CS5231 – Systems Security

Homework 3: System Auditing

Due date and time: 23:59pm, November 13, 2022. This homework MUST be finished independently.

1. Introduction

In this homework, you will be exploring the Linux Auditing System and Linux Security Modules (LSM). You will be given a program called `malicious_prog`, which would repeatedly do the following operations on each file under the directory `/usr/include/linux`: open the file, copy the contents of the file, and save the copied contents into a new file (with a different name) under the specified destination directory. After the operations on each file, the program pauses for 100 ms.

We have created a VM installed with Ubuntu 20.04 (x86_64), which contains the required material for this homework under the directory `~/A3`. The VM image can be downloaded from the following link:

https://nusu-my.sharepoint.com/:u:/g/personal/gao_nus_edu_sg/ETKYjl2-kWFAj8vEs-VlonABlG9t4s3Jpd6FgFCGdrJEjg?e=mfYGGb

The password for the user account `student` is `student`.

The `malicious_prog` is located at the directory `~/A3` in the provided VM. You can run the malicious program using the following command:

```
cd ~/A3
mkdir output
malicious_prog ./output
```

To inspect the system calls that are invoked by `malicious_prog`, you can use the `strace` tool.

```
strace malicious_prog ./output
```

In this homework, you are tasked to intercept, collect, and mediate the system calls of the program `malicious_prog`.

2. Task 1 (14 marks) Audit Log Collection and Analysis

In this task, you are asked to use the system service `auditbeat` to intercept, collect, and parse audit logs of certain system calls invoked by the malicious program `malicious_prog`. The execution of `malicious_prog` will invoke many system calls, e.g., `mmap`, `read`, `fstat`, `close`, `write`. *You need to configure the*

`auditbeat` service to monitor and log the following system calls: `open, openat, read, write, writev.`

2.1 (8 marks) Configuring auditbeat and Generating Audit Logs

Configuring auditdeat. First, you need to properly configure the audit subsystem. To configure the rules for system auditing, you are required to modify the following file: `/etc/auditbeat/audit.rules.d/audit-rules.conf`

This link provides a sample configuration:
https://github.com/Neo23x0/auditd/blob/master/audit.rules
Please refer to the following link on how to define auditing rules: Defining Audit Rules.

Below are useful links for your reference.

1) https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/security_guide/chap-system_auditing
2) https://www.elastic.co/guide/en/beats/auditbeat/current/index.html

Generating Audit Logs. After configuring auditbeat, you can follow below steps to generate the audit logs.

S1) Start Auditbeat. After configuring auditbeat, you can use the following command to start the service:

```
sudo service auditbeat start
```

S2) Run the Malicious Program for 5 Minutes. After the auditbeat service is started, you are required to run the malicious program for 5 minutes.

```
timeout 5m ./malicious_prog ./output
```

S3) Collect Audit Logs. When the auditbeat service is stopped, audit logs are generated in JSON format under the directory /var/log/auditbeat/. You can copy the log file into another directory.

```
sudo service auditbeat stop
```

Remarks. Note that each log entry represents a system call event. Specifically, all the original system call parameters and the return value are contained under the field "auditd", represented under subfields "a0", "a1", …, "exit". If the system call is related to a file open event (e.g., open, openat), a field "file" will indicate which file it operates on. Also note that audit logs of `read` and `write` only contain system call arguments and the return value, while audit logs of `open` will contain the file they operate on. To parse the `read` and `write` system calls, you should get familiar with the meaning of all arguments and return value of every system call.

Parsing Audit Logs. *After the audit logs are collected, you are required to parse these logs to generate an aggregated log file named* **task1_parsed.log**. You can use any

programming language you prefer. Specifically, each log entry should of the following format: <timestamp, subject, operation, object>.

The raw audit logs are verbose and contain redundant information. For each entry in an audit log, you need to parse it to a tuple: <timestamp, subject, operation, object>. Here, each operation is a system call. The subject should always be the `malicious_prog`, and the object should be the file being operated on. For example, the tuple

```
<00:00:00, malicious_prog, write, 1.txt>
```

means that `malicious_prog` writes the file `1.txt`.

2.2 (6 marks) Analyzing Audit Logs

After the audit logs are parsed to an aggregated log file, you are asked to do the following analysis using a program (use any programming language you prefer): printing out the top ten most accessed files under the directory `/usr/include/linux`; for files that have the same number of accesses, select the files alphabetically by their file names.

3. Task 2 (6 marks) System Mediation using LSM

In this task, you are asked to implement file-level access control using Linux Security Modules (LSM). The LSM framework works by introducing hooks into a wide variety of kernel functionalities. Generally, the access control checks occur in the following order: after performing error checking, the kernel consults the discretionary access control mechanism, then calls the hooks for the minor modules if any are present, followed by the hooks of the major security module in place at the time. The full list of the hooks exposed to security modules, along with their description is located in `include/linux/lsm_hooks.h` in the source code of Linux kernel, which can also be access this link:
https://elixir.bootlin.com/linux/v6.0.2/source/include/linux/lsm_hooks.h

We have provided a properly configured Linux kernel source code under the directory `~/A3/linux-hwe-5.15.60`. Specifically, we have added a skeleton LSM called "`cs5231`" to the kernel source code. You only need to work on the skeleton LSM we provided, which is located at the following directory: `~/A3/linux-hwe-5.15.60/security/cs5231`. Please try to grasp the provided source code first before proceeding to write your own code.

During its execution, `malicious_prog` will read many files under the directory `/usr/include/linux`, and two of them are: `if.h` and `un.h`. In this task, we will mark these two header files as *sensitive* files. *You goal is to use the `cs5231` LSM to prevent `malicious_prog` from accessing these two sensitive files, and log it accordingly if such access is detected*. Such logging can be achieved by, for example,

printing a log *"Sensitive file {/path/to/file} access is detected."* (via pr_info()) into the kernel buffer.

(**4 marks**) **Your Task**. Write the pseudocode to achieve the above goals in LSM hook functions. We have provided two skeleton hook functions: `cs5231_bprm_creds_for_exec`, and `cs5231_file_permission` for your reference.

(2 marks) In the two skeleton hook functions: `cs5231_bprm_creds_for_exec`, and `cs5231_file_permission`. Implement the pseudocode in C and compile the kernel. (This 2-mark task can be submitted after the homework submission, by November 30, 2022).

You need to fill in the required code into these two hook functions, as you did in the loadable kernel module task in homework 1.

Instructions on Testing Your LSM. You can follow the following steps to verify that your LSM is loaded properly.

S1) Compile and Install the Linux Kernel. After finishing your LSM code for this task, you need to compile the Linux kernel source code so that your LSM is part of the kernel. To compile the kernel source code, you can use the following commands:

```
cd ~/A3/linux-hwe-5.15.60/
make -j`nproc` bindeb-pkg
```

This could take quite a while. When the compilation is done, you will get a .deb package called `linux-image-5.15.60_5.15.60-1_amd64.deb` under the directory `~/A3`.

To install the new Linux kernel, you can use the following command:

```
sudo dpkg -i ~/A3/linux-image-5.15.60_5.15.60-1_amd64.deb
sudo update-grub
```

S2) Reboot the VM to use the new Linux Kernel. We have properly configured the bootloader (i.e., GRUB) to use the newly installed Linux kernel. You can use the following command to verify that the new Linux kernel has loaded the `cs5231` LSM:

```
dmesg | grep cs5231
```

Below screenshot shows that the `cs5231` LSM was successfully loaded.

```
student@student-VirtualBox:~$ dmesg | grep cs5231
[    0.066609] cs5231_lsm_hook: cs5231 LSM initializing..
[    0.432973] cs5231_lsm_hook: Loaded bprm: /sbin/modprobe
[    0.434140] cs5231_lsm_hook: Loaded bprm: /sbin/modprobe
[    0.435661] cs5231_lsm_hook: Loaded bprm: /sbin/modprobe
[    0.436609] cs5231_lsm_hook: Loaded bprm: /sbin/modprobe
[    0.442414] cs5231_lsm_hook: Loaded bprm: /sbin/modprobe
[    0.443317] cs5231_lsm_hook: Loaded bprm: /sbin/modprobe
[    0.684217] cs5231_lsm_hook: Loaded bprm: /init
[    0.685614] cs5231_lsm_hook: Loaded bprm: /usr/bin/mount
[    0.686495] cs5231_lsm_hook: Loaded bprm: /usr/bin/mount
[    5.291062] cs5231_bprm_creds_for_exec: 553 callbacks suppressed
[    5.291070] cs5231_lsm_hook: Loaded bprm: /bin/false
[    5.300837] cs5231_lsm_hook: Loaded bprm: /bin/false
[    5.310647] cs5231_lsm_hook: Loaded bprm: /bin/false
[    5.319559] cs5231_lsm_hook: Loaded bprm: /bin/false
[    5.340993] cs5231_lsm_hook: Loaded bprm: /usr/libexec/at-spi-bus-launcher
[    5.373294] cs5231_lsm_hook: Loaded bprm: /usr/bin/dbus-daemon
[    5.500489] cs5231_lsm_hook: Loaded bprm: /usr/bin/Xwayland
[    5.936423] cs5231_lsm_hook: Loaded bprm: /usr/bin/ibus-daemon
[    5.950727] cs5231_lsm_hook: Loaded bprm: /usr/libexec/ibus-dconf
[    5.952606] cs5231_lsm_hook: Loaded bprm: /usr/libexec/ibus-portal
[   22.036141] cs5231_bprm_creds_for_exec: 76 callbacks suppressed
[   22.036144] cs5231_lsm_hook: Loaded bprm: /usr/sbin/sshd
[   25.019224] cs5231_lsm_hook: Loaded bprm: /lib/systemd/systemd-user-runtime-dir
[   25.056823] cs5231_lsm_hook: Loaded bprm: /lib/systemd/systemd
[   25.063264] cs5231_lsm_hook: Loaded bprm: /usr/lib/systemd/user-environment-generators/30-systemd-environment-d-generator
[   25.114743] cs5231_lsm_hook: Loaded bprm: /bin/systemctl
[   25.119623] cs5231_lsm_hook: Loaded bprm: /usr/libexec/tracker-extract
[   25.120573] cs5231_lsm_hook: Loaded bprm: /usr/libexec/tracker-miner-fs
[   25.127971] cs5231_lsm_hook: Loaded bprm: /usr/bin/dbus-daemon
[   25.132546] cs5231_lsm_hook: Loaded bprm: /bin/sh
[   25.133136] cs5231_lsm_hook: Loaded bprm: /usr/bin/env
[   34.833126] cs5231_bprm_creds_for_exec: 82 callbacks suppressed
[   34.833134] cs5231_lsm_hook: Loaded bprm: /usr/bin/dmesg
[   34.833154] cs5231_lsm_hook: Loaded bprm: /usr/bin/grep
student@student-VirtualBox:~$
```

4. Submission

You are required to write a report for this homework. For task 1, your report should include **your configuration** for `auditd` (only include the lines that you added); you also need to submit the parsed audit log named **task1_parsed.log**. For task 2, your report should include the **pseudocode** added by you, as well as a brief explanation on them.

For the 2-mark task of in-kernel implementation and compilation, you also need to submit the following files: the C file with your code and **~/A3/linux-image-5.15.60_5.15.60-1_amd64.deb**.

Your submission into Canvas is a single .zip file. Please name this file with the following format: ***your_matric_id*-hw3.zip**, e.g. A19930314-hw3.zip. The .zip file should include **your report**, **task1_parsed.log**. If you submit the compiled kernel, also include the C file and **linux-image-5.15.60_5.15.60-1_amd64.deb** into the zip. If you submit the compiled kernel later, you can zip the C file and the deb file into ***your_matric_id*-hw3-kernel.zip**, and submit them.

Note that we only accept the following formats for compressed file: .zip, .tar.gz, .tar.bz, and .tar.bz2 . Also, please make sure your report does not **EXCEED 3 pages**, excluding references and appendices.

For any questions, please contact the teaching assistants via cs5231ta@googlegroups.com. Hope you can have fun in exploring the basics in systems security. This homework also helps you to choose your final project topic.