

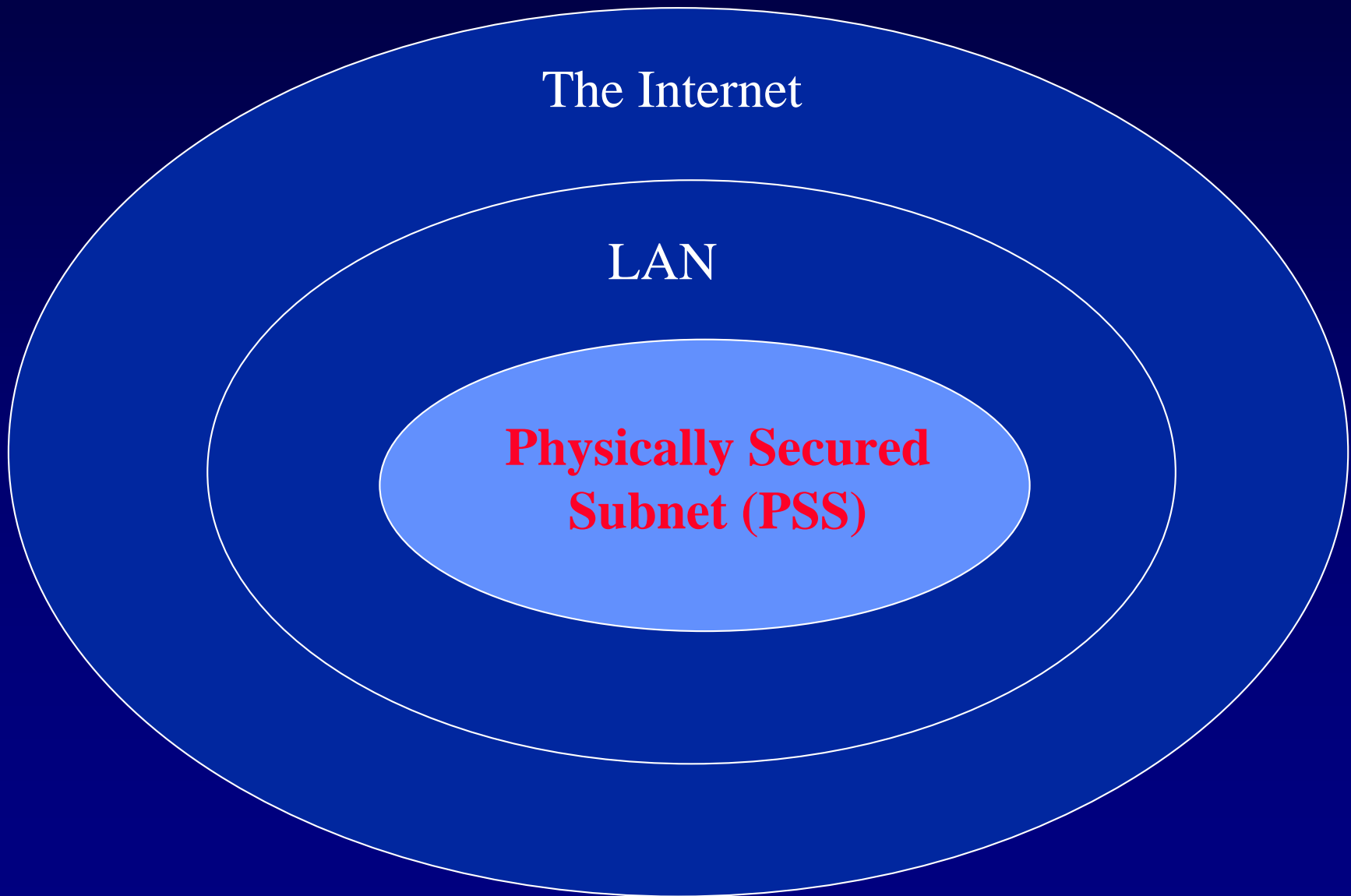
Tracing the Root of a *Rootable* Process

Amit Purohit, Vishnu Navda,
Tzi-cker Chiueh

Introduction

- Many existing access control checks are based on some notion of password
- These checks cannot protect a system against attacks using compromised passwords, which could come from password cracking, social engineering, or insider attack
- What is the main differentiator between a remote attacker and a legitimate user? **Physical access**
- Key idea: Restrict critical privileges to users that can physically log into a subset of hosts protected by physical security

Physically Secure Subnet (PSS)



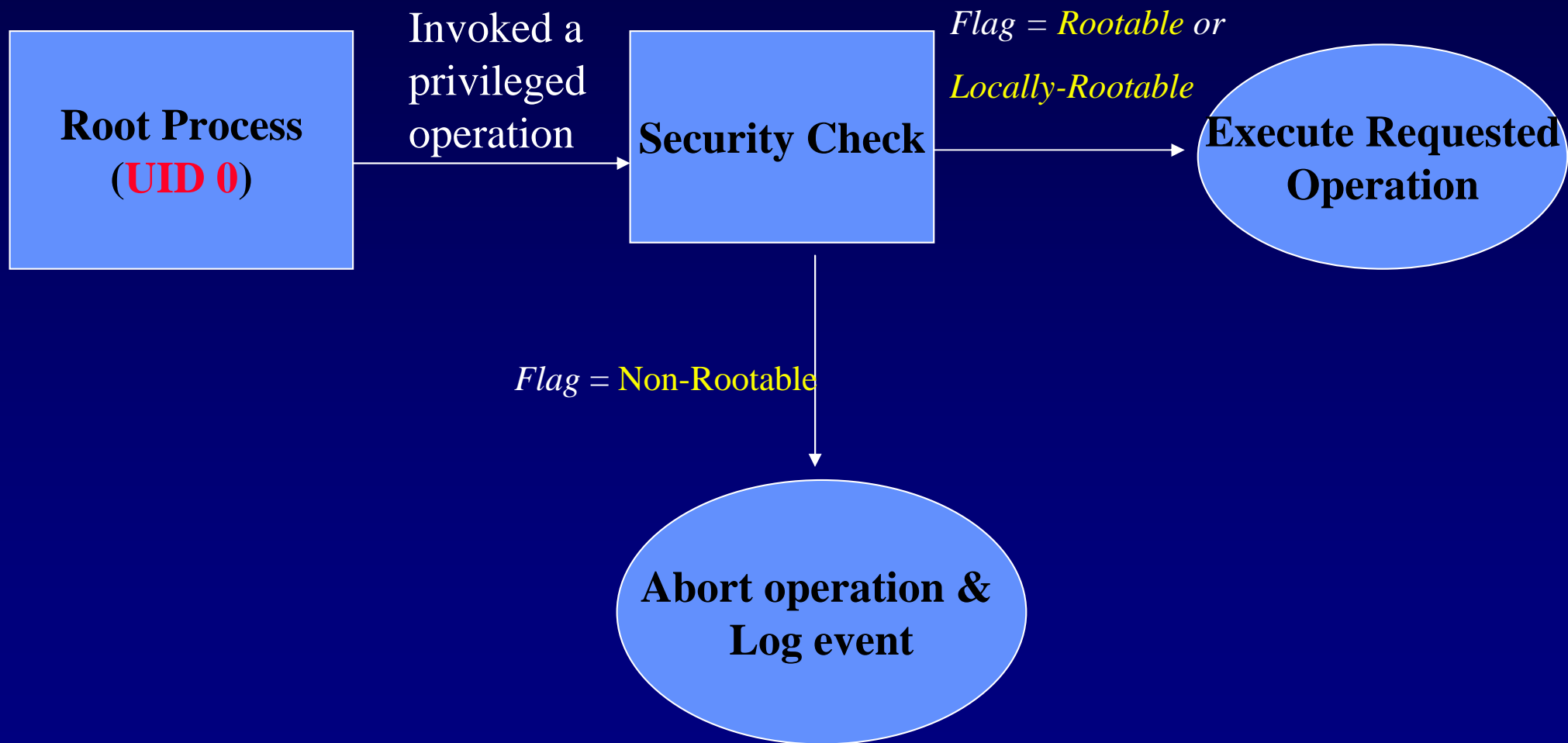
Proof of Physical Access

- Users are allowed to perform high privileged operations only if they can show that they have physical access to hosts belonging to PSS
- To enforce this we associate each process in the system with an additional **privilege** flag
- How to assign and propagate privilege flags?

Value of Privilege Flag

- **Rootable**: can perform privileged operation on local host, and the privilege can be propagated to other hosts
- **Locally Rootable**: can perform privileged operation on local host, but the privilege cannot be propagated to other hosts
- **Non-Rootable**: cannot perform any privileged operation

Privilege Flag Usage



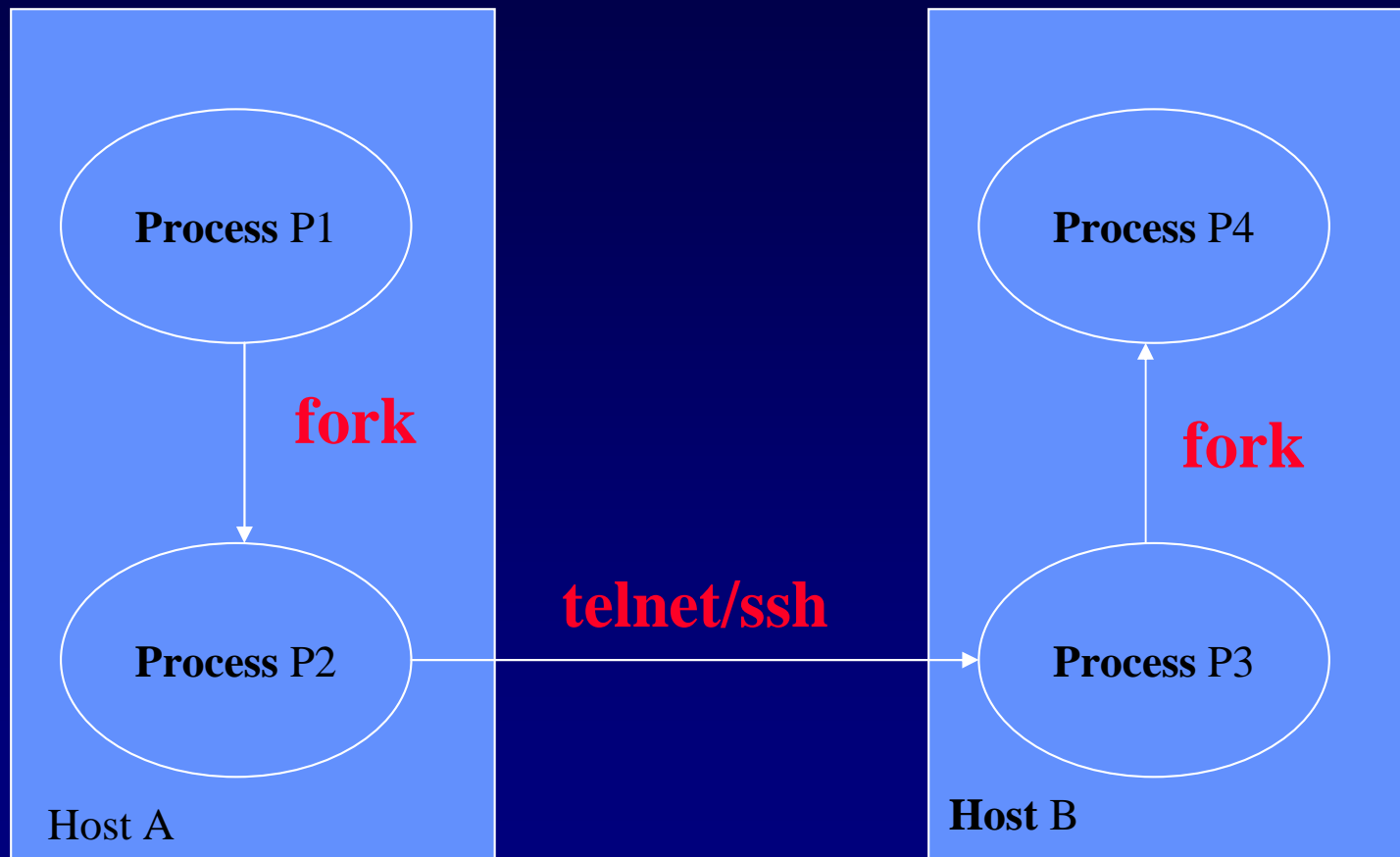
Assignment of Privilege Flag

- The privilege flag of a process that originates from a console login on a PSS machine is Rootable
- The privilege flag of a process that originates from a console login on a non-PSS machine is Locally Rootable
- All daemons that are started at boot-up time and require root privileges are Locally Rootable
- Everything else is Non-Rootable

Privilege Flag Value Propagation

- **Vertical** Propagation
 - From parent to child process on a local host
- **Horizontal** Propagation
 - Between processes belonging to different hosts
 - This can happen through various kinds of network applications. Here we focus on remote login programs such as telnet/ssh

Propagation of Privilege Flag



P1 à P2: VERTICAL Propagation

P2 à P4: HORIZONTAL Propagation

Rules of Propagation

Remote Host	Source Privilege Flag	Destination Privilege Flag
In PSS	Rootable	Rootable
In PSS	Locally Rootable	(Non-)Rootable ?
In PSS	Non-Rootable	Non-Rootable
Out of PSS	Rootable	Rootable
Out of PSS	Locally Rootable	Non-Rootable
Out of PSS	Non-Rootable	Non-Rootable

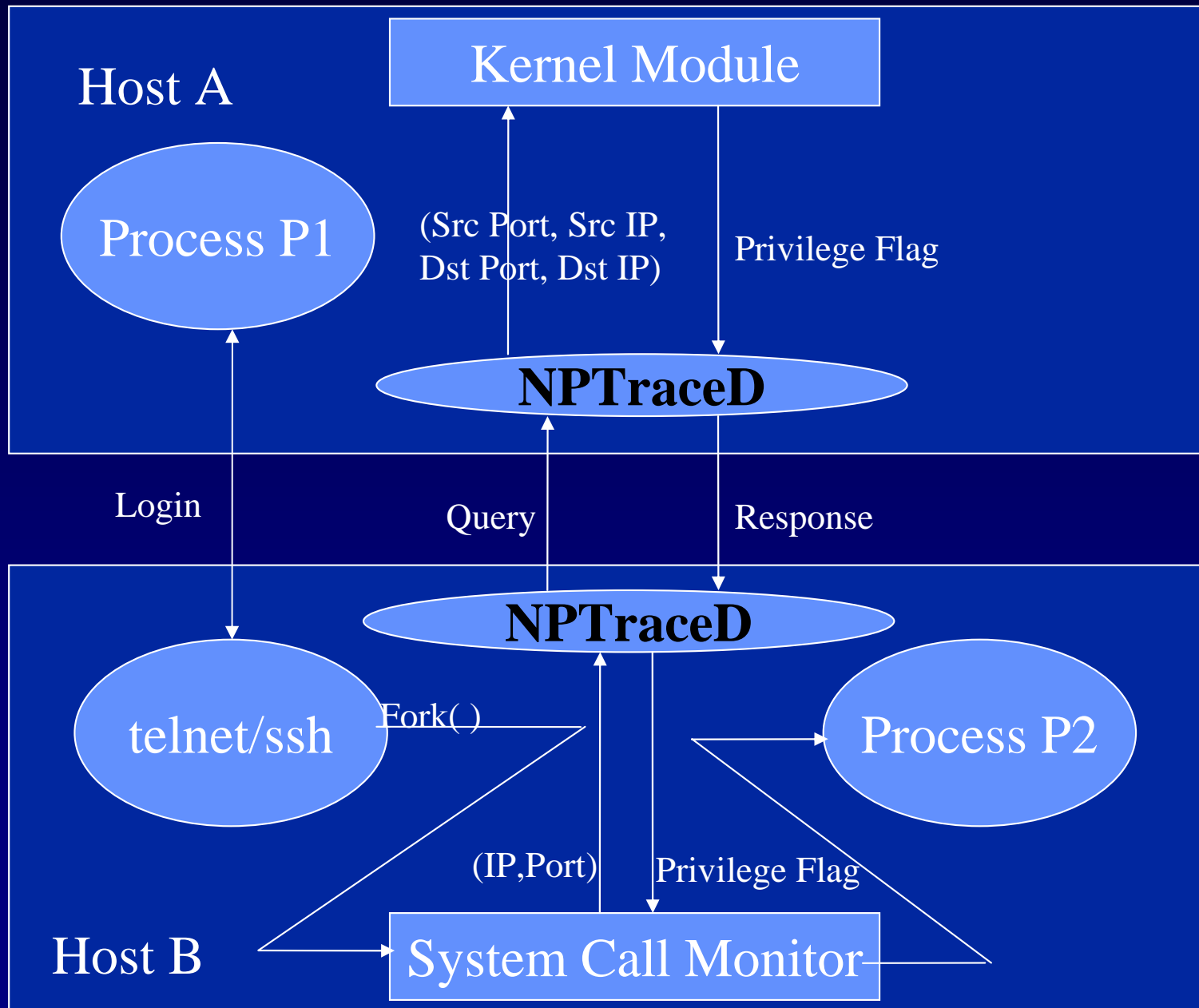
Invariant

- *All rootable processes can be traced back to a process started by a user that has physically logged in to one of the PSS hosts*
- *Therefore, no remote attackers can perform privileged operation even if they log into a remote machine as root*

NPTrace Implementation

- Authorization check includes privilege flag
- When process P forks a new process Q,
 - Q's flag is Rootable if P is a local login program, the input is directly from keyboard, and the host is in PSS -> **Initial assignment**
 - Q's flag is Locally Rootable if P is a local login program, the input is directly from keyboard, and the host is not in PSS -> **Initial assignment**
 - Q's flag is the same as that of a remote process if P is a telnet/ssh daemon -> **Horizontal propagation**
 - Q's flag is the same as that of P otherwise -> **Vertical propagation**

Application Independence



Implementation Issues

- Need to identify telnetd/sshd to apply horizontal propagation rules
 - Telnetd/sshd listen on well-known ports. So we can intercept `sys_bind()` to find which are the processes listening on those ports.
 - Modify the daemon startup scripts to register its pid
 - Rely on the binary image name/path of the process
- Mapping a socket back to the process that is using that socket

Attack Analysis I

- What if an NPTrace daemon on a non-PSS host is compromised and mark all local processes Rootable?
 - Process that creates a Rootable flag also generates a unique key
 - Pass the key on along with the Rootable flag
 - Check with the origin process in PSS, which keeps track of all its Rootable children, about the validity of the key

Attack Analysis II

- Cannot prevent control-hijacking attacks such as buffer-overflow attacks against Locally Rootable daemons
 - However, such breaches cannot generate Rootable processes, only Locally Rootable ones
 - Backdoor programs cannot allow attackers to log into other machines

Attack Analysis III

- What if a PSS machine is compromised?
 - May need to move NPTrace daemon to the kernel and demand that reboot also require physical access
- Difference between scope of a root password and proof of its origin

Performance Overhead

- Additional login overhead under SSH:
12.5 msec
 - Consulting with the NPTrace daemon in the previous hop about the remote process's privilege flag and associated key
 - Checking with the origin process in PSS about the validity of the key
 - These communications are based on OpenSSL and secret key

Conclusion

- Physical security is our friend -> similar idea can be used to protect laptops or desktops
- With NPTrace, knowing root password is not good enough; need to have proof of physical access
- Not a panacea, but an effective mechanism to counter compromised passwords
- Should be applied to all critical accesses guarded by passwords, such as database, system configuration