Module 2.1 - Basic and advanced buffer overflow lab assignments

Ali Abbasi, 3 September 2015

In these assignments, you will become familiar with the concept of a buffer overflow and you will learn how to exploit this vulnerability on the ARM platform.

You should first study the ARM architecture and gain some basic knowledge about overflow vulnerabilities. Then in the basic exploit you will exploit a basic buffer overflow. In order to pass the lab you must complete this basic exploit and the two assignments leading up to it to the satisfaction of the student assistants.

If you are already familiar with the basic techniques then you will have time left for a more challenging assignment in which you will also practice return oriented programming. If you complete the advanced exploit satisfactorily you will gain 0.5 bonus point towards your final test result for the OS part of module 2.1.

Put the answers to all the questions below in one PDF file and submit this via Black Board.

1. Study basic online resources (mandatory for all)

Watch this video and read the Wikipedia page to learn about ARM architecture:

- The ARM University Program, ARM Architecture Fundamentals
- Wikipedia Arm Architecture

Watch this video and read the following Wikipedia pages to learn about buffer overflow vulnerabilities and protection measures:

- Buffer overflow attacks explained with beer!
- Wikipedia Buffer Overflow
- Wikipedia Stack Buffer Overflow
  Wikipedia Address Space Layout Randomization

Read the following PDFs to become familiar with buffer overflow on the ARM architecture:

- Practical ARM Exploitation Lab Manual Preview
- Exploiting ARM Linux Systems An introduction

Question

Explain in less than 100 words:

(a) How a buffer overflow can be exploited to take control of the execution flow.
(b) How the ASLR technique can prevent such exploitation.
(c) How the ARM handles the arguments to a functions call as compared to the X86.

2. Disable ASLR on your raspberry pi (mandatory for all)

Address Space Layout Randomization (ASLR) is a protection mechanism that will prevent exploiting a buffer overflow; hence you must turn it off before you can begin your assignment. To do this, please enter following commands.

```
$ sudo su
$ echo 0 > /proc/sys/kernel/randomize_va_space
$ su pi
```

Question
Does this setting survive a re-boot? Explain your answer in less than 100 words.

3. Basic exploit (mandatory for all)
The application "BOF1" in the Download.zip that you obtained from the course web site is vulnerable to a buffer overflow attack. There is a function in the application that will execute the "cat /etc/passwd" command. The function is never called by the application, it is basically dead code. Your task is to force the application to call the function by redirecting the execution flow after a buffer overflow attack.

Question
Describe step by step what you did, including the scripts and code that you wrote.

4. Study advanced online resources
If you want to get the bonus point please take a look at the following videos and web pages to become familiar with Return Oriented Programming in ARM architecture

1. Return 2 Zero-Protection (ret2zp) technique introduced in Defcon 18
2. Non-Executable Stack ARM Exploitation Defcon 18 Presentation
3. Return Oriented Programming for the ARM Architecture by Zynamics
4. ARM Exploitation ROPMAP
5. ROP Gadget finder for X86 and ARM
6. Return-Oriented Vulnerabilities in ARM Executables
7. Free Online ROP Gadget Search

Question
Explain in less than 100 words:

(a) Why we cannot simply use the ret2libc technique for the ARM?
(b) How to find a gadget in libraries for your ROP Exploit?
(c) What the difference is between the simple ROP and the Ret2ZP technique?

5. Advanced exploit (0.5 bonus point)
There is a second application "BOF2" in the Download.zip, that is also vulnerable to a buffer overflow attack. This time there is no function hidden inside the application that we can exploit. However, we still want to execute the "cat /etc/passwd" command. To reach this goal you have to use ROP technique. You have to change the execution flow to an appropriate libc system function and execute the "cat /etc/passwd". To achieve that you must prepare the function argument register(s) to contain your command. You must find a function in the libc (or any other library) and use its instructions to prepare the appropriate registers for your exploit. The end result should be the execution of cat /etc/passwd command.

Question

Describe step by step what you did, including the scripts and code that you wrote.

Twente Hacker Club

If, after completing this assignment on security you think: "*I want to know more!*" then maybe the Twente Hacker Club (THC) is something for you.

THC is a group of bachelor and master students (currently about 8) who enjoy hacking. We usually meet every week, or every other week depending if we have an upcoming event. In these meetings one of the students chooses a subject and gives a presentation about it. There are no strict rules as to what topics are allowed or how the presentation should be held. Last year's subjects includes code Injections, Capture the flags, fuzzers, etc. The goal is simply to encourage each other to learn new stuff in a very relaxed atmosphere (also, it's a nice place to meet new people who share the same interest).

The THC also participates in International 'Capture the Flag' events, read more about a CTF that we recently participated in here: http://ictf.cs.ucsb.edu/

So if you think this sounds interesting, or you simply want to know more just ask Ali during one of the labs, or send him an email: a.abbasi@utwente.nl