Kenny Cao
Tiffany He
Jin Zhen
Alvin Lin

CSE 331 - Antivirus for Linux

Link to GitHub Repository:

https://github.com/caokenny/CSE331-Antivirus

Requirements:

- Ubuntu 16.04 (**strictly** 32-bit version only)
- VirtualBox
- Python3

Setting Up the Database:

Our whitelist and virus databases are stored in their own respective text file. To securely store our database files (to prevent unauthorized agents from accessing or tampering with the database), we are utilizing GitHub's secure web-based hosting services. This method of storage would require the user to authenticate himself/herself to GitHub before receiving authorization to access the database files.

The whitelist database contains the SHA1 hashes of standard linux utilities. The viruses database contains the signatures of "fake" malicious files that we created. Signatures are partial binaries of malicious files that are used by antivirus programs for virus detection.

For creation of the whitelist database, we wrote a python script that traversed through the directories /bin and /usr/bin and generated the SHA1 hashes of all the files in these directories. The SHA1 hashes are stored in a plaintext file called whitelist.txt.

For creation of the virus database, we manually created a couple of "fake" malicious files and extracted their partial binaries. The partial binaries are stored in a plaintext file called viruses.txt.

Updating the Database:

The goal of *Updating the Database* is to fetch the changes that were made to the whitelist and virus database over the internet. It is good practice to always update the database before performing any file/directory scanning operations.

Instructions for Updating the Database:

1. Make sure the antivirus program is in your current directory.
2. Make sure the antivirus program is executable. If it is not, add permissions.
    a) Run the following command in your terminal:

    chmod u+x antivirus
3. Update the whitelist and virus database.

    Run the following command in your terminal:

    ./antivirus -u || ./antivirus --update

By following these instructions, our antivirus program will get the most recently modified version of the whitelist and virus database from a server hosted on Github. Any file that is satisfied by the whitelist database will be not have to be scanned against the virus database.

On-Demand Scanning:

The goal of *On-Demand Scanning* is to allow the user to run our program (our program is named “antivirus”) to scan a file or a directory for viruses.

Instructions for On-Demand Scanning:

1. Make sure the antivirus program is in your current directory.

```
tiffany@tiffany-VirtualBox:~$ cd Desktop/CSE331-Antivirus-master
tiffany@tiffany-VirtualBox:~/Desktop/CSE331-Antivirus-master$ ls
antivirus
```

2. Make sure the antivirus program is executable. If it is not, add permissions.
    a) Run the following command in your terminal:

    chmod u+x antivirus

```
tiffany@tiffany-VirtualBox:~/Desktop/CSE331-Antivirus-master$ ls -l
total 4
-rw-rw-r-- 1 tiffany tiffany 3413 Dec  2 15:26 antivirus
tiffany@tiffany-VirtualBox:~/Desktop/CSE331-Antivirus-master$ chmod u+x antivirus
tiffany@tiffany-VirtualBox:~/Desktop/CSE331-Antivirus-master$ ls -l
total 4
-rwxrw-r-- 1 tiffany tiffany 3413 Dec  2 15:26 antivirus
```

3. Make sure the whitelist and viruses database is updated.
    a) Run the following command in your terminal:

    ./antivirus -u || **./antivirus --update**

```
tiffany@tiffany-VirtualBox:~/Desktop/CSE331-Antivirus-master$ ./antivirus -u
```

4. Run the antivirus program to scan a file or directory. You can provide the absolute or relative path of the file or directory. The antivirus will scan all files/directories for the provided file or directory, which includes hidden files/directories.

a) Tree structure of the directory we are testing with:

```
tiffany@tiffany-VirtualBox:~/Desktop$ tree -a dir
dir
├── .hiddensubdir
│   └── thisisavirus
├── notavirus
├── subdir
│   └── thisisaworm
└── .thisisatrojan

2 directories, 4 files
```

b) Run the following command in your terminal:

./antivirus -f FILE || **./antivirus --file FILE**

Note: **FILE** is the relative/absolute path of the file or directory you want to scan

```
tiffany@tiffany-VirtualBox:~/Desktop/CSE331-Antivirus-master$ ./antivirus -f ~/Desktop/dir
Virus detected: /home/tiffany/Desktop/dir/.hiddensubdir/thisisavirus.infected
Virus detected: /home/tiffany/Desktop/dir/subdir/thisisaworm.infected
Virus detected: /home/tiffany/Desktop/dir/.thisisatrojan.infected
```

5. If the provided file or any file in your provided directory is detected as a virus,

   - A message will be displayed to alert the user in the terminal that the file is infected. The alert message follows the following format:

     "Virus detected: /abs/path/of/file".

```
tiffany@tiffany-VirtualBox:~/Desktop/CSE331-Antivirus-master$ ./antivirus -f ~/Desktop/dir
Virus detected: /home/tiffany/Desktop/dir/.hiddensubdir/thisisavirus.infected
Virus detected: /home/tiffany/Desktop/dir/subdir/thisisaworm.infected
Virus detected: /home/tiffany/Desktop/dir/.thisisatrojan.infected
```

   - All permissions for the infected file is removed.

```
tiffany@tiffany-VirtualBox:~/Desktop/dir$ ls -al
total 20
drwxrwxr-x 4 tiffany tiffany 4096 Dec  2 20:24 .
drwxr-xr-x 5 tiffany tiffany 4096 Dec  2 19:55 ..
drwxrwxr-x 2 tiffany tiffany 4096 Dec  2 20:24 .hiddensubdir
-rw-rw-r-- 1 tiffany tiffany    0 Dec  2 19:59 notavirus
drwxrwxr-x 2 tiffany tiffany 4096 Dec  2 20:24 subdir
---------- 1 tiffany tiffany   36 Dec  2 20:17 .thisisatrojan.infected
tiffany@tiffany-VirtualBox:~/Desktop/dir$ cd .hiddensubdir
tiffany@tiffany-VirtualBox:~/Desktop/dir/.hiddensubdir$ ls -al
total 12
drwxrwxr-x 2 tiffany tiffany 4096 Dec  2 20:24 .
drwxrwxr-x 4 tiffany tiffany 4096 Dec  2 20:24 ..
---------- 1 tiffany tiffany   39 Dec  2 19:57 thisisavirus.infected
```

```
tiffany@tiffany-VirtualBox:~/Desktop/dir/subdir$ ls -al
total 12
drwxrwxr-x 2 tiffany tiffany 4096 Dec  2 20:24 .
drwxrwxr-x 4 tiffany tiffany 4096 Dec  2 20:24 ..
---------- 1 tiffany tiffany   34 Dec  2 19:58 thisisaworm.infected
```

- The infected file has an ".infected" extension appended.

```
tiffany@tiffany-VirtualBox:~/Desktop/CSE331-Antivirus-master$ tree -a ~/Desktop/dir
/home/tiffany/Desktop/dir
├── .hiddensubdir
│   └── thisisavirus.infected
├── notavirus
├── subdir
│   └── thisisaworm.infected
└── .thisisatrojan.infected

2 directories, 4 files
```

Explanation of On-Demand Scanning Code:

When the antivirus program is executed, the program first checks to see if the "-f" or "--file" optional flag is provided as an argument. The program then checks to see if an absolute path was provided as an argument in the command. If not, the program determines the absolute path to aid in the performance of the rest of the operations. Once the absolute path is determined, the program checks to see if the provided path is for a directory or a file. If a file was provided, the program scans the file against the whitelist and virus database. If a directory was provided, the program scans the entire directory recursively, including subdirectories.

To scan a file, the scanFile() function is executed. The function compares the SHA1 hash of the file with the whitelist database. If the SHA-1 hash of the file matched any of the SHA-1 hashes in the the whitelist database, the function does not need to scan this file any further and returns. Otherwise, the function continues to scan. Next, the function checks to see if any of the virus signatures in our viruses database is a partial match in the sequential bytes of the file. If the file contains any of the virus signatures, the function removes all permissions for the file and appends .infected to the file's filename. Lastly, the function alerts the user via the terminal console that there has been a virus detected for the file.

To scan a directory, the scanDir() function is executed. The function scans each file and subdirectory in the directory recursively. Once a file is reached in the recursive scanning, the file is passed to the scanFile() function. By the end of the recursive scanning, all the files in the directory with virus signatures will have all permissions removed and an ".infected" extension appended to the filename. Also, there will be a message alert in the terminal console for each of the infected files.

On-Access Scanning:

Set-Up Instructions:

1) Fetch files from github onto a virtual machine running 32-bit Ubuntu version 16.04
2) Run: sudo cat /proc/kallsyms | grep sys_call_table
3) Set the value of sys_call_table in antivirus_open.c to the returned value of the command above as an unsigned long
    a) making sure to place **0x** before the value !!

4) In the linux terminal make your way to the directory where the fetched github files are stored.
5) Compile antivirus_open.c by running this command: sudo make clean all
6) Run: sudo insmod antivirus_open.ko to begin the On-Access Scanning
7) The program will now hook onto the kernel and we have set the program to “listen” for open calls (through step 3)
8) The program will now wait until the operating system makes a call to open to scan the opened file. In the case of a user he will simply open a file.
9) When a file is opened:
    a) The kernel will pick this up through the sys_call_table that we set up earlier and will immediately “intercept” this open call
    b) In doing so, it redirects the flow of the program into our own ‘open’ function which we set up to check for certain cases
10) In the custom open function:
    a) The program will check to see if the file being opened is on the whitelist database file
        i) If it is on the whitelist
            (1) Return access back to the original open function and let it proceed normally
        ii) else
            (1) Check the files, being opened, if there is any virus code from virus file database by calling the same code as our On-Demand

(a) If there exists a virus code from the list of possible viruses in the virus database file then the program will change the extension of this file to .infected and continue to listen for any more open calls

(b) If there are no viruses present then the kernel will return control to the program and proceed with normal open calls and then continue listening for any other open calls

11) Finished checking for any open functions then user need to run: rmmod antivirus_open.ko

References:

- https://github.com/ex0dus-0x/hijack

  This link was the source code of an example of hijacking the kernel call: open()

  Most of this code was used to implement On Access Scanning, to provide credit the MODULE_AUTHOR was left unchanged in antivirus_open.c

- http://se7so.blogspot.com/2012/07/hijacking-linux-system-calls-rootkit.html

  This resource explained to us what each line of code did for the source code above.

Group Member Roles:

Database Setup and Updating: Kenny Cao

On Demand Scanning: Tiffany He

On Access Scanning: Alvin Lin, Jin Zhen, Tiffany He, Kenny Cao

README.md:

CSE331 Project - Antivirus for Linux

Before running the following commands:

- Make sure the antivirus program is in your current directory
- Make sure the antivirus program is executable
- Make sure you have Ubuntu 16.04 (strictly 32-bit version only) and Python3 on VirtualBox

usage: ./antivirus [-h] [-f FILE] [-u]

optional arguments:

-h, --help	show this help message and exit
-f FILE, --file FILE	a file or directory for scanning
-u, --update	update whitelist and viruses database

Update Whitelist/Viruslist Database

./antivirus -u

OR

./antivirus --update

On Demand Scanning

./antivirus -f FILE

OR

./antivirus --file FILE

On Access Scanning

sudo make clean all
sudo insmod antivirus_open.ko

Stop On Access Scanning

```
sudo rmmod antivirus_open
```