Contents

1 Introduction **4**

2 Background **4**

2.1 TrustedBSD Mandatory Access Control Framework 4

2.2 Linux Security Modules Framework 6

2.3 Framework Comparison 7

3 SELinux **10**

3.1

6.7 Network Support 28
6.8 Miscellaneous Module and System Entry Points 28
6.9 Entry Point Comparison 29

7 Future Development **29**

8 Recent Linux Changes **30**

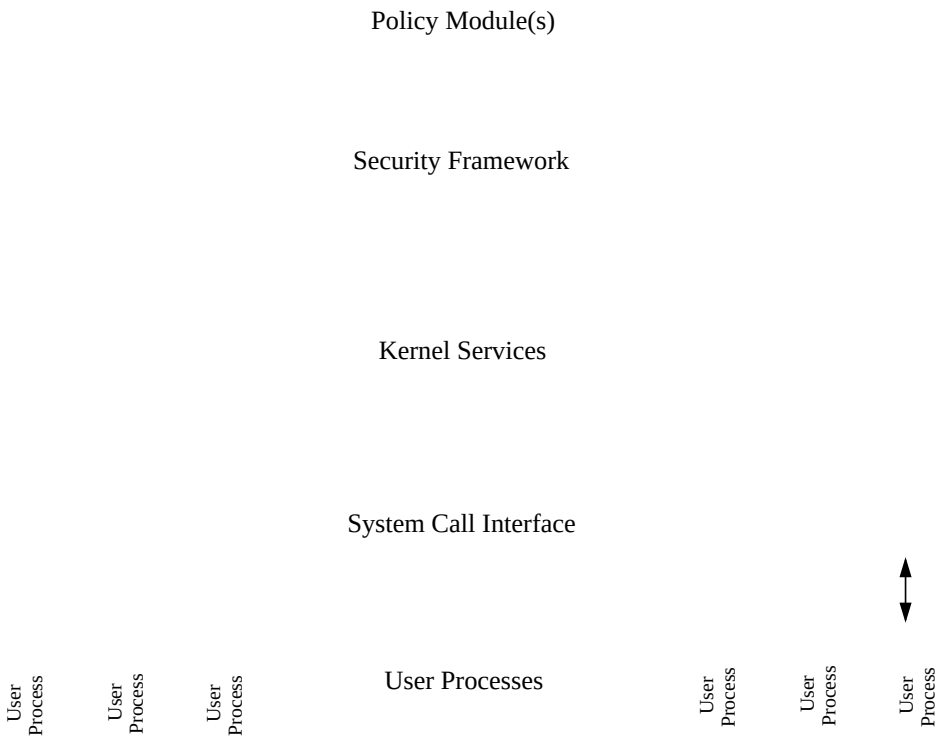
9 Getting the Software **31**

1 Introduction

Network Associates Laboratories has completed an initial port of the Flask security architecture[1] and other components of Security Enhanced Linux (SELinux)[2] to the FreeBSD[3] operating system. This project, called

grams, file systems, pipes, files, sockets, packets, network devices, and System V IPC objects.

SELinux


Policy Module(s)
Security Framework
Kernel Services
System Call Interface
User Processes
User Process
User Process
User Process
User Process
User Process
User Process

hooks and label management tools, but enforces no semantics

to provide a clean separation between policy enforcement and policy interpretation. The Flask security architecture also includes an access vector cache (AVC) component to help minimize the performance overhead from the access control computation. While policy enforcement code is largely system specific,

4 SEBSD

This section provides an overview of the SEBSD security module architecture. Much like the SELinux module, TrustedBSD MAC policies are built as loadable kerne

The MAC Framework is initialized early in the boot process, shortly after basic kernel primitives are initialized (memory allocation, system console, and locking primitives), but prior to probing devices and starting any ker15.es

Program	Description
getfmac	Retrieve file label
setfmac	Set file label
getp	

5.1.1 Process Labels

The `task_security_struct`, defined in `sebsd_labels.h`, contains security information for systePrstra1Td(informT Q q50 T45he)Tj /erne.6254 0014(syste)l Q q501926(for)Tjproc.625

```
struct mount_fs_security_struct {
        security_id_t sid;
};
```

Field	Description
sid	SID for the mount
uses_psids	Flag: this file system supports persistent SIDs
uses_task	Flag: use creating task SID for vnodes
uses_genfs	Flag: use `security_genfs_sid` for vnodes
uses_trans	Flag: whether to call `security_transition_`

```
struct file_security_struct {
        security_id_t sid;
};
```

5.1.5 Network and System V IPC Labels

The SEBSD module does not yet provide labels for network objects, and the TrustedBSD MAC framework does not currently provide labeling or access control entry points for most of the System V IPC subsystem. A prototype implementing access

Entry Point	Description
sebsd_init_cred_label sebsd_init_	

5.3 Internalize/Externalize Operations

In order to translate between kernel object labels and user space textual representations, the TrustedBSD MAC framework provides entry point functions to internalize and externalize process and file labels. SEBSD generates a string representation of labels by composing the module name

to always deny requests.

The changing of process labels at execution time has proven to be sufficiently different from the behavior of other MAC policies, that the TrustedBSD MAC framework needed to be modified to support it.

This execution time lab

these checks. This locking requirement led to the breakup of the relabel operationiren

Hook	Source	Target	Permission
`sebsd_check_vnode_access` `sebsd_check_vnode_open`	credential	file	read, write, search append, or execute
`sebsd_check_vnode_chdir` `sebsd_check_vnode_chroot`	credential	directory	search
`sebsd_check_vnode_create`	credential	directory file filesystem	add name, create file, create directory

6.4 Pipe Entry Points

The SEBSD pipe entry points perform access control for interprocess communication pipes. The permissions checked for pipe objects are similar to those checked for file objects. These permissions are listed in the following table:

Hook	Source	Target	Permission

6.7 Network Support

While the TrustedBSD MAC framework provides labeling and access control for network devices, sockets, and messages, the SEBSD module does not yet implement any of these entry point functions.

6.8 Miscellaneous Module and System Entry Points

These entry point functions are called to register and de-register the the SEBSD module. The following table lists these entry points, along with the generic system call entry point.

Synchronize the AVC and Security Server. Since SEBSD originally branched from the SELinux development tree, the SELinux project has made many changes and improvements; SEBSD needs to make certain all important changes

All calls that return contexts or context arrays now provide automatic allocation of context buffers and context array buffers of the proper size. This simplifies the interface, as user applications no longer must guess an initial size and retry with a larger buffer upon failure. SEBSD implements