Implement a Security Attack and Penetration Test System

In this assignment you will learn how to attack a reference monitor. The reference monitor you will be testing uses the security layer framework (encasement library, etc.) for the Seattle testbed. It is possible to do this assignment separately however it is recommended that this assignment be completed after SetMaxFileSizePartOne. Either way you should already have a working security layer or access to one. Testing the security layer is done by running a series of test cases that an adversary may use to circumvent your system.

Table of Contents

Overview

In this assignment you are a tester. You have been sent a bunch of reference monitors that need testing before they are deployed. Your job will be to test security layer with different test cases that try to exceed the file size beyond permission. If you are able to do so, then the security layer is not secure. The future of the system depends on your ability to test code thoroughly!

Three design paradigms are at work in this assignment: accuracy, efficiency, and security. Your test cases should test all three of these.

- Accuracy: The security layer should only stop certain actions from being blocked. All other actions should be allowed. For example, If a user tries to read and write within size limit, then he should be allowed to do so.
- Efficiency: The security layer should use a minimum number of resources, so performance is not compromised.
- Security: The attacker should not be able to circumvent the security layer. Hence, if the data can be written past permitted file size, the security is compromised.

You will submit a zip file containing all of the tests you have created. You will gain points for every student's reference monitor you find a flaw in. It is okay if multiple tests of yours breaking a student's reference monitor, but you will not gain additional points.

Prerequisites

This assignment assumes you have Python2.5 or Python2.6, Repy and RepyV2 installed on your computer. If you don't already have them please go to SetMaxFileSizePartOne for a tutorial on how to get them.

Helpful links

The following links will aid students in becoming comfortable with Python, Repy and seattle:

- Python tutorial: **http://docs.python.org/tutorial/**
- Seattle tutorial: **https://seattle.poly.edu/wiki/PythonVsRepy**

- list of RepyV2 syntax: **RepyV2API**

Testing security layers

Examples of tests

Test cases are briefly described in SetMaxFileSizePartOne and RepyV2SecurityLayers. Below is another example of a test case you may want to consider. This test case gives the right 'style' for your all your test cases, but lacks in the number of test cases. A good attack will include many test cases.

Test case 1:

```
if "look.txt" in listfiles():
  removefile("look.txt")
myfile=openfile("look.txt",True)  #Open a file

#write something
myfile.writeat("Test",0)

#set maximum file size allowed to 10
myfile.setmaxfilesize(10)

#the following writeat offset and data is within filesize limit, so security layer shoul
try:
  myfile.writeat("Testing",2)
except SeekPastEndOfFileError:
  log("Security layer failed because it blocked a valid write operation \n")
else:
  #user is allowed to write if its at valid place
  pass


#Close the file
myfile.close()
```

Code analysis

It is important to keep in mind that only lowercase file names are allowed. So in the above code, specifically:

```
# Open a file
myfile=openfile("look.txt",True)
```

look.txt is a valid file name, however Look.txt is not. Examples of other invalid files names are, look@.txt, look/.txt, and look().txt. Essentially all non-alphanumeric characters are not allowed.

In this code first we write something in file and then set the size of file to 10 using setmaxfilesize() method. Then we try to write again within valid offset and file size range. If security layer doesn't allow us to do and raises error for a valid write operation that it is failed.

Test case 2:

```
if "look.txt" in listfiles():
  removefile("look.txt")
myfile=openfile("look.txt",True)  #Open a file

#write something
myfile.writeat("TestCode",0)

#set maximum file size allowed to 10
myfile.setmaxfilesize(10)

#the following writeat will try to write a data that starts within permitted size but go
myfile.writeat("CheckPassOrFail",4)
data = myfile.readat(None,0)
if data == 'TestCheckP':
  #security layer properly truncated extra characters
  pass
else:
  log("file size compromised")

#if we set the offset to 10 then it should raise error. User can try that too
```

```
#Close the file
myfile.close()
```

More information on: Try, Except, Else, Finally

The try, except, else and finally statements are part of **exception handling**. For more information on exception handling please visit:

- http://docs.python.org/tutorial/errors.html
- http://wiki.python.org/moin/HandlingExceptions
- http://www.tutorialspoint.com/python/python_exceptions.htm

Hints and Ideas for testing

When writing your own tests it is important to test for a complete set of possible penetrations. Keep in mind, it only takes one test case to break through a security layer. Some of the things you may want to test for include:

- threading
- writing to multiple files

And more! Remember a good security layer can't be broken by anyone! Which is all part of the fun! It's about solving a puzzle. First you make the puzzle - write the security layer, then you solve the puzzle - try to bypass it. If your puzzle is "good enough", no one will be able to break it, no matter what.

Notes and Resources

- The following link is an excellent source for information about security layers: http://isis.poly.edu/~jcappos/papers/cappos_seattle_ccs_10.pdf
- In repy 'log' replaces 'print' from python. Many students find this to be a stumbling block.

Important Note

Your Test Case should not output anything if the security layer passes it. It should print appropriate error message only if the security layer fails.

Extra Credit

Find bugs in the extra credit reference monitors given the altered threat model. You should include more test cases in the extra credit!

How to run your tests on many reference monitors

If you are using Mac or Linux, you can do something like the following:

Put all security layer and attack cases in the same directory you were using before to run single security layer and attack program.

Then you can type the following in the bash shell to execute the testcases with the reference monitors:

```
for referencemonitor in reference_*;  do for testcase in netid_*.r2py; do python repy.py
```

What to turn in?

- Turn in the test cases used to attack a given reference monitor in a zip file. The name of each testcase must start with your net id in lowercase. For example: us341_securitytest1.r2py justincappos_goodaccuracytest.r2py are both valid names.
- Optionally turn in the test cases used to attack the extra credit reference monitors in a zip file. Note that in this case, you can expect that your code is run more than once. In the name of the

file, say if it needs to be run multiple times. For example: jcappos_run_twice_metadata_removal.r2py jcappos_run_once_threading_hack.r2py.