CSE 509, SYSTEM SECURITY PROJECT

ANTIVIRUS FOR LINUX

TEAM MEMBERS:

Pratik Gaikar

Nehil Shah

Tushar Jain

Vishwajit Bhat

Table of Contents

Contents

Problem Statement:

Develop an antivirus scanner that supports updates over the Internet. The antivirus would be able to detect malware based on signatures of malicious programs. Each signature is a series of sequential bytes from the assembly instructions of an executable. The antivirus would run in two scanning modes, on-demand and on-access.

In On-demand scanning, the user can run a program to scan a file, or a directory containing files, e.g. antivirus-scan /home/jack/Downloads. If a file is found to be infected it removes all permissions (e.g. chmod 000 virus), append a ".virus" string in the file's name and alert the user.

In On-Access Scanning, when the user is trying to execute, or open a file, we scan the file before it is opened. If the file is found to be virus, then the kernel would refuse to open the file, removes all permissions, appends a ".virus" string in the file's name and alert the user.

Tools and Technologies:

VMware workstation pro.
Kernel version: 4.2.0.27
Visual Paradigm 13.2

Design:

The detection logic of antivirus is implemented in the kernel-space. The logic is at only one location in the kernel and is called by both the on demand and on access modes. The system call table is hijacked to perform an additional operation in the **open, execve, openat and execveat** system call**s**. Before actually opening the file, it is checked whether the file is a virus file. If we find that the file is not a virus we go ahead and continue opening the file. That would return file's "fd" to the caller and thus successfully **open** the file. If the file is a virus file, we rename the file **appending ".virus",** remove the permissions on the file, and notify the user of the file being a virus file. Thus the file couldn't be opened.

Modes of Operation:

On demand mode:

- Input is a directory: When the input given is a directory, then the user program recursively goes through each of the files and subdirectories in the given directory, and opens each of the file. When a file is opened, it goes to the hook in the kernel space and checks for virus. It returns a negative value and a errno in case of a virus and "fd" if not a virus. If the file was successfully opened, we close that file and move to the next file. When the file was found to be virus, the kernel communicates the user through the use of netlink socket. A user program listens to a port and gets notified as soon as the kernel writes the virus file name. Send notify is used to show virus file name graphically to the user.
- Input is a file: Just open the file which hits the hook in the kernel and the operation is same as above.
- Input can be multiple files and directories, in which all the given files and directories are scanned for virus. For each directory we recursively go through all the files same as the above case 1.

On access mode:

- When a user opens a file using any editor or through some user program, the hook in the kernel is called which checks the file for virus and allows opening only if it's not a virus. In that case a valid "fd" (file descriptor) is returned. If the file is virus, negative value is returned which does not open the file.
- The use of netlink is important in this mode as we need a way to notify to user about virus from kernel, which can thereby provide a pop up to the user about the virus file. Without netlink, In case of file being virus, the file would not be opened and there would be no pop up, which might make the user feel clueless. If just printk statements were used then the user would have had to execute dmesg to know about the virus file.

Linux System Call Hijacking:

The system call table is made read write and the following four system calls are hijacked:

a. <u>Open system call</u>

Whenever any file is opened using either the on demand or the on access mode, the open system call is invoked and by hijacking it, we first scan the file to be opened. If virus is found, we return an error and do not open the file and if no virus is found then the file is opened regularly.

b. <u>Openat system call</u>

The openat system call operates in exactly the same way as open system call with the difference that openat opens a file relative to a directory file descriptor. Whenever any file is opened using either the on demand or the on access mode, the openat system call is invoked and by hijacking it, we first scan the file to be opened. If virus is found, we return an error and do not open the file and if no virus is found then the file is opened regularly.

c. <u>Execve system call</u>

Whenever any command or file is executed using the on demand or the on access mode, the execve system call is invoked and by hijacking it, we first scan the file to be executed. If virus is found, we return an error and do not execute the file and if no virus is found then the file is executed regularly.

d. <u>Execveat system call</u>

The execveat system call operates in exactly the same way as execve system call with the difference that execveat opens a file relative to a directory file descriptor. Whenever any command or file is executed using the on demand or the on access mode, the execveat system call is invoked and by hijacking it, we first scan the file to be executed. If virus is found, we return an error and do not execute the file and if no virus is found then the file is executed regularly.

Whitelist:

Whitelist file contains the SHA-1 hashes of programs and files that are definitely not malware. This whitelist is populated with the standard programs and files of a Linux distribution to ensure that the antivirus will never break the system. If an input file's hash is on that list, then the file is not scanned against blacklist further.

Blacklist:

Blacklist consists of a set of sequences of bytes which are usually found in malicious programs and files. So, to detect if a program or a file has virus or not, we can do a byte by byte comparison of the virus pattern with the file content and check if the pattern mentioned in blacklist is present in them or not. If present it means that the it is a virus file. Otherwise, it is not a virus file. Since all the possible patterns of viruses are not known this method is not full proof but it can help detecting known viruses.

Netlink socket:

Netlink socket is a special IPC used for transferring information between kernel and user-space processes. It provides a full-duplex communication link between the two by way of standard socket APIs for user-space processes and a special kernel API for kernel modules. It is a nontrivial task to add system calls, ioctls or proc files for new features; we risk polluting the kernel and damaging the stability of the system. The code implementing a system call in the kernel is linked statically to the kernel in compilation time, thus, it is not appropriate to include system call code in a loadable module. With netlink socket, no compilation time dependency exists between the netlink core of Linux kernel and the netlink application living in loadable kernel modules.
We used netlink to communicate about the virus file from kernel space to user space. Once a virus is detected in the kernel space, it will notify the user through the user socket. The user program in turn uses “send-notify” to show about the virus graphically.

Implementation:

Below we have the details of the files in each module and their significance:

Hijacking:

main.c
Once the module is initialized, the system call address is retrieved from the kernel and it is changed from read only to read write in order to intercept it. Then, the original address of the open, openat, execve and execveat system calls are stored and replaced with our new intercepted system call implementations for the open, openat, execve and execveat system calls respectively. For each of these system calls, we first scan the file for virus and if a virus file is detected we return the -EBADF error, otherwise the original system call implementation is invoked to carry out the further process. Once the module is exiting, we again replace our new intercepted system calls with the original system call implementations so that the kernel returns to original state.

Anti Virus Helper:

anti_virus_helper.c
Whenever the new intercepted system call implementation is invoked, the antivirus helper function check_for_virus takes the appropriate parameters from the system call and processes

it. It opens the blacklist and whitelist files in read mode for virus scanning. It also opens the input file with the parameters provided and then the opened file proceeds for scanning. Before the file proceeds for whitelist and blacklist scanning, it is checked whether it's a character or a block device file, such type of files cannot be opened so they can't be scanned. After the file is checked with the whitelist and the blacklist pattern, if the file contains a virus then the rename function is called which renames the file with .virus and sets its permissions to 000.

file_operation.c

This helper file contains file operations methods for opening a file, renaming the file, changing the permissions of the file.

Whitelist:

whitelist.c

This file scans the input file and checks whether its SHA-1 hash is present in the whitelist patterns. The whitelist file is already populated with the SHA-1 of standard linux utilities, i.e. the programs and files of a Linux distribution to ensure that your antivirus will never "break" the system. The input file's contents are read and its sha1 is calculated from its contents. The calculated sha1 is then checked whether it matches with any of the whitelist patterns present in the whitelist file. If a match is found, then the input file is interpreted as a safe file and is not further checked for blacklist patterns. If no match occurs, then the input file is further checked for blacklist pattern.

Blacklist:

blacklist.c

Once the file is checked for whitelist pattern and no match is found then the input file is checked whether its contents matches with one of the blacklist patterns. The blacklist file has all the virus patterns and each pattern is a series of sequential bytes. The input file is scanned and if its contents match with one of the virus patterns then a virus would be detected and that file would be treated as a virus file. If the input file does not match with any of the virus patterns then no action is executed on the file. On detecting a file as a virus, that file is renamed to .virus, its permissions are set to 000 and if we try to open or execute this file an error "Bad File Descriptor" (-EBADF) would occur indicating that the file could not be opened.

Netlink Socket

main.c

The communication between kernel and user space is achieved through socket netlink. Once the antivirus module is initialized, the socket is created between the kernel side and the user side. Whenever, we want to notify the user regarding a virus file, we'll send this message out from kernel space to the user space on this socket connection. Once, the module is uninstalled, the socket connection between the kernel and user is terminated.

user.c

This file contains the user space code of the socket connection between the kernel and the user. It listens for any messages coming from the kernel and if any message arrives, it notifies the user by creating a pop up on the screen so that the user is correctly aware if any virus file is found while on demand or on access scanning.

User Scan (On Demand Scan)

antivirus_check.c
This file is the user space program for the functionality of on demand scanning. The user executes this program in order to scan certain files or directories. The user can provide any number of files or directories while executing this program and all these inputs provided would be scanned for any virus in them. We used nftw library for recursively iterating within the directories so that we could get all the files within the directory provided as input by the user. Each file is scanned and if virus occurs in any of them then the user is appropriately notified about this. Also, a summary is provided to the user with the number of files scanned and the number of virus files detected.

Scripts:

antivirus_update
This script updates the existing blacklist and whitelist files stored at a server. By executing this script, the updated blacklist and whitelist files at a server would replace the current files. Also, the SHA-1 hash of the blacklist and whitelist files are added to the whitelist file.

antivirus.config
This file contains the server path where the blacklist and whitelist files are stored.

antivirus_install
This script performs the following functions:

- **a.** It compiles all the user program files and copies its executables to /usr/bin folder by which the files can be executed from anywhere in the kernel.
- **b.** It creates a directory /etc/antivirusfiles which would contain the blacklist and the whitelist files.
- **c.** It copies the antivirus_update script to /usr/bin and executes it by which the blacklist and whitelist kept at a server are copied into the /etc/antivirusfiles location It builds and starts the execution of the kernel module.
- **d.** It calculates the SHA-1 hash of the whitelist and blacklist files and adds it to the whitelist patterns.

sha1sum.sh
This script calculates the SHA-1 hashes of the linux internal files and utilities.

antivirus_uninstall
This script is used to uninstall the antivirus module.

testprogram
This file is used for programmatically testing all the open and execve related system calls by calling all of them within a C program (testprogram.c)

Installation & Testing:

The antivirus software can be installed only if the user has root privileges. Antivirus scan can be executed by normal user.

Here are the steps to get the antivirus code and install antivirus:

1. The module is installed on the VMWARE virtual machine with linux kernel version 4.2.0-27-generic.
   Run the following git clone command in the /usr/src/linux-headers-4.2.0-27-generic folder to clone the code present in the git repository LinuxAntivirus:
   ```
   git clone https://github.com/pratikgaikar/LinuxAntivirus.git
   ```

2. To update the whitelist and blacklist file, the following script is to be executed when the antivirus module is not inserted:
   ```
   antivirus_update
   ```

3. To install antivirus run the following script from the directory /usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus/scripts/:
   ```
   antivirus_install
   ```

4. After the script is executed, the antivirus module is inserted into the kernel and scanning starts

5. The on demand mode of antivirus scan can be executed using the following command:
   ```
   antivirus_scan testfiles/
   ```
   This would scan the whole of testfiles directory.
   ```
   antivirus_scan testfiles/testfile1
   ```
   This would scan only testfile1 of testfiles directory.
   ```
   antivirus_scan testfiles/   testfiles2/
   ```
   This would first scan testfiles directory, then scan testfiles2 directory.

6. Every file which is opened or executed by the user is scanned for virus and appropriate action is taken.
7. Run cat command to open a virus file - On Access mode
   ```
   cat testfiles/testfile1
   ```
8. The module could also be tested by executing the following command within the /usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus/scripts/ folder by which all the open and execve related system calls are invoked
   ```
   ./testprogram
   ```

Distinguishing Features:

Netlink API:

We have implemented the virus scan in kernel space. This is useful as a particular user might not have access to some directories and files, and thus makes sense to the scan in kernel space. Since we come to know whether a file is a virus or not in the kernel space, we need a way to communicate this to the user space. Various ways to do it are:

Shared Memory: We can keep a file in which the kernel writes and a user program continuously runs reading the file. This involves IO operation and also a performance hit as the User program has to continuously run reading the file.

Notify via socket: We used server and client sockets to communicate from kernel to user space. Once a virus is detected in the kernel space, it writes it to the server socket. The user socket listens to the client port, and gets notified when something is written to its port.

In this way, we avoid IO operation and also avoid user continuously checking if virus is found. This optimizes the performance of the antivirus.

Page Size blacklist checking:

Since kernel uses physical memory, a huge file would not get completely loaded into memory due to limited main memory. So, we read a file into a buffer of PAGE SIZE and compare it with the blacklist bytes. We reuse the same buffer to continue the following bytes of the file. This makes sure that we use only PAGE SIZE of memory to compare the whole file.

Whitelisted all the libc files:

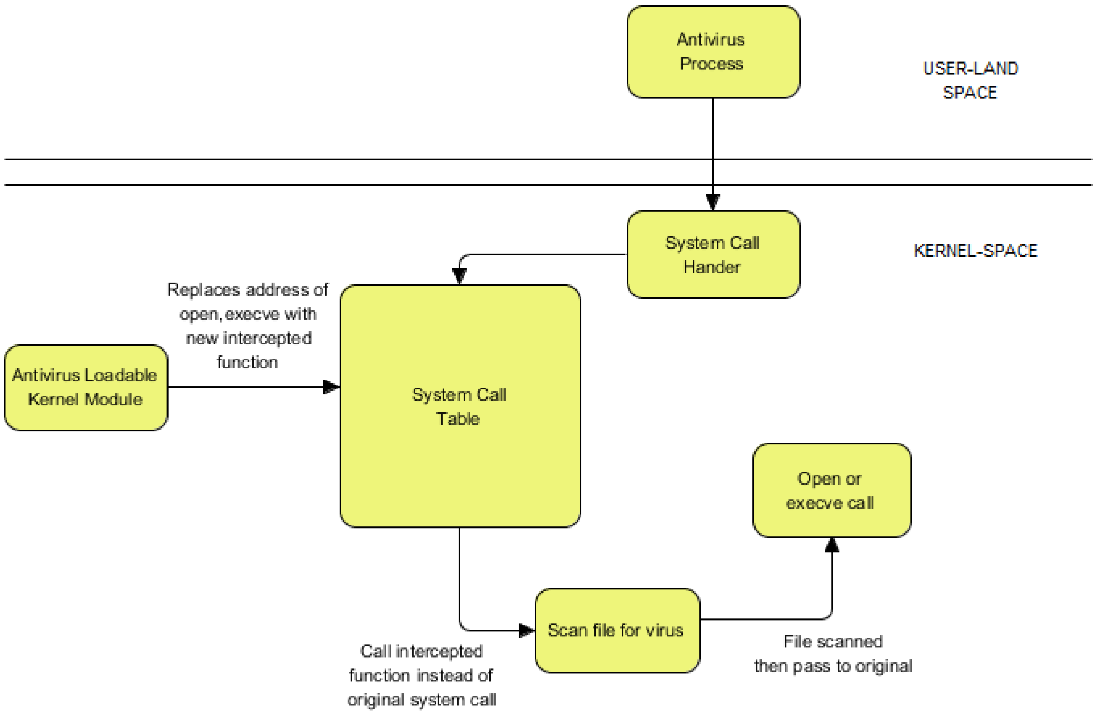
Since we have a hook in 4 system calls, various applications come across this hook and a huge number of system files will encounter this hook.

Linux kernel version 4.2:

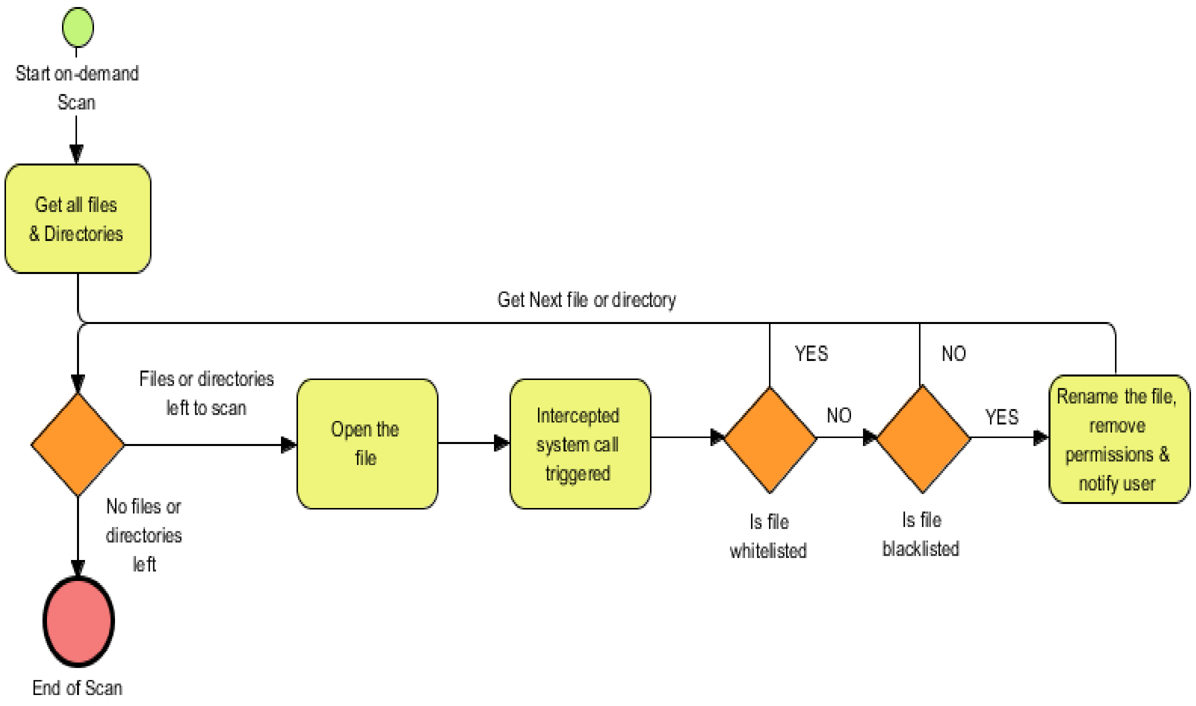
Implemented antivirus on closer to the latest official release version of kernel.

Block Diagram

Antivirus Module

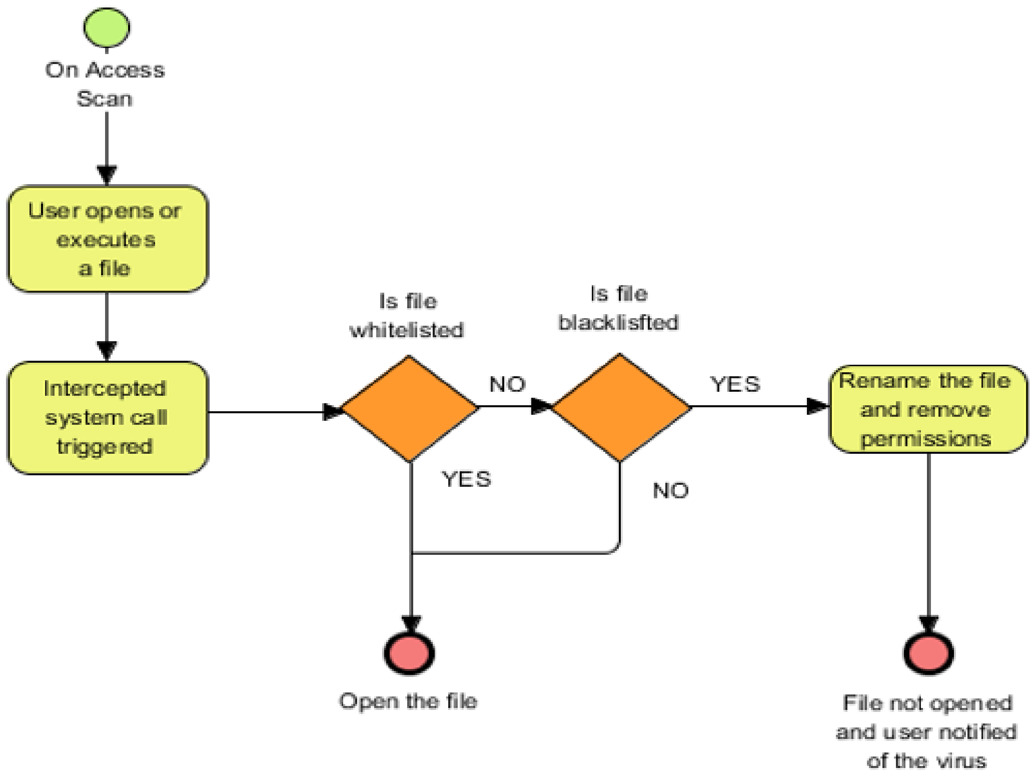

Antivirus Process
USER-LAND SPACE
System Call Hander
KERNEL-SPACE
Replaces address of open, execve with new intercepted function
Antivirus Loadable Kernel Module
System Call Table
Open or execve call
Scan file for virus
Call intercepted function instead of original system call
File scanned then pass to original

On Demand Scan



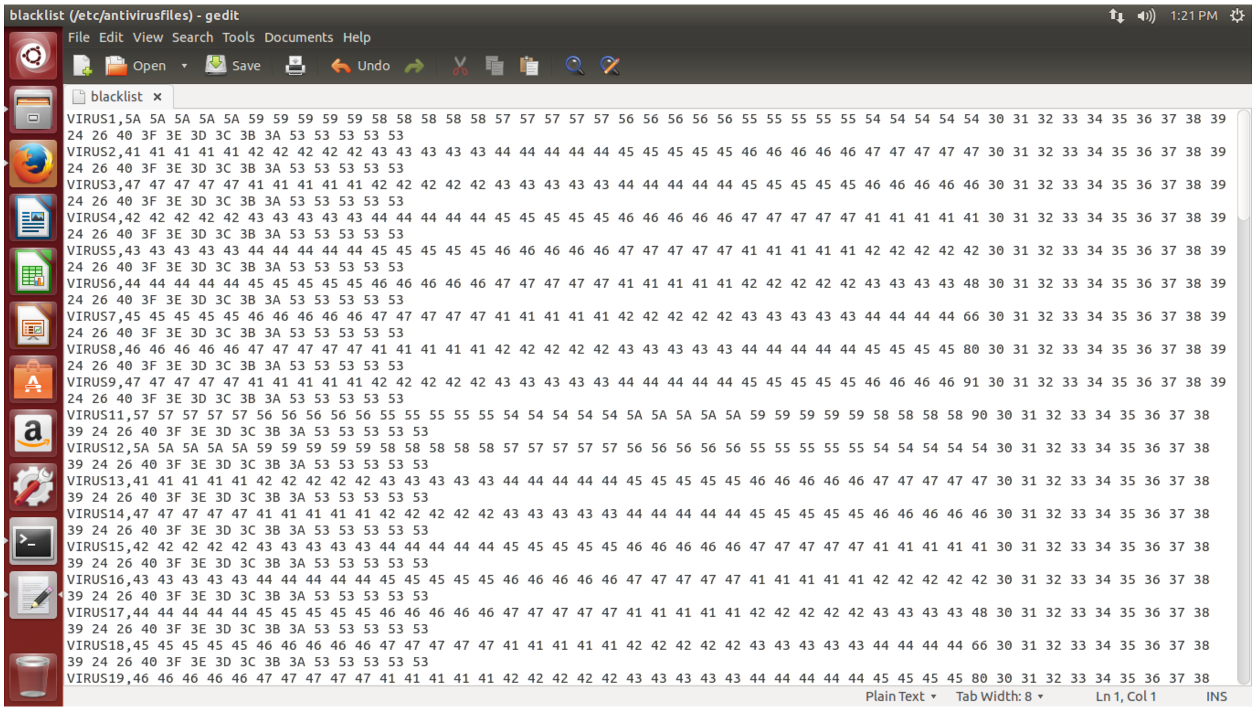
Start on-demand Scan
Get all files & Directories
Get Next file or directory
Files or directories left to scan
Open the file
Intercepted system call triggered
YES
NO
NO
YES
Is file whitelisted
Is file blacklisted
Rename the file, remove permissions & notify user
No files or directories left
End of Scan

On Access Scan


On Access Scan
User opens or executes a file
Intercepted system call triggered
Is file whitelisted
NO
Is file blacklisfted
YES
Rename the file and remove permissions
YES
NO
Open the file
File not opened and user notified of the virus

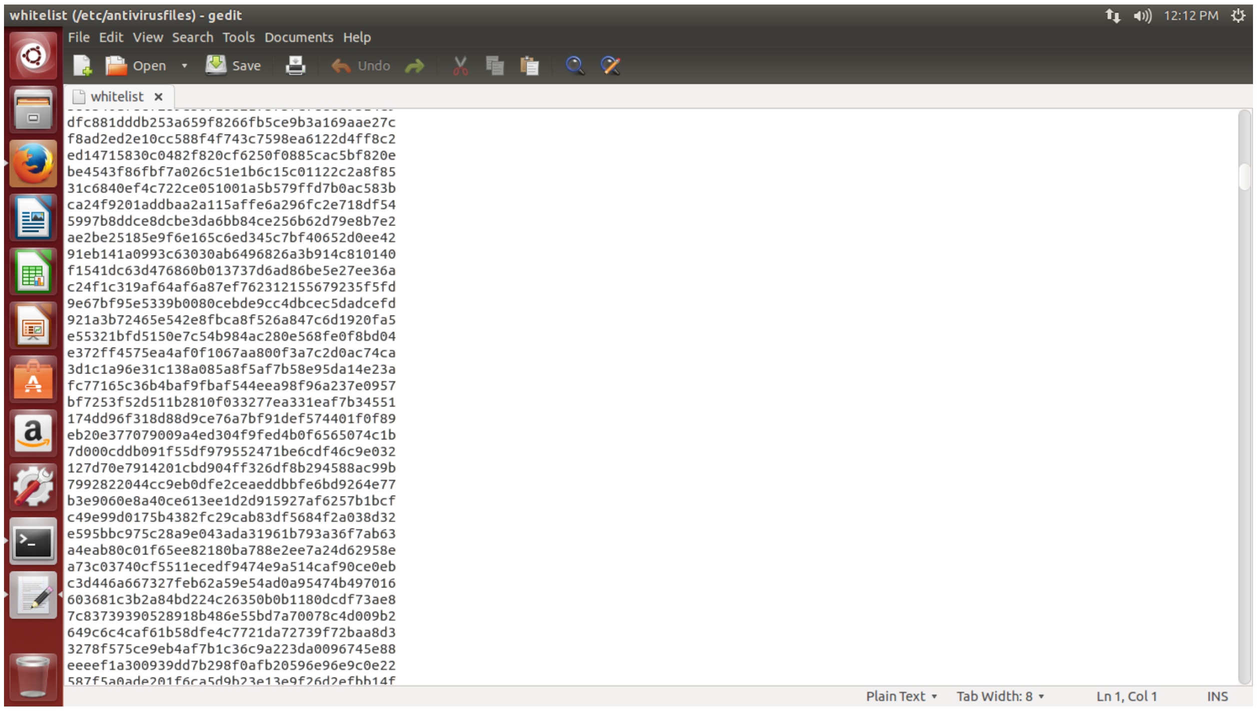
Snapshots:

1. Blacklist file:


blacklist (/etc/antivirusfiles) - gedit
1:21 PM
File Edit View Search Tools Documents Help
Open
Save
Undo
blacklist
Plain Text
Tab Width: 8
Ln 1, Col 1
INS

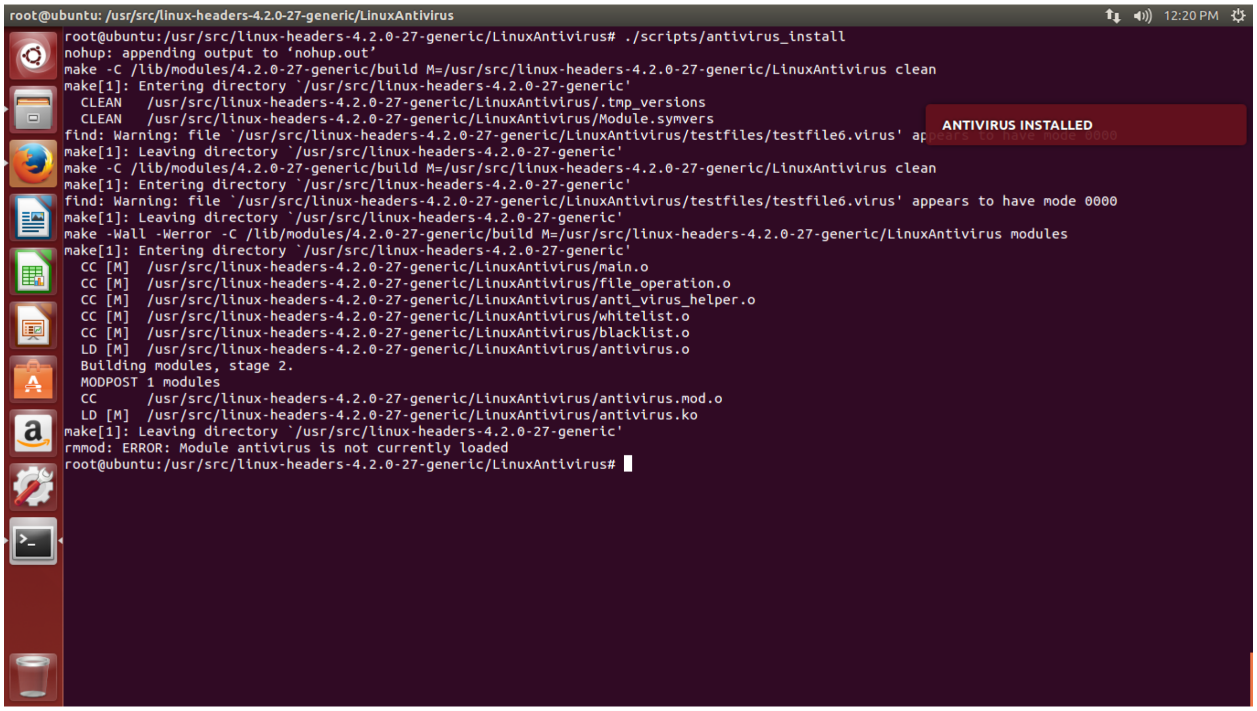
```
VIRUS1,5A 5A 5A 5A 5A 59 59 59 59 59 58 58 58 58 58 57 57 57 57 57 56 56 56 56 56 55 55 55 55 55 54 54 54 54 54 30 31 32 33 34 35 36 37 38 39
24 26 40 3F 3E 3D 3C 3B 3A 53 53 53 53 53
VIRUS2,41 41 41 41 41 42 42 42 42 42 43 43 43 43 43 44 44 44 44 44 45 45 45 45 45 46 46 46 46 46 47 47 47 47 47 30 31 32 33 34 35 36 37 38 39
24 26 40 3F 3E 3D 3C 3B 3A 53 53 53 53 53
VIRUS3,47 47 47 47 47 41 41 41 41 41 42 42 42 42 42 43 43 43 43 43 44 44 44 44 44 45 45 45 45 45 46 46 46 46 46 30 31 32 33 34 35 36 37 38 39
24 26 40 3F 3E 3D 3C 3B 3A 53 53 53 53 53
VIRUS4,42 42 42 42 42 43 43 43 43 43 44 44 44 44 44 45 45 45 45 45 46 46 46 46 46 47 47 47 47 47 41 41 41 41 41 30 31 32 33 34 35 36 37 38 39
24 26 40 3F 3E 3D 3C 3B 3A 53 53 53 53 53
VIRUS5,43 43 43 43 43 44 44 44 44 44 45 45 45 45 45 46 46 46 46 46 47 47 47 47 47 41 41 41 41 41 42 42 42 42 42 30 31 32 33 34 35 36 37 38 39
24 26 40 3F 3E 3D 3C 3B 3A 53 53 53 53 53
VIRUS6,44 44 44 44 44 45 45 45 45 45 46 46 46 46 46 47 47 47 47 47 41 41 41 41 41 42 42 42 42 42 43 43 43 43 48 30 31 32 33 34 35 36 37 38 39
24 26 40 3F 3E 3D 3C 3B 3A 53 53 53 53 53
VIRUS7,45 45 45 45 45 46 46 46 46 46 47 47 47 47 47 41 41 41 41 41 42 42 42 42 42 43 43 43 43 43 44 44 44 44 66 30 31 32 33 34 35 36 37 38 39
24 26 40 3F 3E 3D 3C 3B 3A 53 53 53 53 53
VIRUS8,46 46 46 46 46 47 47 47 47 47 41 41 41 41 41 42 42 42 42 42 43 43 43 43 43 44 44 44 44 44 45 45 45 45 80 30 31 32 33 34 35 36 37 38 39
24 26 40 3F 3E 3D 3C 3B 3A 53 53 53 53 53
VIRUS9,47 47 47 47 47 41 41 41 41 41 42 42 42 42 42 43 43 43 43 43 44 44 44 44 44 45 45 45 45 45 46 46 46 46 91 30 31 32 33 34 35 36 37 38 39
24 26 40 3F 3E 3D 3C 3B 3A 53 53 53 53 53
VIRUS11,57 57 57 57 57 56 56 56 56 56 55 55 55 55 55 54 54 54 54 54 5A 5A 5A 5A 5A 59 59 59 59 59 58 58 58 58 90 30 31 32 33 34 35 36 37 38
39 24 26 40 3F 3E 3D 3C 3B 3A 53 53 53 53 53
VIRUS12,5A 5A 5A 5A 5A 59 59 59 59 59 58 58 58 58 58 57 57 57 57 57 56 56 56 56 56 55 55 55 55 55 54 54 54 54 54 30 31 32 33 34 35 36 37 38
39 24 26 40 3F 3E 3D 3C 3B 3A 53 53 53 53 53
VIRUS13,41 41 41 41 41 42 42 42 42 42 43 43 43 43 43 44 44 44 44 44 45 45 45 45 45 46 46 46 46 46 47 47 47 47 47 30 31 32 33 34 35 36 37 38
39 24 26 40 3F 3E 3D 3C 3B 3A 53 53 53 53 53
VIRUS14,47 47 47 47 47 41 41 41 41 41 42 42 42 42 42 43 43 43 43 43 44 44 44 44 44 45 45 45 45 45 46 46 46 46 46 30 31 32 33 34 35 36 37 38
39 24 26 40 3F 3E 3D 3C 3B 3A 53 53 53 53 53
VIRUS15,42 42 42 42 42 43 43 43 43 43 44 44 44 44 44 45 45 45 45 45 46 46 46 46 46 47 47 47 47 47 41 41 41 41 41 30 31 32 33 34 35 36 37 38
39 24 26 40 3F 3E 3D 3C 3B 3A 53 53 53 53 53
VIRUS16,43 43 43 43 43 44 44 44 44 44 45 45 45 45 45 46 46 46 46 46 47 47 47 47 47 41 41 41 41 41 42 42 42 42 42 30 31 32 33 34 35 36 37 38
39 24 26 40 3F 3E 3D 3C 3B 3A 53 53 53 53 53
VIRUS17,44 44 44 44 44 45 45 45 45 45 46 46 46 46 46 47 47 47 47 47 41 41 41 41 41 42 42 42 42 42 43 43 43 43 48 30 31 32 33 34 35 36 37 38
39 24 26 40 3F 3E 3D 3C 3B 3A 53 53 53 53 53
VIRUS18,45 45 45 45 45 46 46 46 46 46 47 47 47 47 47 41 41 41 41 41 42 42 42 42 42 43 43 43 43 43 44 44 44 44 66 30 31 32 33 34 35 36 37 38
39 24 26 40 3F 3E 3D 3C 3B 3A 53 53 53 53 53
VIRUS19,46 46 46 46 46 47 47 47 47 47 41 41 41 41 41 42 42 42 42 42 43 43 43 43 43 44 44 44 44 44 45 45 45 45 80 30 31 32 33 34 35 36 37 38
```

2. Whitelist file:


whitelist (/etc/antivirusfiles) - gedit
12:12 PM
File Edit View Search Tools Documents Help
Open
Save
Undo
whitelist
Plain Text
Tab Width: 8
Ln 1, Col 1
INS

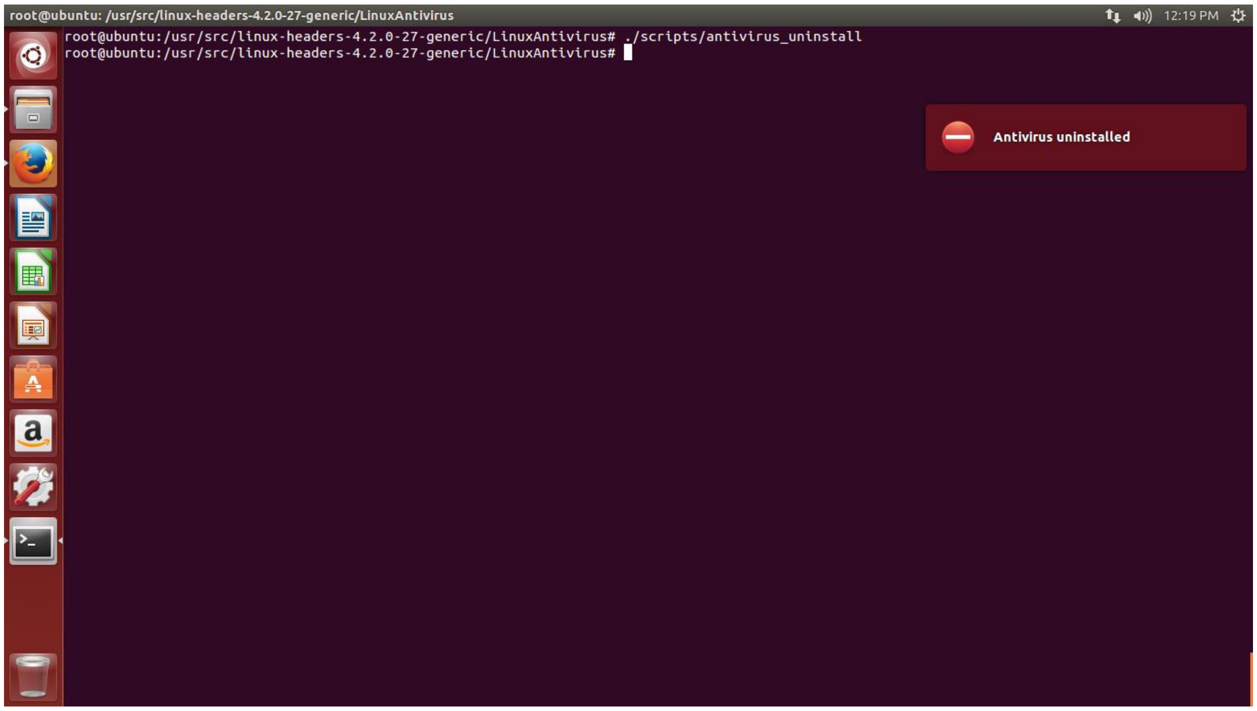
```
dfc881dddb253a659f8266fb5ce9b3a169aae27c
f8ad2ed2e10cc588f4f743c7598ea6122d4ff8c2
ed14715830c0482f820cf6250f0885cac5bf820e
be4543f86fbf7a026c51e1b6c15c01122c2a8f85
31c6840ef4c722ce051001a5b579ffd7b0ac583b
ca24f9201addbaa2a115affe6a296fc2e718df54
5997b8ddce8dcbe3da6bb84ce256b62d79e8b7e2
ae2be25185e9f6e165c6ed345c7bf40652d0ee42
91eb141a0993c63030ab6496826a3b914c810140
f1541dc63d476860b013737d6ad86be5e27ee36a
c24f1c319af64af6a87ef762312155679235f5fd
9e67bf95e5339b0080cebde9cc4dbcec5dadcefd
921a3b72465e542e8fbca8f526a847c6d1920fa5
e55321bfd5150e7c54b984ac280e568fe0f8bd04
e372ff4575ea4af0f1067aa800f3a7c2d0ac74ca
3d1c1a96e31c138a085a8f5af7b58e95da14e23a
fc77165c36b4baf9fbaf544eea98f96a237e0957
bf7253f52d511b2810f033277ea331eaf7b34551
174dd96f318d88d9ce76a7bf91def574401f0f89
eb20e377079009a4ed304f9fed4b0f6565074c1b
7d000cddb091f55df979552471be6cdf46c9e032
127d70e7914201cbd904ff326df8b294588ac99b
7992822044cc9eb0dfe2ceaeddbbfe6bd9264e77
b3e9060e8a40ce613ee1d2d915927af6257b1bcf
c49e99d0175b4382fc29cab83df5684f2a038d32
e595bbc975c28a9e043ada31961b793a36f7ab63
a4eab80c01f65ee82180ba788e2ee7a24d62958e
a73c03740cf5511ecedf9474e9a514caf90ce0eb
c3d446a667327feb62a59e54ad0a95474b497016
603681c3b2a84bd224c26350b0b1180dcdf73ae8
7c83739390528918b486e55bd7a70078c4d009b2
649c6c4caf61b58dfe4c7721da72739f72baa8d3
3278f575ce9eb4af7b1c36c9a223da0096745e88
eeeef1a300939dd7b298f0afb20596e96e9c0e22
587f5a0ade201f6ca5d9b23e13e9f26d2efbb14f
```

3. Antivirus Installed:


root@ubuntu: /usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus
12:20 PM
ANTIVIRUS INSTALLED

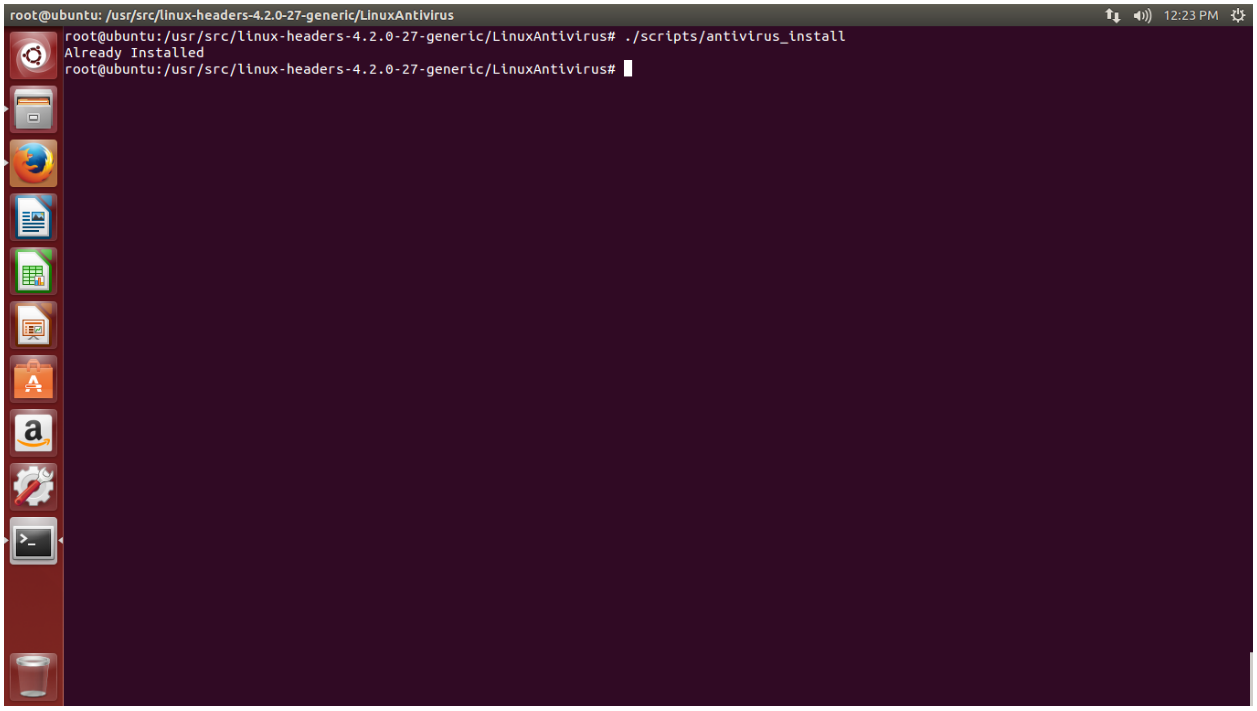
```
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# ./scripts/antivirus_install
nohup: appending output to 'nohup.out'
make -C /lib/modules/4.2.0-27-generic/build M=/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus clean
make[1]: Entering directory `/usr/src/linux-headers-4.2.0-27-generic'
  CLEAN   /usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus/.tmp_versions
  CLEAN   /usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus/Module.symvers
find: Warning: file `/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus/testfiles/testfile6.virus' appears to have mode 0000
make[1]: Leaving directory `/usr/src/linux-headers-4.2.0-27-generic'
make -C /lib/modules/4.2.0-27-generic/build M=/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus clean
make[1]: Entering directory `/usr/src/linux-headers-4.2.0-27-generic'
find: Warning: file `/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus/testfiles/testfile6.virus' appears to have mode 0000
make[1]: Leaving directory `/usr/src/linux-headers-4.2.0-27-generic'
make -Wall -Werror -C /lib/modules/4.2.0-27-generic/build M=/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus modules
make[1]: Entering directory `/usr/src/linux-headers-4.2.0-27-generic'
  CC [M]  /usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus/main.o
  CC [M]  /usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus/file_operation.o
  CC [M]  /usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus/anti_virus_helper.o
  CC [M]  /usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus/whitelist.o
  CC [M]  /usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus/blacklist.o
  LD [M]  /usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus/antivirus.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus/antivirus.mod.o
  LD [M]  /usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus/antivirus.ko
make[1]: Leaving directory `/usr/src/linux-headers-4.2.0-27-generic'
rmmod: ERROR: Module antivirus is not currently loaded
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# 
```

4. Antivirus Uninstalled:


root@ubuntu: /usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus
12:19 PM
Antivirus uninstalled

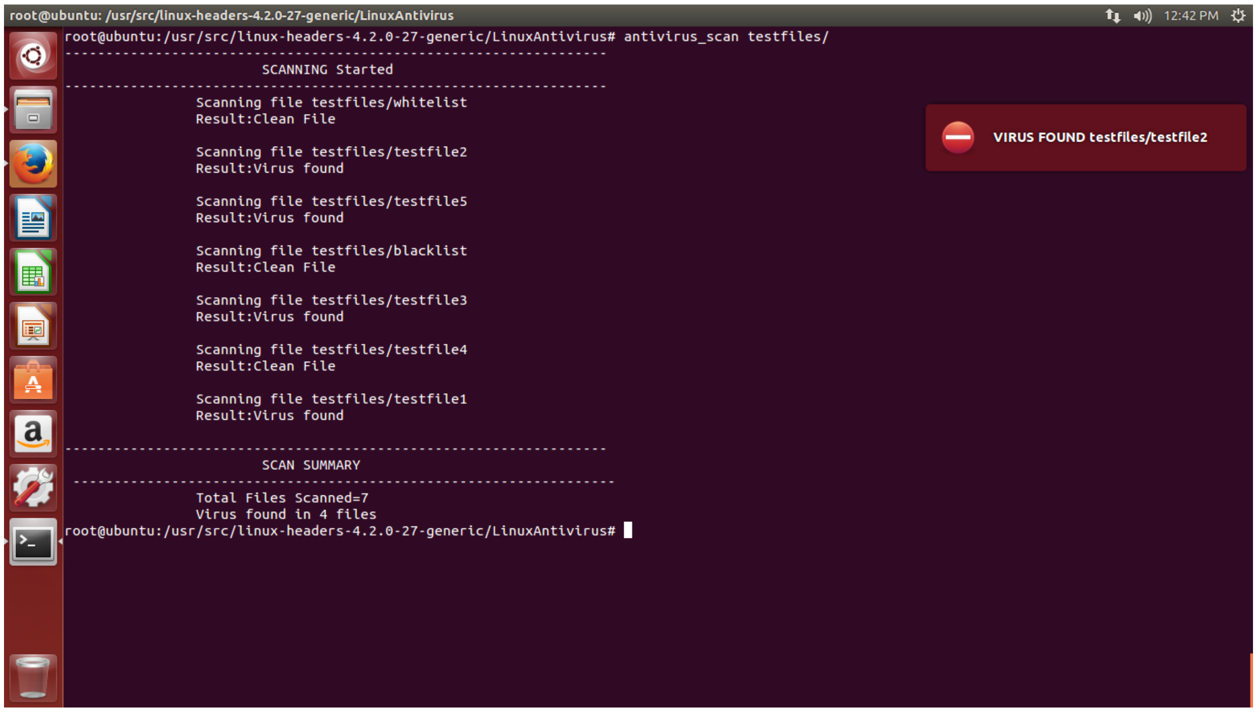
```
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# ./scripts/antivirus_uninstall
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# 
```

5. When tried to antivirus install again, it gives an appropriate message:

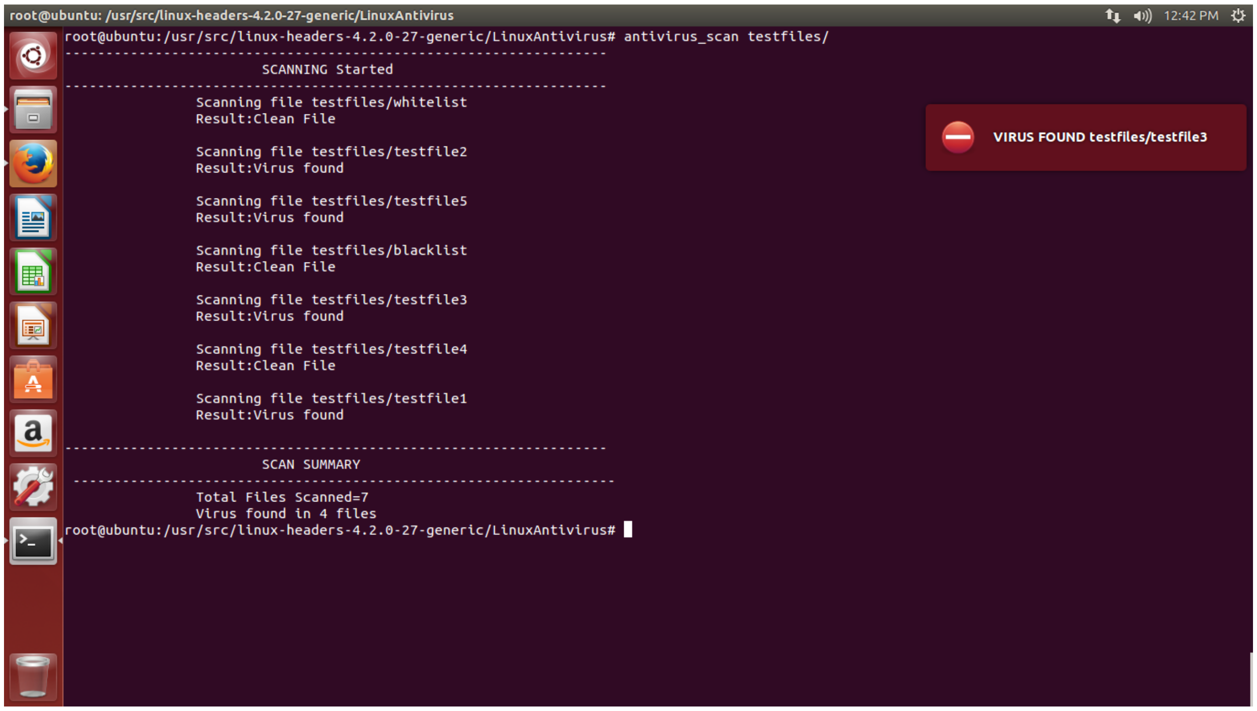

root@ubuntu: /usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus
12:23 PM

```
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# ./scripts/antivirus_install
Already Installed
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus#
```

6. On Demand scan: User notified of virus files:

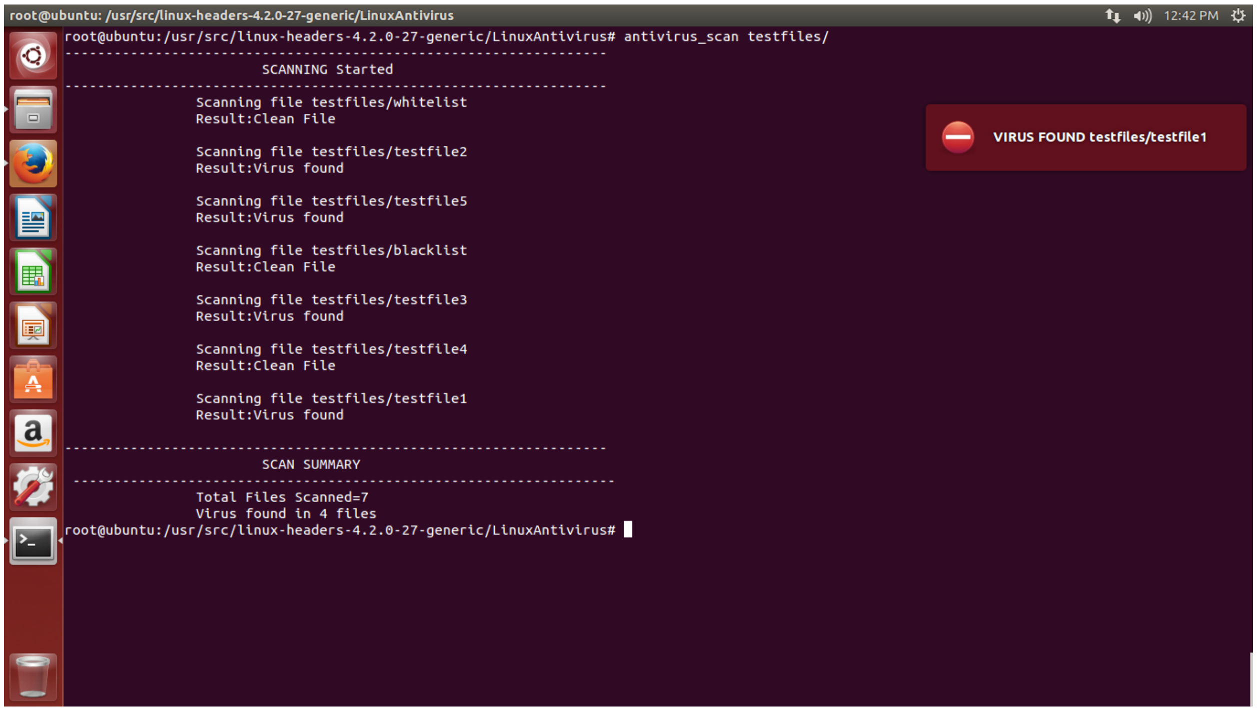

root@ubuntu: /usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus
12:42 PM
VIRUS FOUND testfiles/testfile2

```
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# antivirus_scan testfiles/
------------------------------------------------------------------
                        SCANNING Started
------------------------------------------------------------------
                Scanning file testfiles/whitelist
                Result:Clean File

                Scanning file testfiles/testfile2
                Result:Virus found

                Scanning file testfiles/testfile5
                Result:Virus found

                Scanning file testfiles/blacklist
                Result:Clean File

                Scanning file testfiles/testfile3
                Result:Virus found

                Scanning file testfiles/testfile4
                Result:Clean File

                Scanning file testfiles/testfile1
                Result:Virus found

------------------------------------------------------------------
                        SCAN SUMMARY
 ------------------------------------------------------------------
                Total Files Scanned=7
                Virus found in 4 files
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus#
```



```
root@ubuntu: /usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus                12:42 PM
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# antivirus_scan testfiles/
-------------------------------------------------------------------
                        SCANNING Started
-------------------------------------------------------------------
                Scanning file testfiles/whitelist
                Result:Clean File

                Scanning file testfiles/testfile2
                Result:Virus found

                Scanning file testfiles/testfile5
                Result:Virus found

                Scanning file testfiles/blacklist
                Result:Clean File

                Scanning file testfiles/testfile3
                Result:Virus found

                Scanning file testfiles/testfile4
                Result:Clean File

                Scanning file testfiles/testfile1
                Result:Virus found

-------------------------------------------------------------------
                        SCAN SUMMARY
 -------------------------------------------------------------------
                Total Files Scanned=7
                Virus found in 4 files
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# 
```

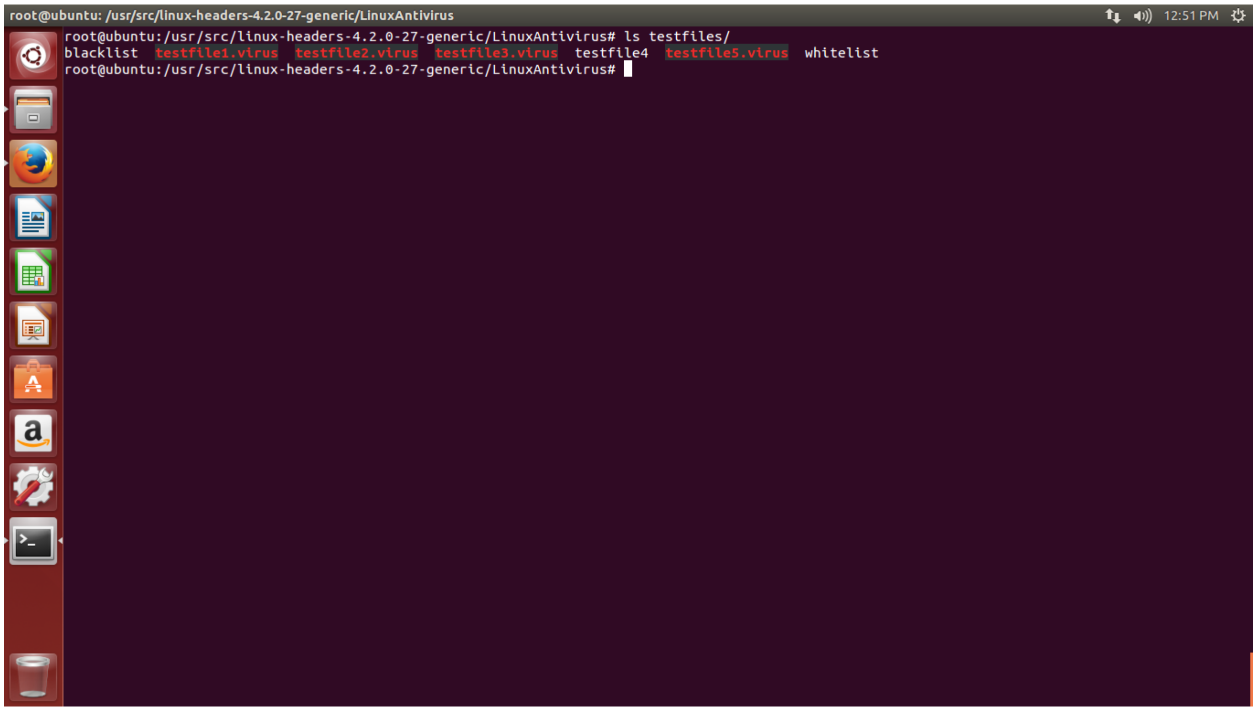
VIRUS FOUND testfiles/testfile3



```
root@ubuntu: /usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus                12:42 PM
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# antivirus_scan testfiles/
-------------------------------------------------------------------
                        SCANNING Started
-------------------------------------------------------------------
                Scanning file testfiles/whitelist
                Result:Clean File

                Scanning file testfiles/testfile2
                Result:Virus found

                Scanning file testfiles/testfile5
                Result:Virus found

                Scanning file testfiles/blacklist
                Result:Clean File

                Scanning file testfiles/testfile3
                Result:Virus found

                Scanning file testfiles/testfile4
                Result:Clean File

                Scanning file testfiles/testfile1
                Result:Virus found

-------------------------------------------------------------------
                        SCAN SUMMARY
 -------------------------------------------------------------------
                Total Files Scanned=7
                Virus found in 4 files
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# 
```

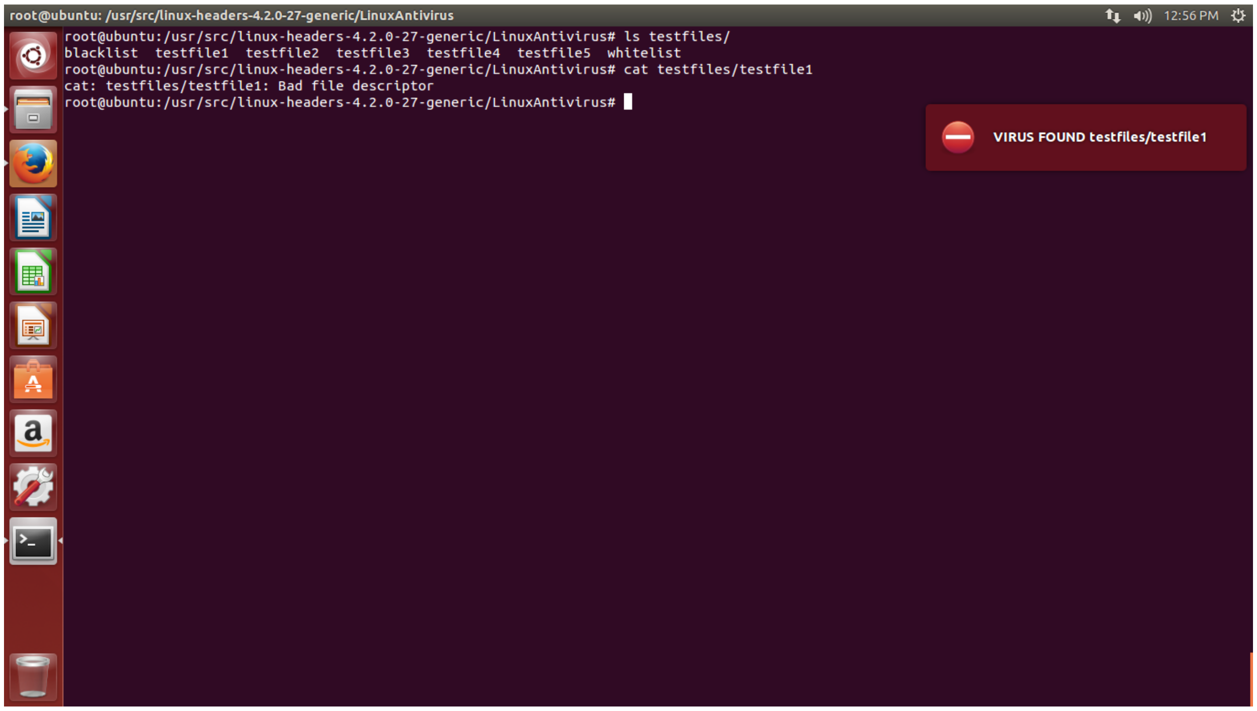
VIRUS FOUND testfiles/testfile1

7. All virus files are renamed, permissions removed and thus colored red:



```
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# ls testfiles/
blacklist  testfile1.virus  testfile2.virus  testfile3.virus  testfile4  testfile5.virus  whitelist
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# 
```

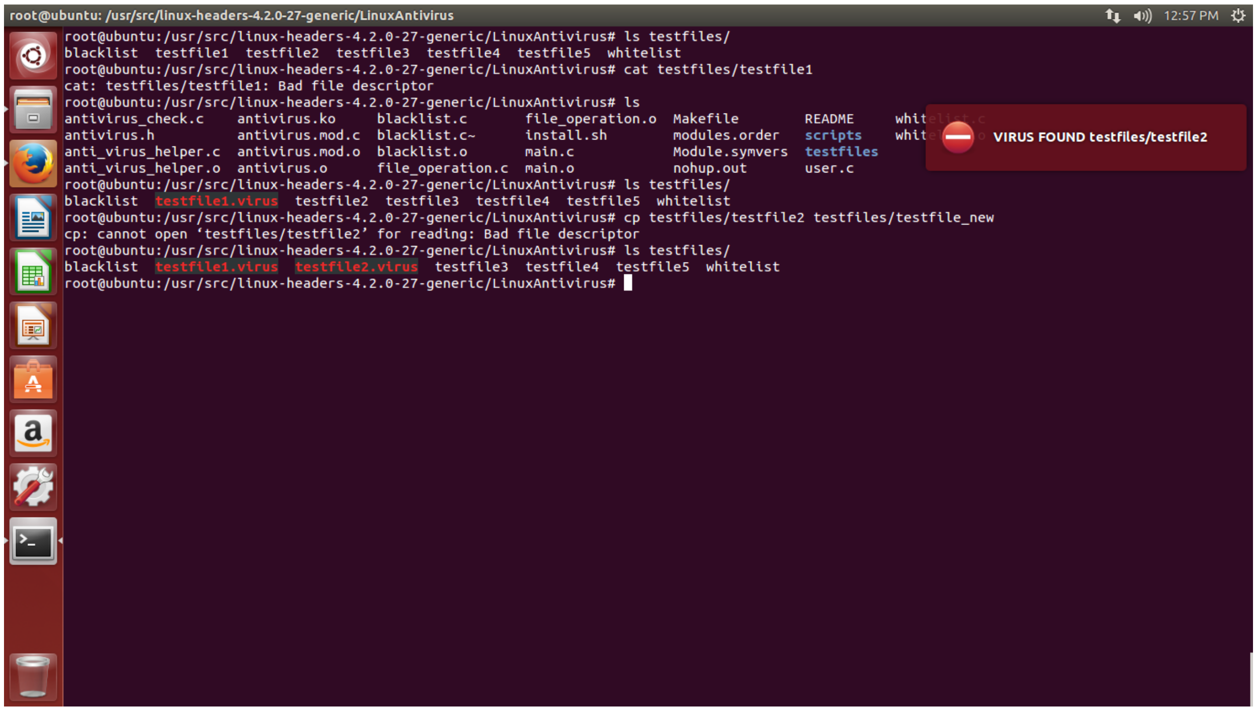
8. On Access mode: When tried to cat a virus file:



```
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# ls testfiles/
blacklist  testfile1  testfile2  testfile3  testfile4  testfile5  whitelist
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# cat testfiles/testfile1
cat: testfiles/testfile1: Bad file descriptor
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# 
```

VIRUS FOUND testfiles/testfile1

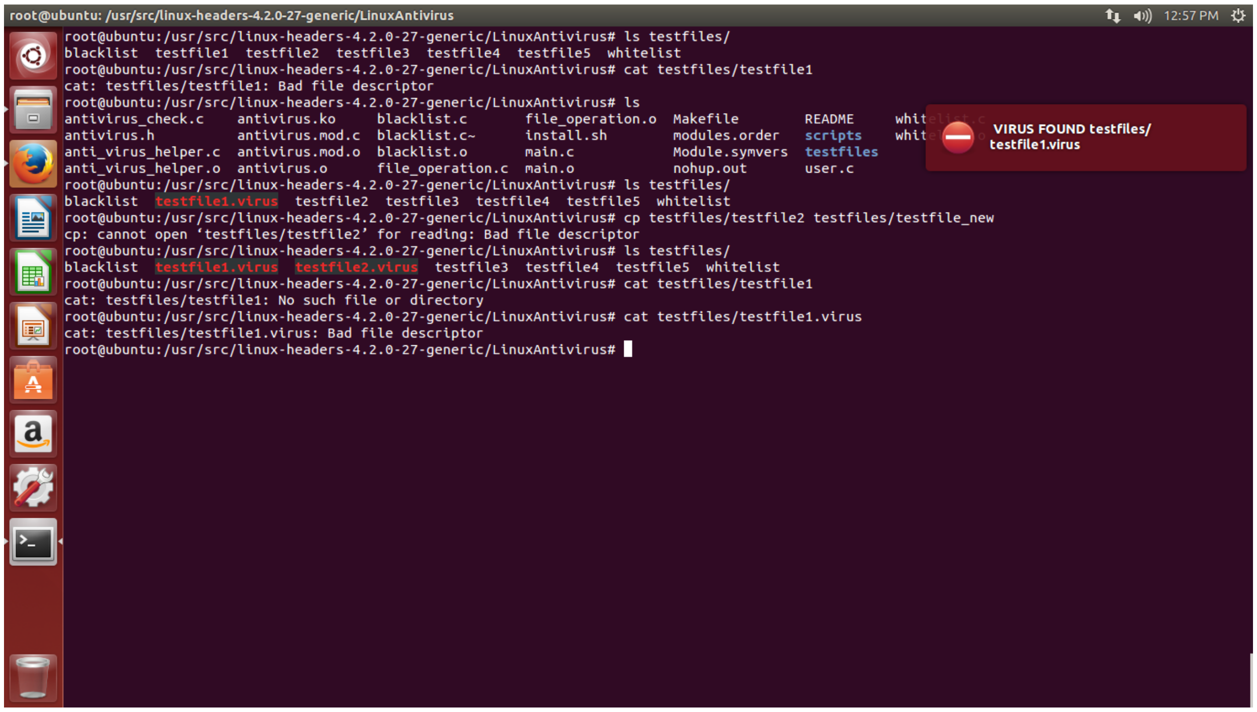
9. When tried to copy a virus file, it notifies the user of virus and does not allow to copy:



```
root@ubuntu: /usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus                12:57 PM
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# ls testfiles/
blacklist  testfile1  testfile2  testfile3  testfile4  testfile5  whitelist
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# cat testfiles/testfile1
cat: testfiles/testfile1: Bad file descriptor
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# ls
antivirus_check.c    antivirus.ko     blacklist.c       file_operation.o  Makefile        README     whit
antivirus.h          antivirus.mod.c  blacklist.c~      install.sh        modules.order   scripts    whit
anti_virus_helper.c  antivirus.mod.o  blacklist.o       main.c            Module.symvers  testfiles
anti_virus_helper.o  antivirus.o      file_operation.c  main.o            nohup.out       user.c
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# ls testfiles/
blacklist  testfile1.virus  testfile2  testfile3  testfile4  testfile5  whitelist
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# cp testfiles/testfile2 testfiles/testfile_new
cp: cannot open 'testfiles/testfile2' for reading: Bad file descriptor
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# ls testfiles/
blacklist  testfile1.virus  testfile2.virus  testfile3  testfile4  testfile5  whitelist
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus#
```

VIRUS FOUND testfiles/testfile2

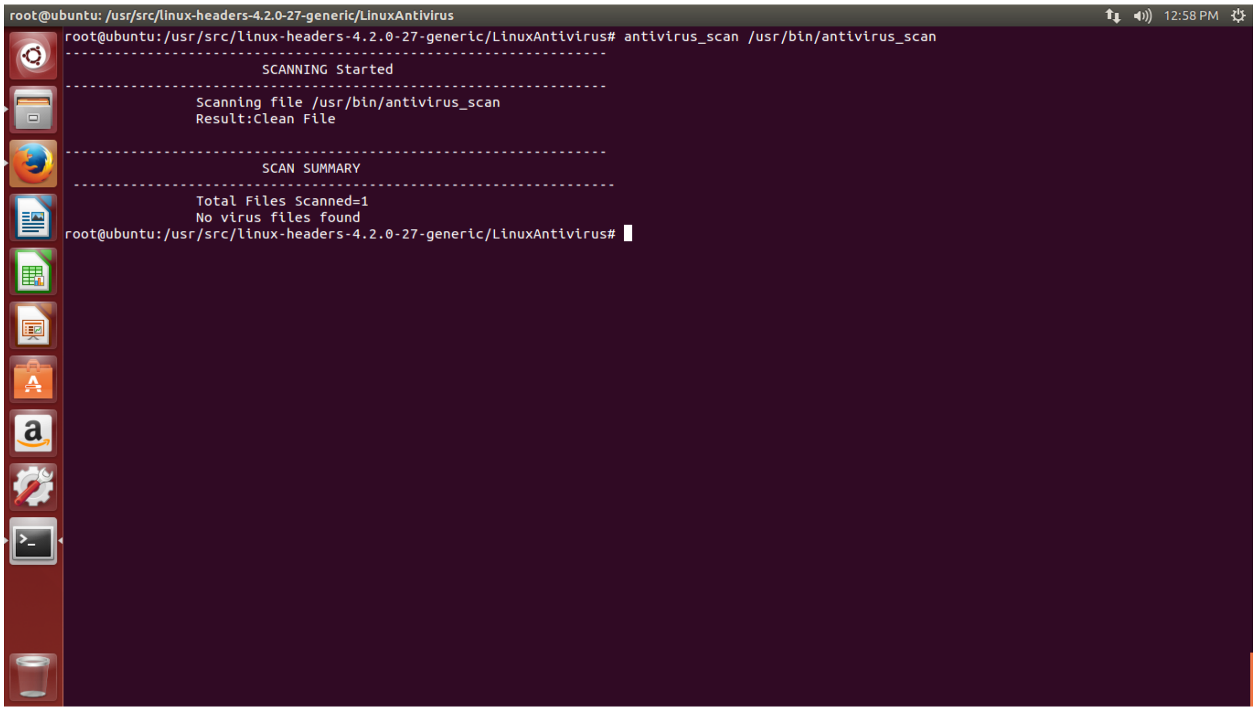
10. When tried to cat a virus file which existed before running the virus program and renamed virus file:



```
root@ubuntu: /usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus                12:57 PM
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# ls testfiles/
blacklist  testfile1  testfile2  testfile3  testfile4  testfile5  whitelist
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# cat testfiles/testfile1
cat: testfiles/testfile1: Bad file descriptor
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# ls
antivirus_check.c    antivirus.ko     blacklist.c       file_operation.o  Makefile        README     whit
antivirus.h          antivirus.mod.c  blacklist.c~      install.sh        modules.order   scripts    whit
anti_virus_helper.c  antivirus.mod.o  blacklist.o       main.c            Module.symvers  testfiles
anti_virus_helper.o  antivirus.o      file_operation.c  main.o            nohup.out       user.c
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# ls testfiles/
blacklist  testfile1.virus  testfile2  testfile3  testfile4  testfile5  whitelist
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# cp testfiles/testfile2 testfiles/testfile_new
cp: cannot open 'testfiles/testfile2' for reading: Bad file descriptor
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# ls testfiles/
blacklist  testfile1.virus  testfile2.virus  testfile3  testfile4  testfile5  whitelist
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# cat testfiles/testfile1
cat: testfiles/testfile1: No such file or directory
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# cat testfiles/testfile1.virus
cat: testfiles/testfile1.virus: Bad file descriptor
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus#
```

VIRUS FOUND testfiles/testfile1.virus

11. When tried to scan an executable:



root@ubuntu: /usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus

```
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# antivirus_scan /usr/bin/antivirus_scan
------------------------------------------------------------------
                        SCANNING Started
------------------------------------------------------------------
                Scanning file /usr/bin/antivirus_scan
                Result:Clean File

------------------------------------------------------------------
                        SCAN SUMMARY
 ------------------------------------------------------------------
                Total Files Scanned=1
                No virus files found
root@ubuntu:/usr/src/linux-headers-4.2.0-27-generic/LinuxAntivirus# 
```

Who Did What

Tasks	Done by
Antivirus Module, Overall design and Brainstorming	All
Netlink Socket: Kernel and User communication.	Pratik
User program for on demand access	Vishwajit
Check whether the file contains blacklist content	Nehil
Hijacking of 4 different system calls : open, openAt, execve, exceveAt	Tushar
Check whether the file contains whitelist content	Vishwajit and Pratik
Send Notify to show a bubble to the user informing about the virus	Nehil and Tushar
Scripts for installation and environment setup	Vishwajit
File rename and permission handling	Pratik and Tushar
Flowchart creation in Visual Paradigm	Nehil
PPT and Report	All
Testing	All
Check for memory leakage, code commenting	All

Citations

http://lxr.free-electrons.com/

http://man7.org/linux/man-pages/man3/nftw.3p.html

http://man7.org/linux/man-pages/man7/netlink.7.html

http://man7.org/linux/man-pages/man2/open.2.html

https://linux.die.net/man/2/openat

http://man7.org/linux/man-pages/man2/execve.2.html

http://man7.org/linux/man-pages/man2/execveat.2.html

http://man7.org/linux/man-pages/man3/exec.3.html

https://tnichols.org/2015/10/19/Hooking-the-Linux-System-Call-Table/

http://stackoverflow.com/questions/3221170/how-to-turn-a-hex-string-into-an-unsigned-char-array