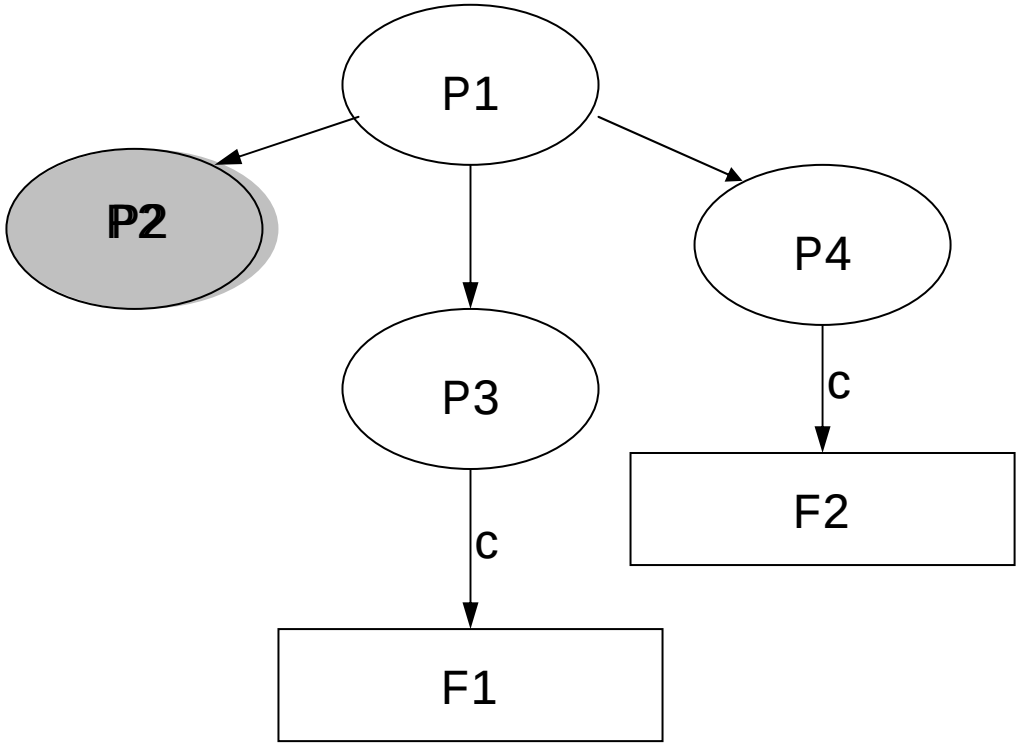
Fig. 1. Figure shows the protected parts of the memory and filesystem that are shaded in pink. This represents the core of the system, which is always protected. The unshaded portions consist of all other files and running programs, which can be compromised at any time.

- **Prevent illegal access to protected zones.**

Tracking

- **Infer dependencies**
  - Track dependencies between files and processes.
  - Parent-child relationship between processes.
- **Store dependencies**
  - As a tree of relationships (dependency tree)
  - Dependency tree is stored in a database.
  - Dependency tree size has to be small enough to provide online automated containment.

Dependency Tree



P1 creates P2

P2 exits

P1 creates P3

P1 creates P4

P3 creates F1

P4 creates F2

F1 is deleted

P4 exits

P1 exits

Dependency Storage

- Size of the dependency tree created is linearly proportional to the number of new files created on the file system.
- Storage requirements are modest.

Automated Containment

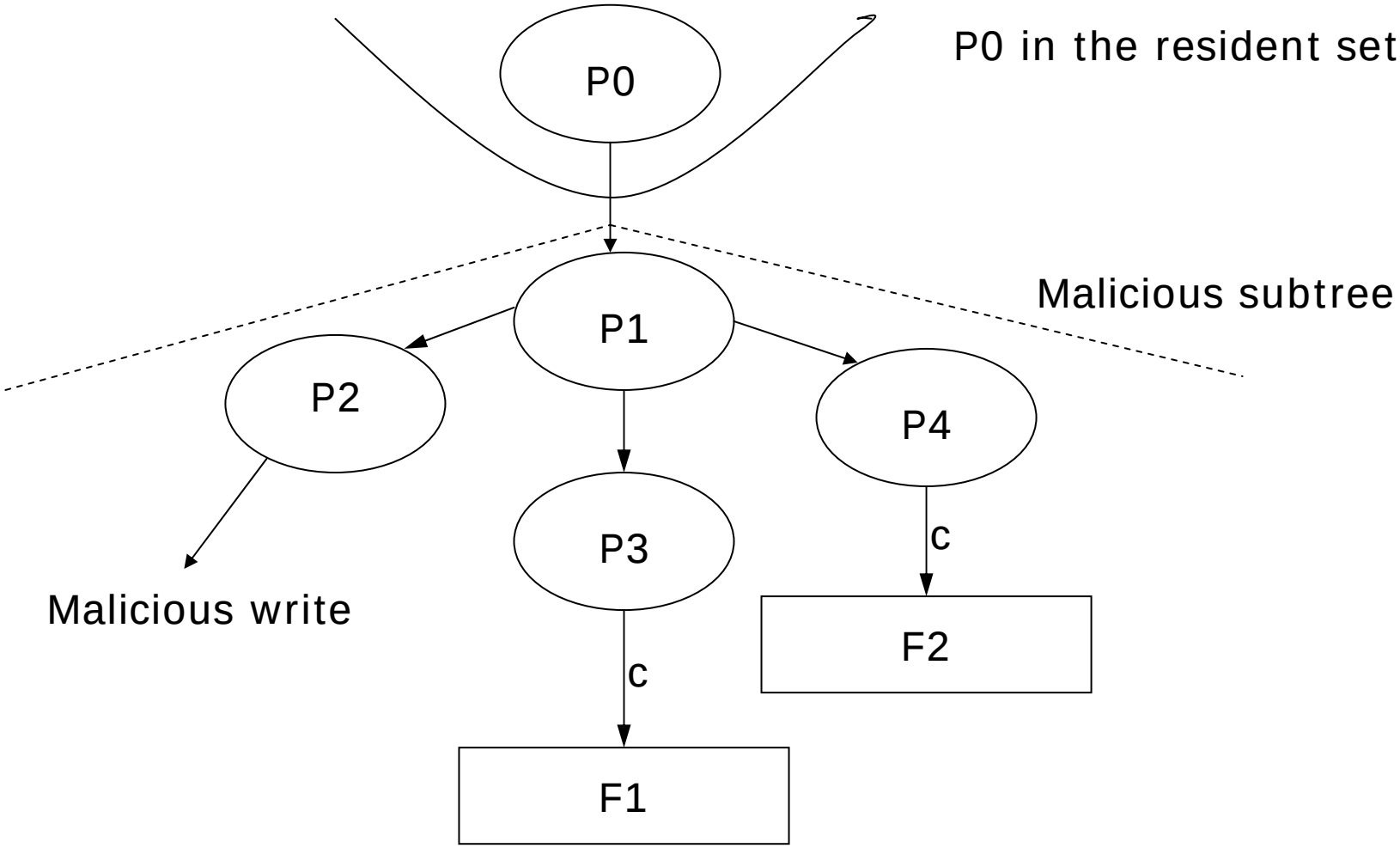
- **Detect** malicious process trying to perform illegal access.
  - **Prevent** illegal access to ***protected zones***.
  - **Track** dependencies between files and processes.

- **Contain** the effects of the attack by identifying and killing malicious processes.

Containment

- Identify and kill malicious processes.
- Assumes a **resident process set** that always exist in the system.
- Prevents ill-effects
  - Installation/Existence of backdoors.
  - Keyloggers

Containment Algorithm



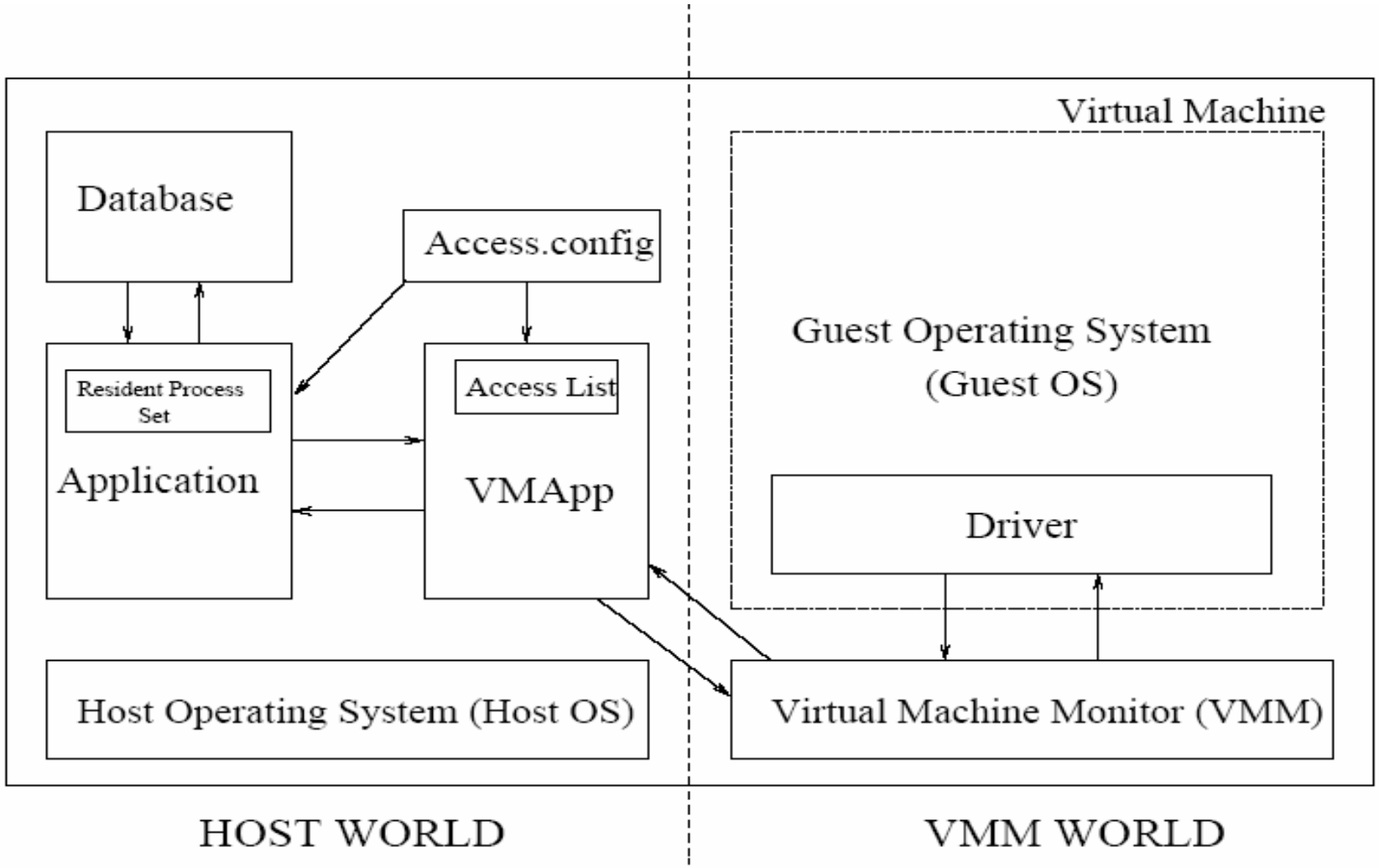
Automated Fingerprinting

- **Detect** malicious process trying to perform illegal access.
  - **Prevent** illegal access to ***protected zones***.
  - **Track** dependencies between files and processes.
- **Contain** the effects of the attack by identifying and killing malicious processes.
- **Fingerprint** attacks by tracking simple changes to the filesystem.
  - Manual inspection of affected files for removal/quarantine.
  - Early attack identification and containment.

Fingerprinting the attack

- Dynamic cloning
  - Spawn a clone upon attack detection
- Sandboxing
  - Reconfigure network properties
- Fine-grained monitoring
  - Watch the processes in the malicious subtree
  - Finer control possible.

Prototype


Virtual Machine
Database
Access.config
Guest Operating System
(Guest OS)
Resident Process Set
Access List
Application
VMApp
Driver
Host Operating System (Host OS)
Virtual Machine Monitor (VMM)
HOST WORLD
VMM WORLD

Performance Evaluation

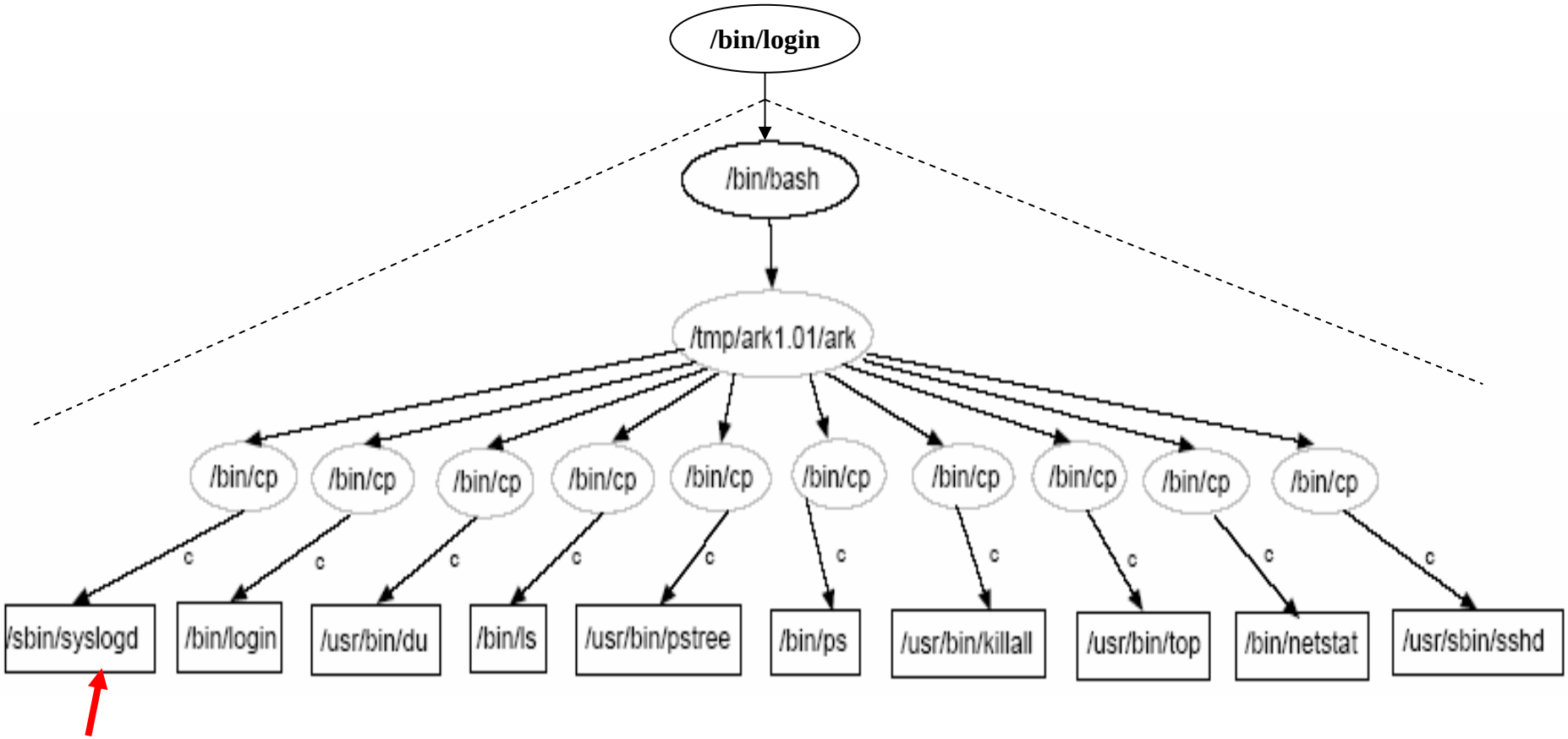
Implemented this framework using VMware workstation software. The database was located on a separate machine. Guest and Host OS were Linux 2.4 kernel.

	Optimized VMM No detection possible	**System calls trap inside VMM** **Without security framework**	**System calls trap inside the VMM** **With security framework**
File Copy	3 mins and 46 secs	**7 mins and 29 secs**	**8 mins 30 secs**
Compile Kernel	31 mins and 54 secs	**53 mins and 3 secs**	**56 mins and 7 secs**

Evalution

- User level rootkit
  - Ambient Rootkit (ARK)
- Kernel Rootkit
  - SuckIt
- Linux Worm
  - Lion

Ambient Rootkit (ARK)



Point of detection

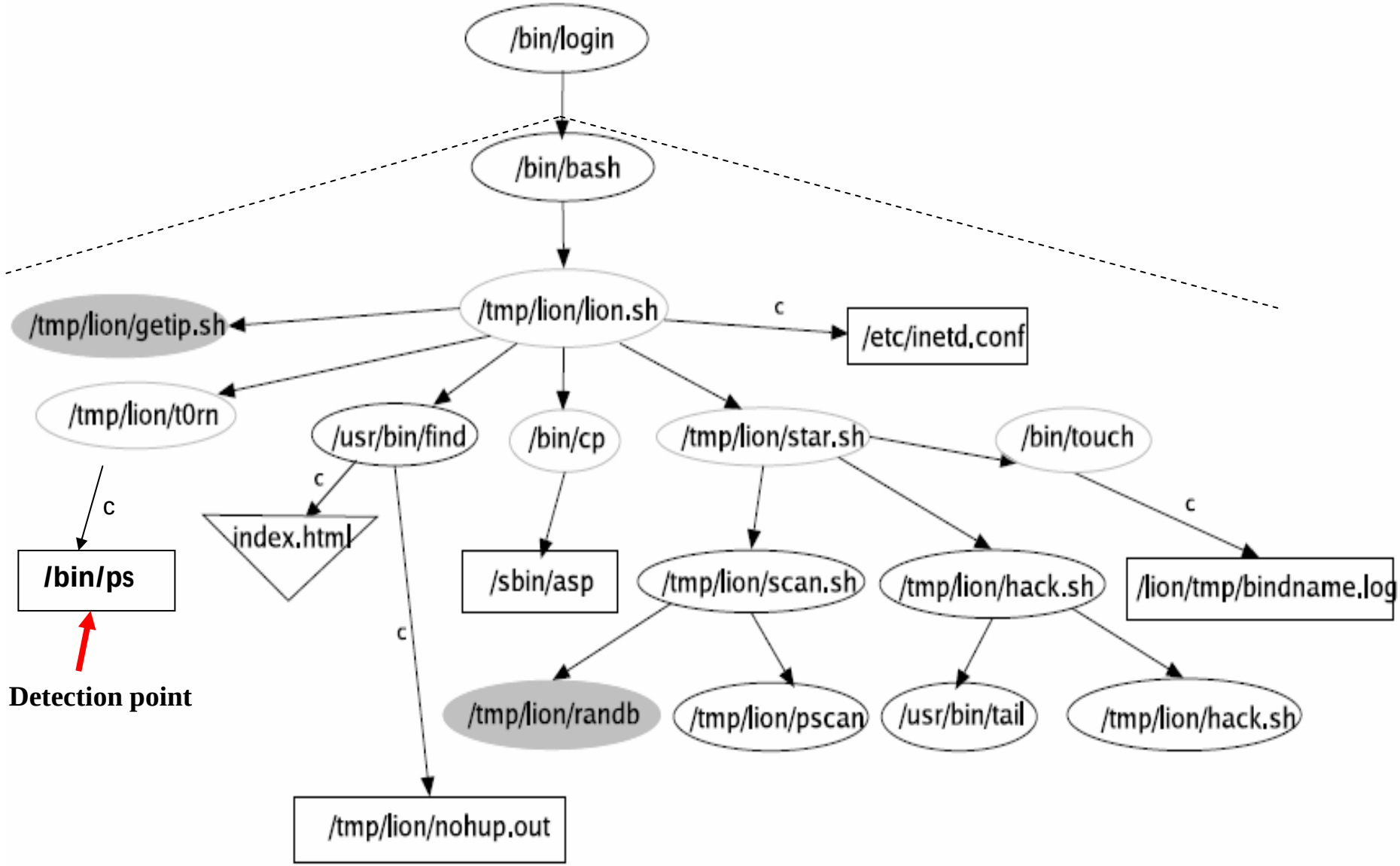
The security framework prevents corruption of all the system tools

SuckIt

- Kernel rootkit that modifies instructions in sys_call to redirect to it's own system call table.
- Does not touch original system call table
- Uses /dev/mem /dev/kmem interfaces
- Provides covert backdoor that is activated on receiving a special packet.

- The security framework prevents corruption of kernel text and installation of the backdoor.

Lion Worm


/bin/login
/bin/bash
/tmp/lion/lion.sh
/tmp/lion/getip.sh
c
/etc/inetd.conf
/tmp/lion/t0rn
/usr/bin/find
/bin/cp
/tmp/lion/star.sh
/bin/touch
c
c
index.html
/bin/ps
/sbin/asp
/tmp/lion/scan.sh
/tmp/lion/hack.sh
/lion/tmp/bindname.log
c
Detection point
/tmp/lion/randb
/tmp/lion/pscan
/usr/bin/tail
/tmp/lion/hack.sh
/tmp/lion/nohup.out

ARK Fingerprint

Processes:	**Files created**
/tmp/ark1.01/ark	**/dev/capi20.20**
/bin/rm	**/sbin/syslogd**
/sbin/syslogd	**/usr/lib/.ark?**
/bin/cp	**/bin/login**
/usr/sbin/sshd	**/usr/sbin/sshd**
/bin/chmod	**/bin/ls**
/bin/cat	**/usr/bin/du**
/bin/hostname	**/bin/ps**
/sbin/ifconfig	**/usr/bin/pstree**
/bin/grep	**/usr/bin/killall**
/bin/awk	**/usr/bin/top**
/bin/sed	**/bin/netstat**
/sbin/modprobe	/var/run/syslogd.pid
/usr/lib/sendmail	/var/spool/clientmqueue/dfj99KxukX001449
/usr/lib/libhesiod.so.0	/var/spool/clientmqueue/tfj99KxukX001449
	/var/spool/clientmqueue/dfj99L0HiX001457
	/var/spool/clientmqueue/tfj99L0HiX001457
	/var/spool/clientmqueue/dfj99L0cv1001466
	/var/spool/clientmqueue/tfj99L0cv1001466
	/var/spool/clientmqueue/dfj99L0w2M001475
	/var/spool/clientmqueue/tfj99L0w2M001475

Future Work

- Performance Optimizations
  - Speed-up system call trapping inside the VMM.
- Implement fingerprinting.
  - Early attack identification through fingerprint matching
    - Find them before they hide
  - Automated identification of attacker's files
  - Collaborative protocols between VMs to share attack fingerprints.

Related Work

- Automated Detection
  - Copilot, VMI, Strider Ghostbuster, Tripwire
- Automated Post-Intrusion Analysis and Repair
  - Repairable File Service
  - Backtracker
- Automated Containment and Fingerprinting
  - Introvert
  - **Paladin**

Automated Detection

- Tools available for rootkit detection
  - Kstat, Chkrootkit, St. Michael, Samhain, F-Secure BlackLight, RootkitRevealer, Tripwire, AIDE
- Copilot
  - Automated detection from an independent PCI device [Security '03]
- Strider Ghostbuster
  - A cross-view diff-based approach. [DSN '05]
- VMI
  - Virtual Machine based Introspection (VMI) for Intrusion Detection [NDSS '03]

Post-Intrusion Analysis and Repair

Aid the administrator in

- Fixing the filesystem by keeping good changes
- Finding how the intrusion happened

- RFS, Taser
  - Design, Implementation and Evaluation of Repairable File Service [DSN ‘03]
  - The Taser Intrusion Recovery System [SOSP ‘05]
- BackTracker
  - Backtracking Intrusions [SOSP ‘03]

People

Arati Baliga, Rutgers University

Liviu Iftode, Rutgers University

Mike Chen, VMware Inc.

Thank You !