6.858 Fall 2014 Lab 4: Attacking server isolation

Handed out: Lecture 13
All parts due: Two days after Quiz 1 (5:00pm)

Introduction

(Note: this lab was given out during this course's initial run, and the instructions will not apply for users accessing this page for the first time. This is meant as a snapshot of the class as it originally progressed.)

In this lab, you will be attacking zoobar's privilege isolation and sandboxing. You will download the source code for the lab 2 submissions of two other students, which include privilege separation and sandboxing for Python profiles, and examine their code for possible vulnerabilities.

For each submission, you will deliver a text file, `lab4-code{0,1}.txt` respectively, that should contain your analysis of the zoobar server's security, including any possible weaknesses, and potential exploits (in Python) for vulnerabilities you have uncovered.

To get started, download the source code for each of the two zoobar sites you will be attacking. Copy the `lab4.tar.gz` file into your virtual machine (e.g., using `scp`, not shown below), and extract it in the home directory:

```
httpd@vm-6858:~$ mkdir lab4
httpd@vm-6858:~$ cd lab4
httpd@vm-6858:~/lab4$ tar -zxvf ~/lab4.tar.gz
...
```

Now, build and run this zoobar site, as shown below.

```
httpd@vm-6858:~/$ cd lab4/code1
httpd@vm-6858:~/lab4/code1$ make
cc -m32 -g -std=c99 -fno-stack-protector -Wall -Werror -D_GNU_SOURCE   -c -o zookld.o zookld.c
cc -m32 -g -std=c99 -fno-stack-protector -Wall -Werror -D_GNU_SOURCE   -c -o http.o http.c
cc -m32  zookld.o http.o  -lcrypto -o zookld
cc -m32 -g -std=c99 -fno-stack-protector -Wall -Werror -D_GNU_SOURCE   -c -o zookfs.o zookfs.c
cc -m32  zookfs.o http.o  -lcrypto -o zookfs
cc -m32 -g -std=c99 -fno-stack-protector -Wall -Werror -D_GNU_SOURCE   -c -o zookd.o zookd.c
cc -m32  zookd.o http.o  -lcrypto -o zookd
cc -m32 -g -std=c99 -fno-stack-protector -Wall -Werror -D_GNU_SOURCE   -c -o zooksvc.o zooksvc.c
cc -m32  zooksvc.o  -lcrypto -o zooksvc
httpd@vm-6858:~/lab4/code1$ sudo rm -Rf /jail
httpd@vm-6858:~/lab4/code1$ sudo make setup
[sudo] password for httpd: 6858
./chroot-setup.sh
+ grep -qv uid=0
+ id
...
httpd@vm-6858:~/lab4/code1$ sudo ./zookld
zookld: Listening on port 8080
...
```

Now that you have the zoobar code you need to review up and running, look at the code and understand how it works before proceeding to the next part.

Part 1: Code review

For the code review process, you do not need to comment on (or exploit) any buffer overflow vulnerabilities in the `zookws` web server from lab 1, or any attacks that involve a victim's web browser (such as exploiting a cross-site scripting vulnerability). The latter will be the focus of subsequent labs. Also remember to review both your `code0` and `code1` submissions. For each one, do the following exercises:

Exercise 1: Attack privilege isolation. Evaluate the security of the privilege isolation design in the zoobar code you are reviewing. Look for possible ways to violate the guarantees that privilege separation was supposed to provide. You may want to look back at the lab 2 description to review what the privilege separation was trying to achieve.

One possible approach may be to examine the RPC interfaces exposed by each service; are there ways to trick the RPC interface into performing an unintended operation? Another approach may be to examine the permissions on files in `/jail`.

Write down your review in `lab4-code{0,1}.txt`. Comment on any particularly good or bad aspects of the design. How did your design differ: was it any better or worse? For possible weaknesses, explain why they may be a bad design, even if you cannot immediately exploit them. For extra credit, develop working exploits that take advantage of any vulnerabilities you may have discovered. Include the Python code for any exploits you developed in `lab4-code{0,1}.txt`.

Exercise 2: Attack credentials. Evaluate the security of the auth and bank services. In particular, check if hashing and salting is properly done, so passwords are reasonably secure, even if the database is stolen by an adversary. Also check whether tokens are properly verified by the bank service before a transfer happens.

Comment on the design in much the same way as for the above exercise. Write down your code review in `lab4-code{0,1}.txt`. For extra credit, include Python code for any working exploits you may have constructed in `lab4-code{0,1}.txt`.

Exercise 3: Attack the Python sandbox. Evaluate the security of the sandbox used to execute Python profile code. Try to look for ways in which code running in one Python sandbox may be able to interfere with the rest of the system, or with code from another user's Python profile.

Comment on the design as in the previous exercises. Write down your code review in `lab4-code{0,1}.txt`. For extra credit, include Python code for any working exploits you may have constructed in `lab4-code{0,1}.txt`.

Challenge! (optional) Use your concolic execution system to analyze the other students' code for possible vulnerabilities. For example, you may be able to hook up your concolic execution system to the RPC interface, to look for malicious inputs that might trigger bugs in an RPC server, or malicious responses that might come back from an RPC server and trigger issues in the client.

If you do this challenge exercise, append the results to `lab4-code{0,1}.txt` and write the word `CHALLENGE` somewhere in the file so that we know to look for it.

Submit the resulting code review by running **`cd ~/lab4 && make submit`**. The resulting `lab4-handin.tar.gz` will be graded.

You are now done with lab 4.

6.858 Computer Systems Security

Fall 2014