Security Module Stacking
Next Steps

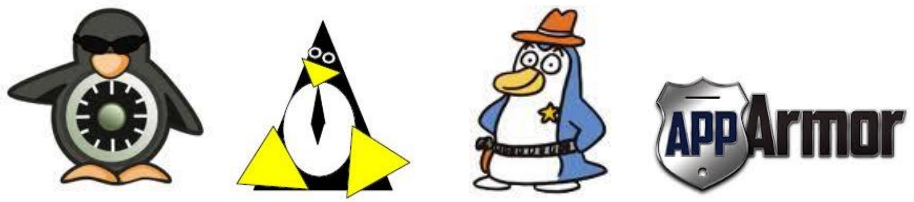
Casey Schaufler

Intel Open Source
Technology Center

The Security Module Stacking Logo

Yama has no logo

LoadPin has no logo



Status And Plans

Stacking Infrastructure in 4.2

First major/minor stacking, then extreme stacking

Linus made a request

Basic handling of multiple modules

Complete generic stacking

Stacking
Extreme Stacking

Stacking as of 4.2

Minor modules

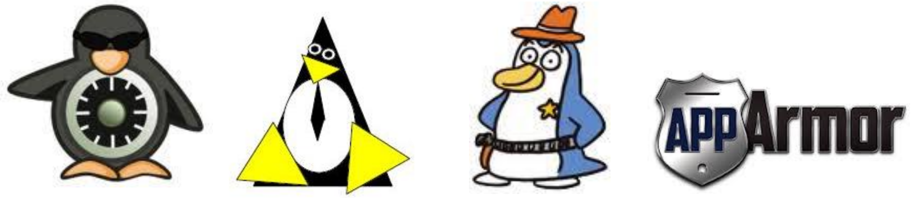
- Don’t use security blobs
- As many as you want
- Fixed order

Yama has no logo

LoadPin has no logo

Major modules

- Use security blobs
- You get one
- Checked last



Stacking as of 4.x

Minor modules

- Don’t use security blobs
- As many as you want

Major modules

- Use security blobs
- You get one
- Improved inode performance

Specified order

Yama has no logo

LoadPin has no logo

Extreme Stacking

All modules treated equally

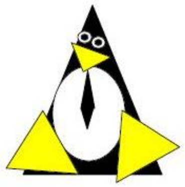
May or may not use security blobs

As many as you want

Specified order

LoadPin has no logo

Yama has no logo



Linus’ Inode Request

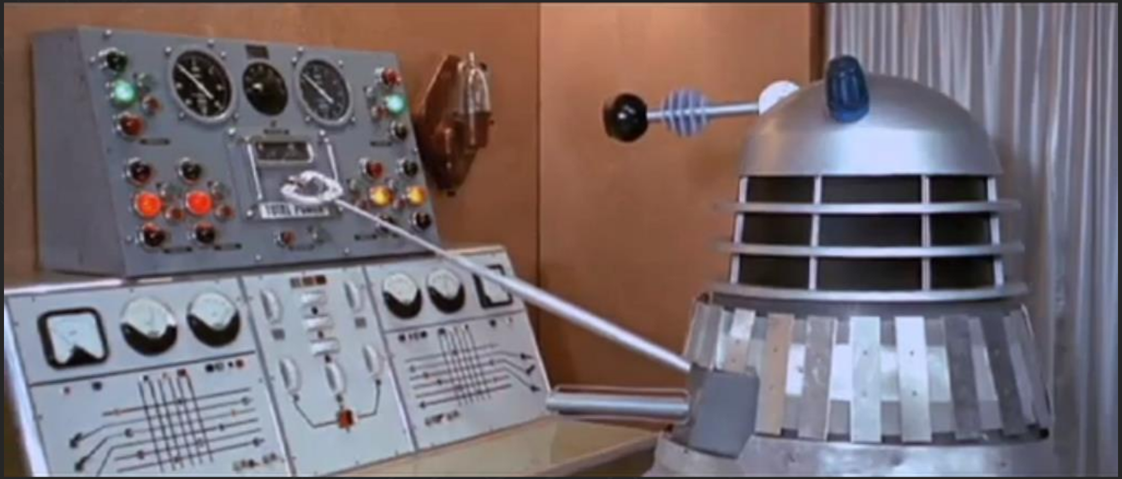
Put The Blob In The Inode

```
struct inode {
     …
     union {
#ifdef CONFIG_SECURITY_SELINUX
          struct inode_selinux     i_selinux;
#endif
#ifdef CONFIG_SECURITY_SMACK
          struct inode_smack       i_smack;
#endif
     };
     …
}
```

Extreme Stacking

```
struct inode {
     …
#ifdef CONFIG_SECURITY_EXTREME_STACKING
     struct {
#else
     union {
#endif
#ifdef CONFIG_SECURITY_SELINUX
          struct inode_selinux     i_selinux;
#endif
#ifdef CONFIG_SECURITY_SMACK
          struct inode_smack       i_smack;
#endif
     };
     …
}
```

Plan B



Intel Software

Intel OpenSource TECHNOLOGY CENTER

Infrastructure Blob Management

Modules tell the infrastructure how much they need

Infrastructure allocates and free blobs

Still a bit of design required

Identifying The Module

Los Angeles, California: Where the term 'police lineup' has come to mean just that.
6
5
PD
THE CHRISTIAN SCIENCE MONITOR
Bennett

Module Selection

Comma separated list of module names

    yama,apparmor

    selinux

Capabilities module is not presented

Order matters

Report

`/sys/kernel/security/lsm`

Module Selection

Boot line option

… `security=yama,smack` …

Kconfig

```
config DEFAULT_SECURITY
string "Ordered list of LSMs to register"
depends on SECURITY
default "(all)"
```

Is this good enough?

Process Attribute Interfaces

Process Attribute Interfaces

```
/proc/…/attr/current

/proc/…/attr/selinux/current
/proc/…/attr/smack/current
/proc/…/attr/apparmor/current

/proc/…/attr/context
```

Security Contexts

`/proc/…/attr/context`

<module="value"/>

    <selinux="jabberwoc_t"/>
    <smack="bandersnatch"/>
    <apparmor="jubjub bird"/>

In libapparmor:

```
    i = sscanf(source, "<apparmor=\"%s\"/>", context);
```

Extreme Security Contexts

`/proc/…/attr/context`

<module="value"/>[<module="value"/>]…

<selinux="jabberwoc_t"/><smack="bandersnatch"/><apparmor="jubjub bird"/>

In libapparmor:

```
    i = sscanf(source, "<apparmor=\"%s\"/>", context);
```

In libselinux:

```
    i = sscanf(source, "<selinux=\"%s\"/>", context);
```

Approaching Extreme Stacking

Security Blobs For Extreme Stacking

```
struct file {
      …
#ifdef CONFIG_SECURITY_EXTREME_STACKING
      struct {
#else
      union {
#endif
#ifdef CONFIG_SECURITY_SELINUX
            struct file_selinux        *f_selinux;
#endif
#ifdef CONFIG_SECURITY_SMACK
            struct file_smack          *f_smack;
#endif
#ifdef CONFIG_SECURITY_APPARMOR
            struct file_apparmor       *f_apparmor;
#endif
      };
      …
}
```

About `secids`

Used in audit

Used in networking

Represent security blobs

Too small for multiple blobs

Cannot be expanded in secmarks

Extreme **`secids`**

Move **`secid`** <-> **`secctx`** mapping

Out of modules

- SELinux
- Smack
- AppArmor

Into the infrastructure

Under **`CONFIG_SECURITY_EXTREME_STACKING`**

Mapping `secid` and `secctx`

Do it the Smack way

```
struct lsm_names {
    struct list_head          list;
    u32                       lsm_secid;
    char                      *lsm_context;
#ifdef CONFIG_NETLABEL
    struct netlbl_lsm_secattr lsm_netlabel;
#endif
}
```

Add It To The Blob

```
struct inode {
	…
#ifdef CONFIG_SECURITY_EXTREME_STACKING
	struct {
		struct lsm_names	*i_names;
#else
	union {
#endif
#ifdef CONFIG_SECURITY_SELINUX
		struct inode_selinux	i_selinux;
#endif
#ifdef CONFIG_SECURITY_SMACK
		struct inode_smack	i_smack;
#endif
	};
	…
}
```

Recalculate As Necessary

Module hooks change their own values

Invalidate the **`lsm_name`** pointer

Triggers recalculation in the infrastructure

Locking done properly, of course

What Remains

Not Addressed

User space changes for extreme contexts

Dynamic module loading and unloading

Blob size optimization

Netlabel reorientation

- Can be made to work, but won’t without SELinux changes

Something else, certainly

Priest Point Park
Olympia, WA, USA