Security Enhanced Linux

Security Group Meeting
29 November 2002

Steven J. Murdoch

http://www.cl.cam.ac.uk/users/sjm217/

Computer Laboratory, University of Cambridge

Summary

- Introduction and Predecessors to SELinux
- Policy Structure
- Software Architecture and Potential
- Example of Policy Configuration

Project Goals 1

- Support confidentiality and integrity requirements
- Fine grained control (compared to standard POSIX privileges)
  - Greater range of permissions to be granted (not simply read, write, execute)
  - Greater range of objects controlled (files, sockets, network interfaces)
  - Greater range of trust in users (no “root” user)

Project Goals 2

- Flexible policy configuration
  - Features:
    - Separation of data and duty
    - Confidentiality
    - Containment of potentially flawed programs
    - Integrity of data and applications
    - Ensure data is processed as required
  - Multi-Level Security (MLS) is not enough

Project Goals 3

- Additional information available for security decisions (as well as User ID and file ownership)
  - Role of the user
  - Function of the program being used
  - Trustworthiness of the program being used
- Mandatory Access Control (MAC)
  - System-Wide security policy

Project Goals 4

- Minimal privilege for each program
  - Child processes may have less privilege than the parent
- Extensible and flexible system architecture
- Integrated with a mainstream operating system
- Small performance overhead
- Some formal verification of architecture's security properties

Previous projects

- DTMach
- DTOS
- Fluke
    - University of Utah, Flux Research Group
- Flask Architecture
- SELinux

Software architecture

- Enforcement and policy separate
- Policy encapsulated by “Security Server”
- Enforcement performed by “Object Managers”
- Configuration language also defined by Security Server
- Flask architecture defines API of Security Server
- Security contexts hidden, system manipulates numerical SIDs

Access Vector Cache

- Once a policy decision is made the result is stored in the Access Vector Cache (AVC)
- Object Managers store reference to the entry in the cache
- When policy is changed the AVC is flushed
- Also object managers can register callbacks which are invoked on policy change
- Mapped file pages **not** invalidated on policy change

Security Policy

- Role Based Access control
- Type Enforcement
- Multi-Level Security (optional and not discussed here)

Security Context Labels

- Each subject (process) and object (file, socket etc. . .) tagged
- Security Context build from
  - User ID (after initial login — orthogonal to Linux User ID)
  - Role (only for processes)
  - Type (object)/Domain (process)
  - MLS Level/Range (optional)

Logview source code

```
...
setuid(0);
system("grep $USER /var/log/messages")
...
```

Example Policy

- Login Roles:

```
user sjm217 roles { user_r sysadm_r };
```

- File tagging:

```
/var/log(/.*)?
 system_u:object_r:var_log_t

/usr/local/bin/logview
 system_u:object_r:logview_exec_t
```

Example Policy

- Permit use:

```
role user_r types logview_t;
every_domain(logview_t)
```

- Automatic domain transition:

```
domain_auto_trans(user_t, logview_exec_t,
 logview_t
```

Example Policy

- Grant Permissions:

```
allow logview_t var_log_t:file
 r_file_perms

allow logview_t logview_t:capability
 {setuid}

can_exec(logview_t, shell_exec_t)

can_exec(logview_t, bin_t)

...
```

Logview execution (permissive mode)

[sjm217@tinfoil sjm217]$ logview | head -n1
Nov 11 15:02:31 tinfoil su(pam_unix)[19462]: session opened for user root by sjm217(uid=500)

[sjm217@tinfoil sjm217]$ export USER="root /etc/shadow"

[sjm217@tinfoil sjm217]$ ./logiew/logview | head -n1
/etc/shadow:root:1L1IEQjXx$5YY8ybUYoaLIRX/bNv. . .

Logview execution (enforcing mode)

```
[sjm217@tinfoil sjm217]$ logview | head -n1
Nov 11 15:02:31 tinfoil su(pam_unix)[19462]: session
opened for user root by sjm217(uid=500)

[sjm217@tinfoil sjm217]$ export USER="root /etc/shadow"

[sjm217@tinfoil sjm217]$ ./logiew/logview | head -n1
grep: /etc/shadow: Permission denied
/var/log/messages:Nov 11 15:02:31 tinfoil
su(pam_unix)[19462]: session opened for user root by
sjm217(uid=500)
```