Report 2

Securing the Apache Server with SElinux

Now that we have the basic understanding of SElinux , security contexts and type enforcement , we try to increase the security of the apache web server with SElinux.

Key tools and technologies discussed in this report include SElinux , Apache web server , Mandatory access control (MAC), getenforce , getsebool and setsebool.We will try to analyse how the SELinux booleans can be used to on or off the specific features of SELinux .

Note : All the experiments were done using centOS 5.5 VM (set up by Oracle VM virtual box & host OS is Ubuntu 11.04)

Security-Enhanced Linux Overview

Security-Enhanced Linux (SElinux) is now an inherent component of the linux operating system. It was developed by the United States National Security Agency for the purpose of Operating System security. Basically , Selinux helps in establishing Mandatory access control (MAC) policies. SElinux has predefined boolean variables that make it easy to use. We use these SElinux boolean variables to secure the apache http daemon.

Installing and running httpd

The apache HTTP server project is an effort to develop and maintain an open-source HTTP server for modern Operating Systems.The goal of this project is to provide a secure,efficient and extensible server that provides HTTP services in sync with the current HTTP standards. Still, there are certain security loopholes in apache webserver (which we analyze and try to remove with SElinux) .

The following commands install and start the http daemon :
[root@localhost ~]# yum install httpd
[root@localhost ~]# service httpd start

We use Selinux policy.21 (policy.VersionNumber) for the experiments. This policy defines the security context for the http daemon as httpd_t. This context can be checked by the following command.

[root@localhost selinux]# ps axZ | grep httpd

root:system_r:httpd_t 16687 ? Ss 0:01 /usr/sbin/httpd
root:system_r:httpd_t 16689 ? S 0:00 /usr/sbin/httpd
root:system_r:httpd_t 16690 ? S 0:00 /usr/sbin/httpd

root:system_r:httpd_t 16691 ? S 0:00 /usr/sbin/httpd
root:system_r:httpd_t 16692 ? S 0:00 /usr/sbin/httpd
root:system_r:httpd_t 16693 ? S 0:00 /usr/sbin/httpd
root:system_r:httpd_t 16694 ? S 0:00 /usr/sbin/httpd
root:system_r:httpd_t 16695 ? S 0:00 /usr/sbin/httpd
root:system_r:httpd_t 16696 ? S 0:00 /usr/sbin/httpd

The Z option to the ps command shows the security context of the httpd daemon as root:system_r:httpd_t.

SElinux policy also defines several **file security context types** to be used with the http daemon. Some of them are :

1) *httpd_sys_content_t* - only files with of this context type can be accessed by the httpd daemon. So files which are available for all httpd scripts and the daemon to access need to have this context type. If a file has type other than httpd_sys_content_t than that file cannot be accessed by the httpd daemon.

eg :- To prevent httpd daemon to access files from user's home directory , the files in the home directory are labelled with the type user_home_t and thus, cannot be accessed by the apache server.

The default http directory is /var/www/ (This default location can be changed by changing the line DocumentPath /givefullpath in the file /etc/httpd/conf/httpd.conf file). Its security context is :-

[root@localhost selinux]# ls -Z /var/www/ | grep html
drwxr-xr-x root root system_u:object_r:httpd_sys_content_t html

This directory is assigned the type httpd_sys_content_t as a part of its security context which allows the http daemon to access the files stored in this directory. Any file or subdirectory inherits the security context of the directory in which it is created.

We tried to access a file created in the user's home directory having labelled with the type user_home_t and then moved to /var/www directory. SElinux prevented the access and gave an error that the httpd can only access files of the type httpd_sys_content and no other files. SELinux generates error messages in /var/log/httpd/error_log and /var/log/messages.

[root@localhost akhil]# pwd
/home/akhil
[root@localhost akhil]# touch index.html
[root@localhost akhil]# mv index.html /var/www/html/
[root@localhost akhil]# wget localhost/index.html
--2011-03-10 06:41:17-- http://localhost/index.html
Resolving localhost... 127.0.0.1

Connecting to localhost|127.0.0.1|:80... connected.
HTTP request sent, awaiting response... 403 Forbidden
2011-03-10 06:41:17 ERROR 403: Forbidden.

More informative SElinux error message : (obtained by using the selinux troubleshoot browser)

SELinux has denied httpd access to potentially mislabeled file(s) (/var/www/html/index.html). This means that SELinux will not allow httpd to use these files. It is common for users to edit files in their home directory or tmp directories and then move (mv) them to system directories. The problem is that the files end up with the wrong file context which confined applications are not allowed to access.
If you want httpd to access this files, you need to relabel them using :
restorecon -v '/var/www/html/index.html'.

2)httpd_sys_script_exec_t : All cgi scripts which need to access files with sys types must have this type as their file security context.

3)httpd_sys_script_ro_t : Set files with this context type if you want httpd_sys_script_exec_t scripts to read the data , and disallow other sys scripts from access.

4)httpd_sys_script_[rw/ra]_t : Set files with this context if you want httpd_sys_script_exec_t scripts to [read/write ; read/append] the data , and disallow other non sys scripts from access.
5)httpd_unconfined_script_exec_t : Set cgi scripts with this context type to allow them to run without any SELinux protection.

SELinux Booleans and HTTPD

SELinux has a set of built-in switches named Booleans or conditional policies that you can use to turn specific SELinux features on or off . SELinux httpd policy is extremely flexible and has several booleans that allow you to manipulate the policy and run httpd with the tightest access possible.

Entering the getsebool -a | grep http command lists the 23 Booleans related to the http daemon. These booleans allow you to customize SELinux policy for the httpd deamon at runtime, without any knowledge of SELinux policy writing and without compiling, or loading a new policy. This allows changes, such as allowing services access to NFS file systems, without reloading or recompiling SELinux policy.

There are three command line tools for working with Booleans:
1. getsebool -a : returns the current state of all the SELinux Booleans defined by the policy.
(-a can be replaced by any specific boolean variable)

[root@localhost akhil]# getsebool httpd_enable_cgi

httpd_enable_cgi --> on

2. setsebool : sets the current state of a particular SELinux boolean or a list of booleans to 1/true (enable it) or 0/false (disable it).

[root@localhost akhil]#setsebool httpd_enable_cgi 0

3.togglesebool : It flips the current value of one or more Booleans.

By , default these 6 booleans are switched on as far as SELinux httpd policy is concerned-

[root@localhost akhil]# getsebool -a | grep http | grep "\-\-> on"
httpd_builtin_scripting --> on
httpd_can_sendmail --> on
httpd_enable_cgi --> on
httpd_enable_homedirs --> on
httpd_tty_comm --> on
httpd_unified --> on

Now we will study the effect of these boolean variables on the security of the apache server. We will see how we can manipulate these variables to achieve a secure apache server.

1) httpd_enable_cgi: If your content does not use the Common Gateway Interface (CGI) protocol, set this Boolean to off. If you are unsure about using CGI in the Web server, try setting it to off . Many attacks to apache server are done via CGI scripts. SELinux can help disable CGI scripting to prevent such attacks.

Lets see an eg:

i) [root@localhost ~]# service httpd restart
Stopping httpd: [OK]
Starting httpd: [OK]

ii) Now we copy a test cgi script into the localhost's cgi-bin directory
[root@localhost ~]# cd /var/www/cgi-bin/
[root@localhost cgi-bin]# cp /usr/lib/perl5/5.8.8/CGI/eg/tryit.cgi ./

iii) Checking the value of the boolean variable before accessing the script via the apache browser.
[root@localhost cgi-bin]# getsebool httpd_enable_cgi
httpd_enable_cgi --> off

The value if **off.** This means that cgi-scripting is disabled by SELinux.

iv) When we open the browser and try to access *http://localhost/cgi-bin/tryit.cgi* we get the following AVC denial by SELinux:

host=localhost.localdomain type=AVC msg=audit(1299758748.377:236): avc: denied { getattr } for pid=23467 comm="httpd" path="/var/www/cgi-bin" dev=dm-0 ino=1343578 scontext=root:system_r:httpd_t:s0 tcontext=system_u:object_r:httpd_sys_script_exec_t:s0 tclass=dir

We use SELinux troubleshooter to get a more detailed error message :

SELinux has denied the http daemon from executing a cgi script. httpd can be setup in a locked down mode where cgi scripts are not allowed to be executed. If the httpd server has been setup to not execute cgi scripts, this could signal a intrusion attempt.

Allowing access:

If we want httpd to be able to run cgi scripts,we need to turn on the httpd_enable_cgi boolean:

The following command will allow this access:
setsebool -P httpd_enable_cgi=1

Now we can run our cgi-script without error.

The next boolean we study is httpd_enable_homedirs.

2) httpd_enable_homedirs : When this Boolean is set to on, it allows httpd to read content from subdirectories located under user home directories. If the Web server is not configured to serve content from user home directories, set this Boolean to off.

So , if the files under the home directory and sensitive and we don't want apache to have access to these files we can set this variable to off. If apache gets comprimised and this variable is on , then the attacker can access files under user's home directories.

Let see an eg:

1. Create /etc/httpd/conf.d/userdir.conf file

2. Add something like following content to file:

<**IfModule mod_userdir.c**>
UserDir enabled testuser
UserDir public_html

```
</IfModule>
<Directory /home/*/public_html>
    Options Indexes Includes FollowSymLinks
    AllowOverride All
    Allow from all
    Order deny,allow
</Directory>
```

3. Start/Restart Apache (httpd)

/etc/init.d/httpd start

4. Create public_html directory/directories .

#mkdir /home/testuser/public_html

5. Change the correct permission to home and public_html directories .
#chmod 711 /home/testuser
#chown testuser:testuser /home/testuser/public_html
#chmod 755 /home/testuser/public_html

6. Change the type of the public_html directory .
#chcon -R -t httpd_sys_content_t **/**home**/**testuser**/**public_html

7. Now on the httpd_enable_homedirs boolean .
setsebool -P httpd_enable_homedirs=1

8. #wget http://localhost/~testuser

We note that we are able to access content.

9. Now off the httpd_enable_homedirs boolean .
#setsebool -P httpd_enable_homedirs=0

10. #wget http://localhost/~testuser

SElinux now prevents us from accessing any files.

3) httpd_can_sendmail : If the apache server does not use sendmail , we should turn off this boolean variable . Many attackers can use the sendmail service to launch spamming attacks. Turning this variable off can prevent unauthorized users from sending spams from the server.

NOTE : We also studied the effects of other variables on the security of apache.

We are reporting experiments only on 2 of them due to lack of space.We will cover more variables while giving the demo.

Httpd & SELinux - configuring ports

http_port_t *:* When /etc/httpd/conf/httpd.conf is configured so httpd listens on a port other than TCP ports 80, 443, 488, 8008, 8009, or 8443, the semanage port command must be used to add the new port number to SELinux policy configuration.

The following example demonstrates configuring httpd to listen on a port that is not defined in SELinux policy configuration for httpd, and, as a consequence, httpd failing to start. This example also demonstrates how to then configure the SELinux system to allow httpd to successfully listen on a non-standard port that is not already defined in the policy.

STEPS

1.Stop the httpd service : # service httpd stop

2.Run semanage port -l | grep -w http_port_t to view the ports SELinux allows httpd to listen on:

```
# semanage port -l | grep -w http_port_t
   http_port_t                tcp      80, 443, 488, 8008, 8009, 8443
```

3.Edit /etc/httpd/conf/httpd.conf as the root user. Configure the Listen option so it lists a port that is not configured in SELinux policy configuration for httpd. In this example, httpd is configured to listen on port 12345:

```
#Listen 12.34.56.78:80
Listen 127.0.0.1:12345
```

4.Run service httpd start to start httpd:

```
# service httpd start
Starting httpd: (13)Permission denied: make_sock: could not bind to address
127.0.0.1:12345
no listening sockets available, shutting down
Unable to open logs                                          [FAILED]
```

5. Now use the setroubleshooter to view the error :

ERROR: setroubleshoot: SELinux is preventing the httpd (httpd_t) from binding to port 12345. For complete SELinux messages. run sealert -l f18bca99-db64-4c16-9719-1db89f0d8c77

6.For SELinux to allow httpd to listen on port 12345, as used in this example, the following command is required:

```
# semanage port -a -t http_port_t -p tcp 12345
```

7.Try to start the httpd service again

```
# service httpd start
```

Starting httpd: [OK]