-rw-r----- professional_u:object_r:bsides_t 1 joek cmd 684 Mar 14 14:25 .workshop

A bespoke, artisan, hand crafted, organic, grass-fed SELinux Workshop

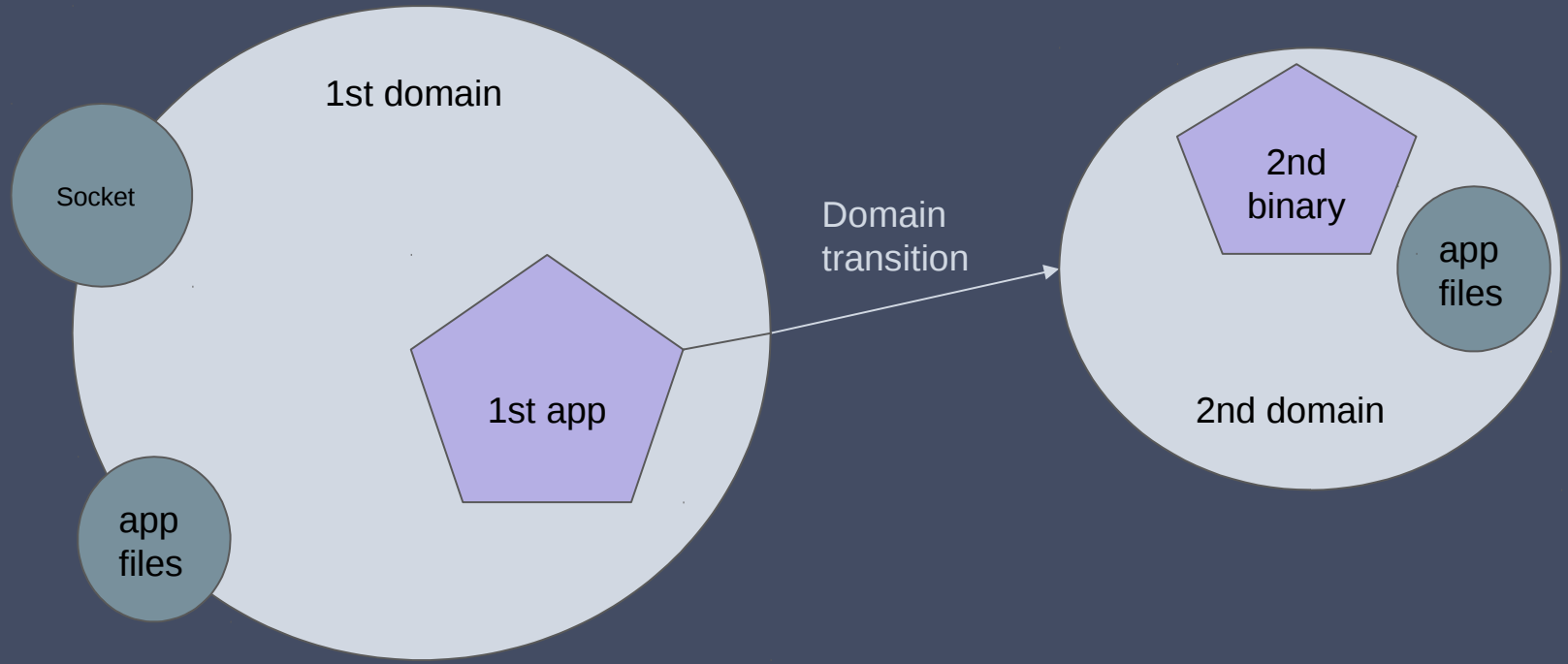
Overview_

- Quick intro to the concepts of SELinux
- Overview of tools and approach for policy creation
- Introduce our vulnerable “test subject”
- Get-stuck-in
- Review
- Discussion of a real-world policy (if time permits)

SELinux concepts_

Mandatory Access Control
Labels
Domain transitions

An example setup_



Key things you need to know about_

Tools

- sestatus
- setenforce/getenforce
- sepolgen
- semodule
- audit2allow/audit2why
- ps -Z / ls -Z
- chkcon/restorecon

There's a lot -
https://github.com/SELinuxProject/selinux/wiki/Tools

Important Locations

- /etc/selinux/config
- /var/log/audit/audit.log
- /usr/share/selinux/

Common things in a SELinux policy_

selinux-policy/

├── policy.fc ----------> list of files your policy targets

├── policy.if ----------> macros

├── policy_selinux.spec ----------> rpm build spec

├── policy.sh ----------> helper script

└── policy.te ----------> the SELinux policy

policy.fc: file_contexts file_

/home/foo(/.*)? --
gen_context(system_u:object_r:web_exec_t,s0)

- # filepath
- # type
- # file label
  - user : role : type
  - Multi-Level Security (MLS) -> # trivia: Bell-LaPadula Model

policy.te: type_enforcement file_

Some points of interest

- # <x>_<y>_<z>(...
- # allow <type x> ...
- # permissive ...

```
policy_module(shellcode-web, 1.0.0)

########################################
#
# Declarations
#

type shellcode-web_t;
type shellcode-web_exec_t;
init_daemon_domain(shellcode-web_t, shellcode-web_exec_t)

permissive shellcode-web_t;

########################################
#
# shellcode-web local policy
#
allow shellcode-web_t self:fifo_file rw_fifo_file_perms;
allow shellcode-web_t self:unix_stream_socket create_stream_socket_perms;

domain_use_interactive_fds(shellcode-web_t)

files_read_etc_files(shellcode-web_t)

miscfiles_read_localization(shellcode-web_t)

sysnet_dns_name_resolve(shellcode-web_t)
```

Learning mode_

```
# sestatus
# getenforce
# setenforce
# <editor-of-choice> /etc/selinux/config
```

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of three two values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

```
# cat /var/log/audit/audit.log | audit2allow -m <mod>
```

This command is so valuable, it will tell you all the violations that were found in your testing and format them for copy-paste into your policy

Type transitions_

type_transition source_type target_type : class default_type;

e.g. *type_transition web_exec_t web_bin_t:file web_t;*

$ ./shellcode_eater 4831c004990f05

This application is super vulnerable, it might also be crap code too. But, if you feed it assembler hex it will exec it 😱

We’re going to try to limit:

- # bad things it can do
- # badnesses that can be done unto it

```c
void main(int argc, char **argv) {
    if (argc != 2) {
        printf("Please provide shellcode input\n");
        printf("example usage:\n./shellcode-eater \"cccccc\"\n");
        exit(1);
    }

    char *data = argv[1];
    size_t datalen = strlen(data);
    char *code = malloc(datalen / 2);

    int j = 0;
    char tmp[3] = {0};
    for (int i = 0; i < datalen; i += 2) {
        tmp[0] = data[i];
        tmp[1] = data[i + 1];
        long number = strtol(tmp, NULL, 16);
        code[j] = (char)number;
        j++;
    }

    int (*func)();
    func = (int (*)())code;
    (int)(*func)();

    free(code);
}
```

WEB 2.0_

I’ve created a website for us to try and protect with SELinux based on the previous tool

Lets get going_

Vagrant/VBox (*recommended*):

https://github.com/cmdinc/selinux-workshop/tree/master/vagrant

Painpoints: labelling_

You can go very wrong if you don’t have a plan for what files should be in which distinct contexts/domains.

Spend time upfront thinking about that so you don’t fall into a trap!

Painpoints: relabelling_

Things to try when relabelling doesn’t work!

try `chcon -v <user>:<object>:<type <yourfile>` and see what it says
  - If that works, then you have an issue with your `policy.fc` file labels
  - When your policy works `restorecon -F -vv <yourfile>` should change the labels
Always have a ` tail -n1 -f /var/log/audit/audit.log` running just in case
To debug whether your file-contexts are even on the system try:
  - `semanage fcontext -l | grep <my labels>`
Double check your helper script `policy.sh` has included restorecon for the newly updated fc files, it’s a common mistake I made

Painpoints: relabelling 2.0_

SELinux doesn’t execute regex matches by most specific 😒
Therefore you may need to go hunting for what is messing with your restore, or just create your directory somewhere that doesn’t conflict

Advanced: macros_

These help when:

- You have repetitive things applying to many domains
- Want to share/reuse some complex config across SELinux policies

The GNU M4 macro syntax is weird IMO

** Just note, ` == push, ‘ == pop*

```
50 ########################################
51 ## <summary>
52 ##  Allow type_t to teardown shm and sockets
53 ## </summary>
54 ## <param name="type">
55 ## <summary>
56 ## This macro allows things like shutdown_t access to close the cmd_daemon sock
57 ## </summary>
58 ## </param>
59 #
60 interface(`cmd_daemon_t_shm_destroy',`
61     gen_require(`
62         type cmd_daemon_t;
63         class shm { destroy read unix_read unix_write write };
64         class unix_stream_socket connectto;
65     ')
66
67     allow $1 cmd_daemon_t:shm { destroy read unix_read unix_write write };
68     allow $1 cmd_daemon_t:unix_stream_socket connectto;
69 ')
```

Thanks for your time

Slides + webapp + shellcode_eater + the policy Can be found at:

https://github.com/cmdinc/selinux-workshop

I also plan to add more updates to that repo with some current work I’m doing on “how to integration test SELinux policies”