OVS Assignment: Casper Hacker Wargame

Neline van Ginkel and Frank Piessens

1 Introduction

Low-level software vulnerabilities have existed for quite some time and are still a favorite among real-world hackers to break into systems. Erlingsson et al. [1] describe 4 classes of low-level vulnerabilities and a number of countermeasures to fend off attacks against these vulnerabilities. The goal of this assignment is to get some hands-on experience with finding, attacking and exploiting low-level vulnerabilities in the context of a hacker wargame.

A wargame is a live computer-system offering a hacker playground with a number of challenges or levels to be solved. Solving these levels is a technical challenge requiring intimitate security knowledge of computer systems and programming languages, good problem solving skills and a lot of creative thinking. Because of this, wargames can truly be considered hacker games in which individuals or teams can compete with eachother and showcase their skills.

For this assignment, we have created a wargame "casper" specifically suited to the task of exploring the low-level software vulnerabilities described in the paper. This game consists of 12 basic levels (casper0 to casper11), of which the first 4 (casper0 to casper3) are for demonstration purposes. The game also contains some advanced levels that are a slight variation on casper4 to casper11.

You can find more details about this game on its website at `http://casper.haxx.be`.

2 Assignment

Category	Level(s)
Start level	casper0
Introductory levels	casper1 and casper2
Level solved in example	casper3
Stack-based buffer overflow	casper4 and casper5
Advanced Stack-based	casper40-casper42 and casper50-casper52
Heap-based buffer overflow	casper6 and casper7
Advanced Heap-based	casper60-casper62 and casper70-casper72
Return-to-libc	casper8 and casper9
Advanced Return-to-libc	casper80-casper82 and casper90-casper92
Data-only vulnerability	casper10 and casper11

Table 1: Categories and their levels

The levels 4-11 of the casper wargame contain levels in the 4 categories outlined in [1]. These 4 categories, together with the levels that they contain, are listed in Table 1. The first three categories also have some advanced levels.

For the assignment, select 4 levels out of casper4-casper11, 1 level from each category, and exploit them. For the first three categories, choose one of the advanced levels of the corresponding base-level, and exploit them as well. For example, if you solved casper4, you need to choose one of casper40, casper41 and casper42. These advanced levels contain a flawed defense against some usual exploits for the base-levels. You will need to solve 7 challenges: 4 base-levels and 3 advanced levels.

3 Practicalities

This project is to be worked on individually and takes up 30 hours of your time. If you do not manage to finish in time, complete as many level-reports as possible. Focus on the base-levels first.

We expect 2 files as deliverables: a PDF report (report.pdf) and a tar-ball with exploits (exploits.tar.gz). Dutch-speaking students are allowed to write their report in Dutch.

3.1 Report

An example report discussing level 3 can be found in the About section on the website.

The report must be a single PDF file and contain 5 sections: an overview and 4 solutions, 1 per base-level. The overview should contain a table listing your 7 chosen levels, together with the password for each level and the total time spent on exploiting it. This time includes analysis, experimentation, exploitation and reporting.

For each base-level you solved, the following topics must be discussed in the report:

- What does the level do?
- What is the vulnerability?
- How did you exploit it?
- How do we use your exploit? This section should only contain notes in case your exploit does not follow the general guidelines (see the next section).
- What solution do you propose to remove the vulnerability?
- (if applicable) What advanced level did you choose?
- (if applicable) Why does your exploit for the base-level not work on the advanced level?
- (if applicable) How did you modify your exploit to evade the additional checks?
- (if applicable) Why does your exploit for the base-level still work despite the additional defense?

You are allowed to discuss the project with fellow students. It is not allowed to share code or exploits. If you use code you did not write yourself, mention the source in your report as well. Make sure you understand all code and text you submit.

3.2 Tar-ball with exploits

An example tar-ball with an exploit for levels 2 and 3 can be found in the About section on the website

The evaluation of these submitted exploits will be automated. Therefore it is extremely important that you follow these guidelines **precisely**:

- Use a tar-ball (.tar.gz) to send in your exploits. This tar-ball should be named "exploits.tar.gz" and contain a Makefile in its top-level directory.
- Do not include binary files in your tarball. If your exploits are written in C, then include the sourcecode and adapt the Makefile to compile them, instead of including the binary itself.
- The Makefile must have a target per level your have an exploit for. The target should be called "exploitX" if it builds and launches the exploit for "casperX". E.g. if you have exploited casper5, then "make exploit5" should build and execute your exploit for casper5 successfully.
- The endresult of the exploit is to spawn the `/bin/xh` shell. Careful! That is `/bin/xh`, not `/bin/sh`. Do **not** execute any other shell from your exploit. The **/bin/xh** shell will contain special logging code to indicate that your exploit works and log the success. If you are using another shell, no log will be written and we will assume that the exploit doesn't work. To help you, when you execute **/bin/xh**, a banner will be shown to indicate that you are using the correct shell. Look for that banner!
- Make sure the tarball unpacks fine with the command "tar -xzf exploits.tar.gz", and that the make targets work when executed on casper as user casper0. Login and out a couple times when testing your exploits, because they might be susceptible to changing environment variables.
- Finally, test your exploits.tar.gz file with the command: **casper_verify_tarball.py exploits.tar.gz**. This tool will verify that your tarball is valid for submission.

4 Material

Aside from the paper in the References section, the following materials may also be useful:

- Course notes, this assignment text, the example solution PDF report and exploits tar-ball.

- There will be a labsession to work on the assignment at 23 november (group A) and 24 november (group B).
- The course forums on Toledo for further questions.
- The website `http://casper.haxx.be` contains relevant reading material per level, as well as an About section with useful information.
- The source code for each level can be found in `/casper/`, once logged in to the casper server through SSH.
- Cheat sheets for many scripting languages, e.g. at `http://hyperpolyglot.org/scripting`, ...

References

[1] Ú. Erlingsson, Y. Younan, and F. Piessens. Low-level software security by example. In P. P. Stavroulakis and M. Stamp, editors, *Handbook of Information and Communication Security*, pages 633–658. Springer, 2010.