# Introduction to Information Security - HT2020

Assignment 2 - Group 56:

Tzeni Bolena

Magd Khalil

Xavier de Verdun

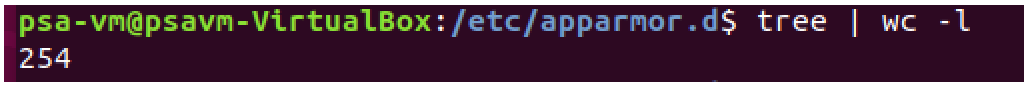
# CONTENTS

## 1. Design

For this project, we had to configure an AppArmor profile for a text editor. AppArmor was a new tool for us, so we decided to proceed gradually.

The first step was to read the documentation about the AppArmor tool. As you can see in the references part, most of the documentation is available online. Ubuntu Foundation Wiki and AppArmor Gitlab were useful to understand the rules of AppArmor and the given possibilities.

Then we tried to install AppArmor on the given operating system, a Kali Linux virtual machine. As it was not successful and made us lose lots of time, we decided to move on a Ubuntu 20.04 virtual machine. We completed the installation of AppArmor and used the additional package named "apparmor-profiles". This package is useful because it provides many profiles to discover in the folder /etc/apparmor.d.



```
psa-vm@psavm-VirtualBox:/etc/apparmor.d$ tree | wc -l
254
```

We used those profiles to have examples for the theory given by the Ubuntu Wiki and other references.

Then it was time to proceed to the profile creation. We chose to create a profile for the Gedit text editor. It is an editor providing a GUI similar to software like SublimeText or NotePad++.

The profile created can be found in Appendix A.

In the interest to understand the profile, it is important to understand that AppArmor, by default, denies everything. Any software will be allowed to do whatever is specified within the AppArmor profile associated.

We made the design of the file to regroup the lines by their purpose. The security enforcement level is high.

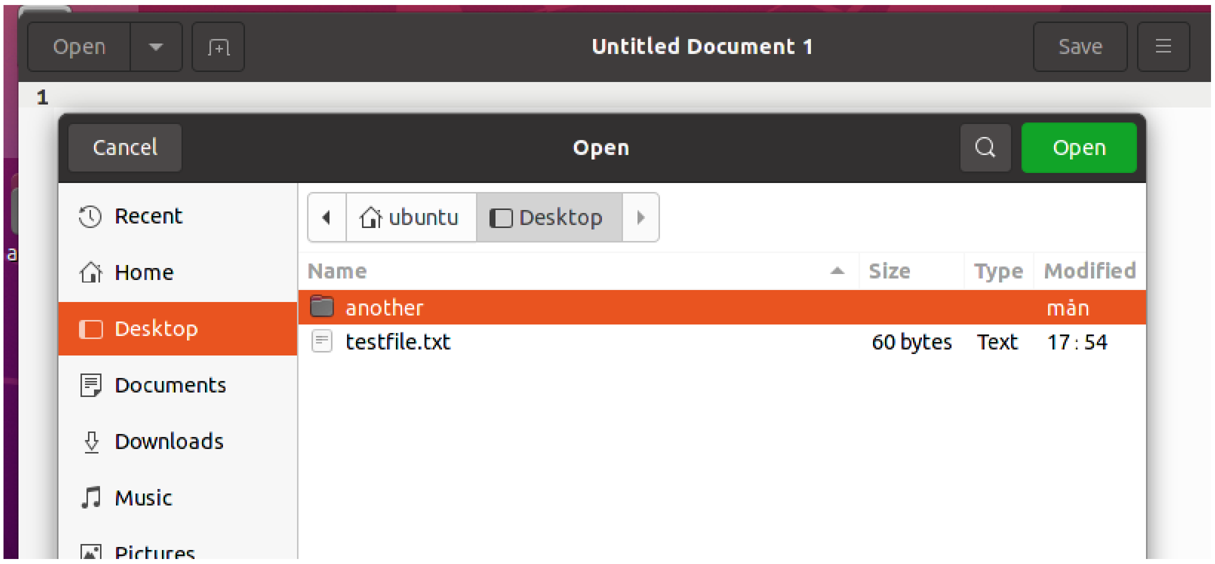
First of all, the file start with a couple of #include lines. It makes us able to avoid repetition using already existing low to mid-level security profiles. The next part is about configuring directory exploration. Gedit allows the user to explore a directory with its included FileExplorer, in order to restrict the area of exploration, the FileExplorer has been limited to the HOME directory and its subdirectories. Gedit is allowing a user to save its modifications. We decided only to grant Gedit to save modification in the HOME directory and its subdirectories. Then, Gedit access to specific files has been restricted. Finally, the last lines are enabling Gedit to access reading some folders so it can run properly.

## 2. Experiment results

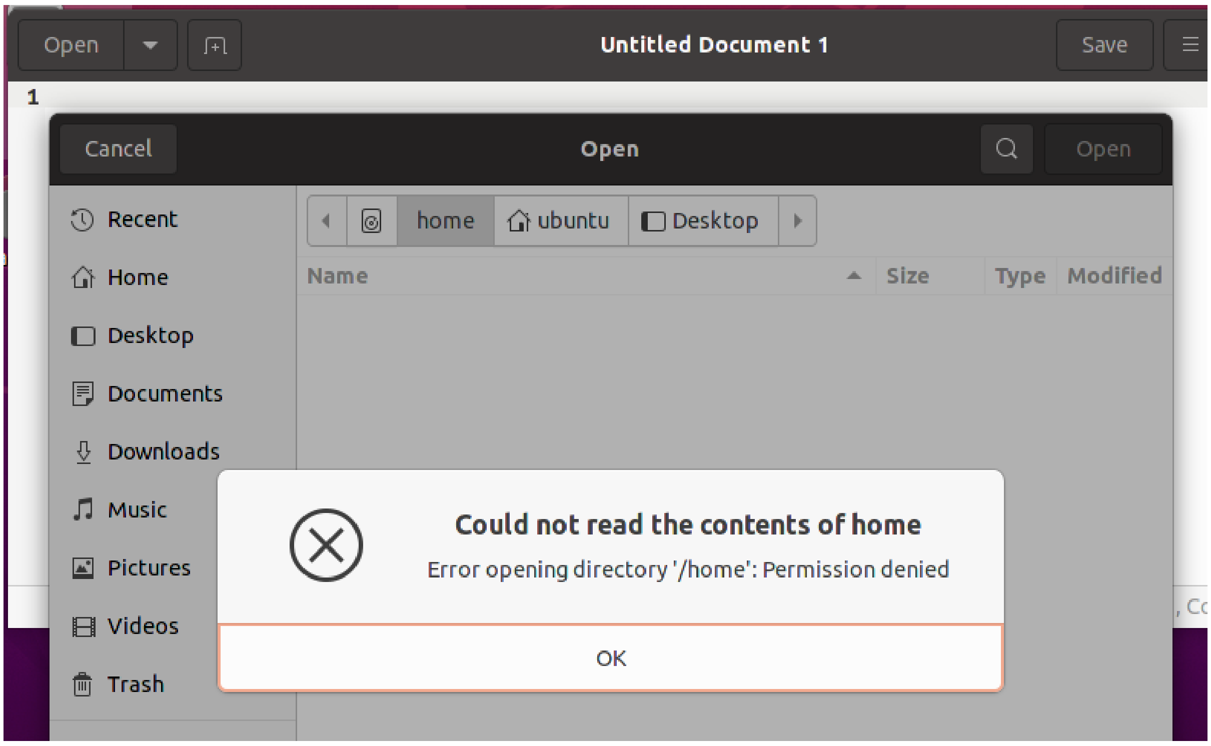
This chapter will be describing the results of the performed experiment. To assess whether the experiment was successful, we will first create a number of tests, which we will try to pass. The following list contains the tests:

1. See folder & Files: When using Gedit “open file” function, we must be able to see directories and files
2. Cannot access directories and files, that gedit does not have privileges for
3. Cannot edit files, if only read-privileges are granted
4. Can read and save files, when read and write rights are granted
5. Gedit can create a lock on the file
6. gedit starts up without throwing an error (i.e. due to missing fonts, no access to libraries or something similar)

After creating the profile, using the method described in chapter 1, all these tests were passed. Gedit was able to startup without any problems, and AppArmor did not log any failed accesses. The following figures show that the first and second tests were passed. While files within the user's home directory can be edited or at least viewed, this is not possible for files that are not in said directory.



| Name | Size | Type | Modified |
|---|---|---|---|
| another | | | mån |
| testfile.txt | 60 bytes | Text | 17 : 54 |

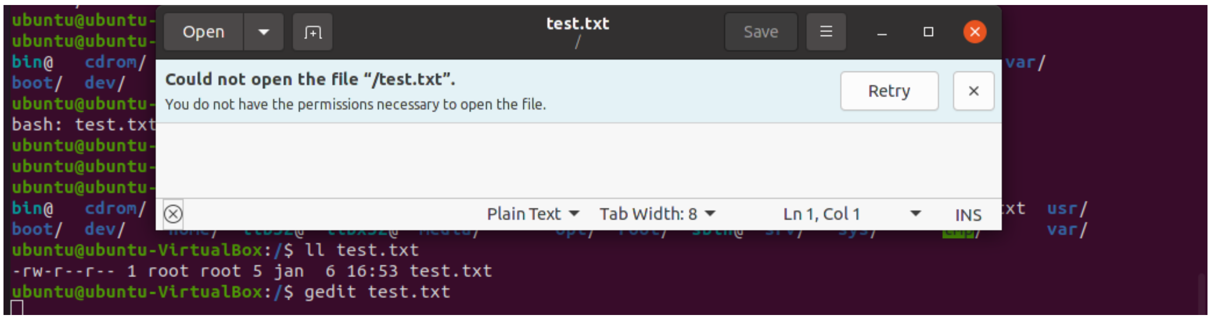


**Could not read the contents of home**

Error opening directory '/home': Permission denied

OK

When trying to open a file that the user / gedit does not have access for, gedit outputs an error message as shown in the following figure.

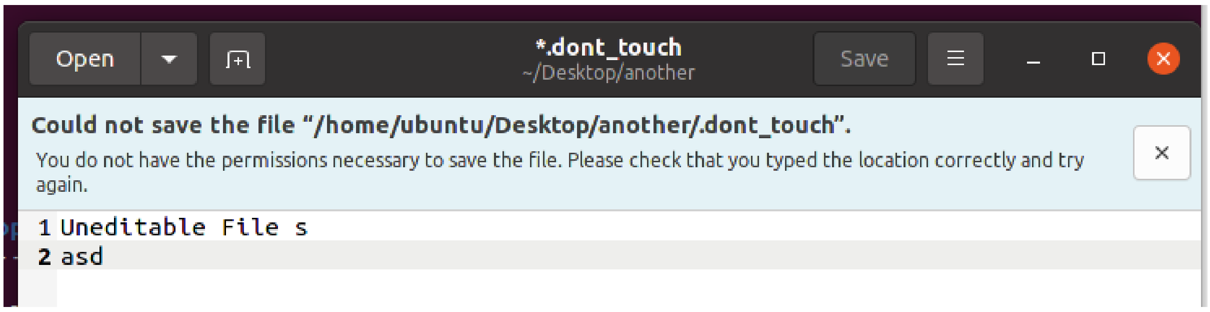


**Could not open the file "/test.txt".**

You do not have the permissions necessary to open the file.

```
ubuntu@ubuntu-VirtualBox:/$ ll test.txt
-rw-r--r-- 1 root root 5 jan  6 16:53 test.txt
ubuntu@ubuntu-VirtualBox:/$ gedit test.txt
```

When then trying to edit a file, for which we permitted to read but not write privilege, like sensitive or hidden files that start with a dot, we would get the expected behaviour:



**Could not save the file "/home/ubuntu/Desktop/another/.dont_touch".**

You do not have the permissions necessary to save the file. Please check that you typed the location correctly and try again.

```
Uneditable File s
asd
```

## 3. Time summary

Some parts of the work were split. When necessary (which was quite often) work was done by everyone together, via zoom and in-person meetings.

| 24/12 | 26/12 | 27/12 | 29/12 |
|---|---|---|---|
| - Reading of the assignment.<br>- Deciding upon the task to choose. | - Learning more about AppArmor.<br>- Reading documentation.<br>- Getting a better understanding of profiles. | - Starting with Kali Linux. Lots of work put down to get App Armor work on it.<br>- Due to it(Kali) having issues(probably because it is an old version), we decided to move on and use Ubuntu. | - Work on Ubuntu.<br>- Download necessary Utilities.<br>- Observe the already existing profiles.<br>- Create a basic and simple profile. |

| 30/12 | 3-4/01 | 5-7/01 |
|---|---|---|
| - Working on the profile by adding more “constraints” and experimenting with it. | - Finalising the profile and testing it(experiment results). | - Writing the final report.<br>- Reflecting on the profile and its results. Making minor changes were considered necessary in the profile. |

## 4. Reflections

Working with AppArmor gave us a better and more detailed understanding of enhancement of the operating system security. AppArmor provides the opportunity to restrict the actions that processes can take and therefore enforces better security. As a MAC(mandatory access control), AppArmor can and does protect objects(files, programmes etc.). It is flexible and has a fine granularity.

The insights that we gained into AppArmor have shown us that it is indeed a very powerful tool. Furthermore, it is easily understandable, even if one has no prior knowledge of AppArmor, as long as one has basic Linux understanding. Since it uses separate files for each program and allows for abstractions, it is also very well structured. Abstractions are profiles that can be included in other profiles and solve common problems and privilege requirements. One example of an abstraction is the file `/etc/apparmor.d/abstractions/ubuntu-browsers.d/user-files`, which allows the process to access the files that belong to the current user.

While reading the documentation of AppArmor (mostly the man page), we noticed that it is very well written and includes all information needed to create a profile. It has sections for all major points (e.g. globbing, macros, meaning of AppArmor specific names etc) and also includes examples, which helped us a lot.
Creating the profiles manually, as we did it, will help in gaining a better understanding of AppArmor's functionality. This is however not necessary. Using `aa-genprof`, `aa-logprof` and `aa-cleanprof`, we can create profiles more or less automatically. While genprof uses an interactive process to help the user to set up the profile, logprof checks the Kernel or AppArmor logs for events and gives the user the chance to add the correct privileges for these events. Cleanprof cleans the profile. This includes removing duplicate rules and restructuring the document. It must be done carefully, as it also removes the comments.
Testing the profiles was also rather easy because AppArmor provides different modes: the enforce and the complain mode. As the name suggests, one enforces the rules, while the other one will allow access to everything, but complain about it (i.e. writing to the log). We mainly used the enforce mode, as this made it very clear when something was not working. We would then check the log-files after performing our tests, to see if anything was printed. We also tried the complaint mode, as till will complain even when the audit flag was not set.

Like any other software tool, AppArmor along with all its positives brings some negatives. Some of them being more serious than others.
When it comes to setting up AppArmor some steps have to be taken and therefore to be done manually. The same rule applies to the maintenance of it. Nevertheless, this can be done fairly easily and is not a major factor for stopping someone to engage with it, but can get quite annoying.
A serious issue that can be phased is that new versions of third-party software might not be compatible with the last AppArmor profile. The new version might try to access directories, for which it does not have privileges.
It is good to keep in mind and realize that AppArmor cannot assure 100% security of the system. As any other security system AppArmor can be exploited. As a piece of software, it may contain bugs and vulnerabilities that can lead to weird behaviour and be taken advantage of by attackers. That is also indicated by the fact that newer versions are developed and released continuously. Also older versions might have gotten cancelled(version 2.14). Unexpected behaviour must not be ruled out. An example of that is "poking holes in apparmor profiles" where software could be run under another username, as long as there was no privilege escalation.

Overall AppArmor provides a good additional layer of security in the operating system. After using it during this project, we got to appreciate its capabilities and its high learning curve. In conclusion it is a very powerful security tool.

## 5. References

Debian, ed. 2020. *App Armor / HowToUse*. N.p.: Wiki Debian. https://wiki.debian.org/AppArmor/HowToUse.

Endeavouros. 2020. “Fail to enable AppArmor.” Forum Endeavouros. https://forum.endeavouros.com/t/fail-to-enable-apparmor/7627.

Johansen, John. n.d. “App Armor Documentation.” AppArmor / apparmor Gitlab. https://gitlab.com/apparmor/apparmor.

OpenSuse. 2018. “Building Profiles from the Command Line.” OpenSuse. https://doc.opensuse.org/documentation/leap/archive/42.3/security/html/book.security/cha.apparmor.commandline.html#sec.apparmor.commandline.build.

Ubuntu. n.d. http://manpages.ubuntu.com/manpages/xenial/man8/aa-cleanprof.8.html.

Ubuntu. 2019. “Man Page - App Armor.” Man Pages Ubuntu. http://manpages.ubuntu.com/manpages/bionic/man5/apparmor.d.5.html.

Ubuntu | Canonical. 2020. “Security - AppArmor.” https://ubuntu.com/server/docs/security-apparmor.

Wiki Archlinux Team. 2020. “AppArmor.” Wiki Archlinux. https://wiki.archlinux.org/index.php/AppArmor#Custom_kernel.

Azimuth Security
http://blog.azimuthsecurity.com/2012/09/poking-holes-in-apparmor-profiles.html.

## 6. Appendix

### A. AppArmor profile for gedit

```
#============================================================
# Assignment 2 in the IntroSec-Course
# HT2020, Stockholm University
#    Group 56
#    Members: Tzeni Bolena, Xavier de Verdun, Magd Khalil
#
  # The files/directories must be in the users HOME
  include <abstractions/user-write>

  # Acess to the user tmp and the global tmp
  include <abstractions/user-tmp>
#==============================================================

# incude global tunables, such as $HOME etc
include <tunables/global>

profile usr-bin-gedit /usr/bin/gedit {

  # This provides a base, which allows us to keep this file
shorter
  # The first one is the general base class
  include <abstractions/base>

  # grants read/write privileges for files and directories
owned by the user
  # This is needed to allow reading files in the home
directory
  # Allows to read directories recursively
  @{HOME}/** r,

  # This is needed for saving files in the home directory
  # k allows the editor to lock the file0...
  owner @{HOME}/** rwk,

  # Then revoke access to any file in the user directory that
starts with a dot
  # k for locking files, as needed by the editor
  audit deny @{HOME}/**/.* wklm,


  # Also allow access to the users' cache
  owner @{HOME}/.cache/* rw,
  # Since newer versions of ubuntu, running programs might
save their files into another user-specific directory within
/run/user, therefore allow rw access there as well
  owner /run/user/[0-9]*/** rw,

  # This is the same as above does however block access to
some dangerous files
  # That should not be changed by regular users.
  include <abstractions/ubuntu-browsers.d/user-files>


  # allow SOCK_STREAM connection
  unix (connect, receive, send) type=stream,

  # Allow access to the user-shared files (subdirectories and
files)
  # These include things like the theme, volume monitor and
more
  /usr/share/** r,

  # GTK is a toolkit for cross-platform ui-elements
  # We will allow read access to the settings file.
  /etc/gtk-3.0/settings.ini r,

  # Allow read access to the configuration files
  owner @{HOME}/.config/** r,
  owner @{HOME}/.local/** r,

  # Allow reading the system fonts
  # and font cache
  # Note: If noticing any problems, import the fonts
abstraction instead
  /etc/fonts/* r,
  /etc/fonts/fonts.conf/* r,
  /var/cache/fontconfig/* r,
  /home/ubuntu/.cache/fontconfig/* r,

  # We could not find out why this is needed and therefore
blocked it.
  # In case this causes a problem, we audit it, because it
will make it easier to find the root of the problem
  audit deny /etc/machine-id r,

  # Allow all DBus communications
  include <abstractions/dbus-session-strict>
  include <abstractions/dbus-strict>
  dbus,
}
```