Teaching Security with CTF-like challenges

Cédric Lauradoux

Teaching Security with CTF-like challenges

Cédric Lauradoux
Univ. Grenoble Alpes, INRIA
Email: cedric.lauradoux@inria.fr

***Abstract*—We present a course given at ENSIMAG which is an introduction to security. The originality of this course is that challenges are given to the students during the labs and at the mock/final exams. It may give you ideas for your courses.**

I. Targeted audience

The course is followed by 60 students of the Embedded systems and connected devices program of ENSIMAG known as SEOC. This course is also the first time that the students are exposed to security. The background of the students is heterogeneous: some students have good programming skills but others have virtually no programming experience. This program is shared between ENSIMAG and Phelma which explain this variability on the students programming level. Due to this constraint, the course and the labs are restricted to command lines and bash scripts.

II. Course content

During 6 weeks, the course is organized as follows. On Wednesday, the students follow a 1.5 h lecture. Students are split into two groups for the labs (Group 1 Thursday) and (Group 2 Friday). The course is oriented on system security and addresses five topics: *file format, forensics, symmetric cryptography, asymmetric cryptography and password security.* The last week is dedicated to a mock exam to prepare the students to the final exam. The full content of the course is described in Table I and is accessible at https://bit.ly/2v3Tr7N.

Lecture 1: Intro. + File format	**Lecture 4:** Asymm. crypto
Lab 1: *fuzzing, archive bomb, decompression bomb, chameleon, werewolf.*	**Lab 4:** *hybrid encryption, signature, certificat.*
Lecture 2: Forensics + Hashing	**Lecture 5:** Password security
Lab 2: `sleuthkit`, *file system, carving, secure deletion, covert channels.*	**Lab 5:** *digest breaking,* `john`*, crawling, wget,* `cewl`*.*
Lecture 3: Symmetric crypto.	**Mock exam**
Lab 3: *mode of encryption, password, salt, fuzzing.*	**Partial Correction**

TABLE I
Main concepts introduced during the lectures and the labs.

A. Labs organization

Students have virtual machine at their disposal with the latest version of Kali Linux. The main softwares used during the class are: `dd`, `hexdump`, `openssl`, `exiftool`, `sleuthkit`, `foremost`, `strings`, `shred`, `john` (Jumbo), `wget` and `cewl`.

The labs are very short (1.5h), so their difficulty is progressive. During the first half hour, the students have direct illustrations of the softwares they will used. They just need to enter commandline and observe the results. In the next half hour, they will do one or two guided exercises. During the remaining time of the lab, the students have to solve challenges. The time distribution here is approximative.

Students start to do challenges at Lab 2: they need to use the skills from the previous labs to obtain the new one. It can be for instance decrypting a file to get the text, exploring a file system or breaking a digest to recover the password of the zipped text. These challenges are very basic to validate the fundamental knowledge of the students.

B. Motivations

The course is concentrated on the notion of file. Through my years of teaching, I realize that many of my students (even the best) who followed my *"more classical"* lectures on security were still confused by the notion of files. They have issues when they need to put into practice all the theoritical framework of security they have learned. This is particularly true with cryptography: the students have no issues understanding lectures on symmetric encryption and modes of operation, but when they have to encrypt a file they are in total disarray and make blunders. By centering everything around the file, it gaves me the opportunity to put the students in front of concrete situations they must solve.

C. Reading materials

The first lab is based on two articles on security of file format [4] and decompression bombs [3]. The textbook of Takanen *et al.* [6] on fuzzing is given to the students who want to explore this domain. Forensics and carving are explained in the second lab using references to Chapter 3 of [1] (the complete book is a great reference but this chapter is available for free). Lab 3 and 4 are dedicated to cryptography and have been inspired by exercises from [5], [7].

Finally, the last lecture on password security is based on an excellent tutorial given by Blase Ur and Michelle Mazurek at Usenix SOUPS 2016 (the slides are available at http://users.umiacs.umd.edu/~mmazurek/soups16_tutorial.htm). The only difference is that the lab is using `john` instead of `hashcat`.

III. Examples

We give two examples exercises or challenges that the students need to do/solve during the labs. The first example is the most original one on file format. The second example is on forensics with a final twist on file format.

A. Fuzzing file's metadata and chameleon

One of the main activity of the first lab consists to play with the `tarball` file format. Students discover the structure of the header of a `tarball` using the manual http://www.gnu.org/software/tar/manual/html_node/Standard.html. Then, they have to complete a draft of bash script. The goal of this script is to perform a crude fuzzing on the header of a `tarball` by injecting random values. The script is composed of 3 parts:

- injection of random data in a given field of the header (*using `dd` and done by the students*)
- computation of a valid checksum (*given to the students*)
- analysis of the result (*done by the students*)

Once their script is completed, they can use it on a valid `tarball` file. This very simple introduction to fuzzing shows them that some fields can be filled with random data without affecting the user's experience.

They also need to transform this script to read the header of a `tarball`. The file `covert.tar` is given to them and they can discover a message hidden in the `linkname` field. The conclusion of this exercise is the creation of a chameleon [4]: a file with multiple format signature. They inject a gzip'ed version of the `eicar.com` file into the filename field of innocuous `tarball` file `chameleon.tar`. The compressed file `eicar.com.gz` consists of 98 byte. A software can detect the file as a compressed `gzip` file or a `tarball` depending on its format detection logic. Students test the gzip'ed version the `eicar.com` and their chameleon file `chameleon.tar` on https://www.virustotal.com. The result observed during the labs is 43/56 anti-viruses detect `eicar.com.gz` as the `eicar.com` file but the result drops to 18/56 anti-viruses for `chameleon.tar`: 25 anti-viruses can be still abused by a chameleon.

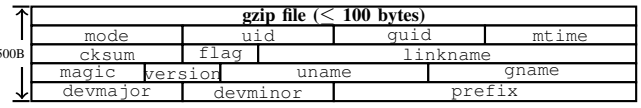

Fig. 1. Header of `tarball` chameleon.

The rest of the lab includes decompression bombs, archive bombs [3] exercices and more generally the exploration of more file's metadata using `exiftool`.

B. Forensics (and some pdf fun)

After taking some time to get familiar with the basic commands (`mmls`,`fsstat`,`fls` and `icat`) of the `sleuthkit`, students start to investigate a file named `system.dd`. It appears to be a FAT16 file system labeled `NOHOPEFORYO`.

Students recover all the `tar.bz2` files using `icat`. Then, the most adventurous students try to open `a.tar.bz2`. It is actually a compressed archive (it contains 401 files containing each 30MB of random data $\approx$ 12GB, to freeze their virtual machine). The smartest students have a look at the `$MBR` which says that openning `a.tar.bz2` is not a good idea. They turn to `b.tar.bz2` which is a temporary file resulting from creating `a.tar.bz2`. It contains only a few random meaningless files (it does not freeze the virtual machine).

Listing 1. First image

```
$ fls system.dd
r/r 3:  NOHOPEFORYO (Volume Label Entry)
r/r * 5:        b.tar.bz2
r/r 7:  c.tar.bz2
r/r 9:  a.tar.bz2
v/v 1604467:    $MBR
v/v 1604468:    $FAT1
v/v 1604469:    $FAT2
V/V 1604470:    $OrphanFiles
```

Finally, they decompress `c.tar.bz2` and obtain the image `new.fs`. This time, it contains a `ext2` file system. All the students think it is going to be easy money that it is as simple as `icat new.fs 12 > suite.pdf`. They are disappointed to obtain an empty file.

Listing 2. Second image

```
$ fls new.fs
d/d 11: lost+found
r/r * 12:       suite.pdf
V/V 7585:       $OrphanFiles
```

As explained during the lecture, carving (`foremost`) is always there when you have no hope. `foremost` is able to recover a broken `pdf` file. Most of the students have used online tools to repair the `pdf` files but it can be done using `mutool clean target.pdf repaired.pdf 1`.

IV. Exams

The exam is organized as a series of challenges that the students have to solve on a desktop. The goal of each student is to recover 7 student's logins following the course. The answers of each student are randomly chosen to secure the exam. For the mock exam, all the students share the same logins.

The challenges are organized using a tree structure (Fig. 2). Each edge of the tree is a challenge and each node is a file. The challenge connecting a node to its left child is simpler than the challenge connecting the right child. Solving successfully a challenge provides one login except for the challenge between File 1 and File 4. Reaching File 3 and its left child only requires to apply what was done during the labs: the student is granted 11 points for recovering 3 logins. Solving the other challenges provide extra points (+2 points and +3 for the most difficult challenge). It is challenging for the best students without putting aside the other students.

A. Sampled exercise: forensics + encryption

Students have received an image of FAT32 partition. If they mount the partition, they observe many files with extension `.jpg.enc`: the `file` command detects them as `data` file

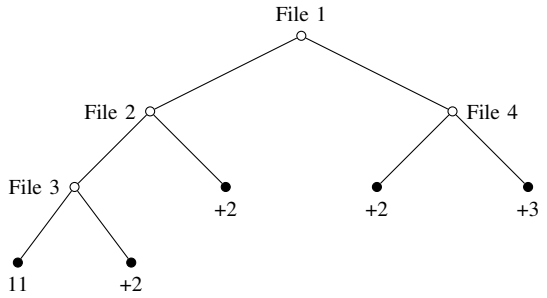

Fig. 2. Organization of the challenges in the mock and final exams.

and they look random. The files are meaningless at the first sight. After exploring the deleted files, students discover some evidence that some files were deleted using `srm` and a bash script named `ransomware.bash`. This basic malware encrypts all `jpg` files located in the current directory using `openssl` (AES in CTR mode with 128-bit key). At the beginning of the script, a key and an initiatization vector are generated using the command `whoami`. Most of the students recover each file by hand using `openssl`. The best students realize that running the malware again but changing the targets `jpg` by `.jpg.enc` is enough to recover the original files thanks to the nice properties of the CTR mode. The `jpg` files are pictures of students sampled from the students gallery. They can recover the corresponding logins but the issue is that only one of them is valid. They can discover that a digest is hidden in the unallocated space of the partition using the `strings` command which matches one `jpg` file. I admit that this last step is tricky but it aims at reminding them that `strings` is a powerful tool.

Students often realize that writing a ransomware is difficult: *how to generate the key and the initialization vector ? How to ensure that the targeted user is unable to recover the key ? How to make the ransomware creator get the key ?*

V. Feedbacks

A. Teacher

With the labs reaching a stable version, most of the work in the course is done. There are two demanding parts: verifying compatibility issues and creating the mock and real exams.

In the first year this course was given with this agenda and content, it was also the time of major revision for `openssl`. I was using my computer to generate the challenges using a different version of `openssl`, the one used by the students. It creates lots of compatibility issues which are now handled properly by `openssl`.

Sacrifying a week (lectures + one lab for each student) to make a mock exam is valuable for the students otherwise they have difficulties to combine all the skills acquired. The side effect is that you need to create two original exams every year.

B. Students

This is the second year that this course has followed this format. ENSIMAG provides a survey to the students to grade their course. 25 students commented the course and 24 rate it on a scale between Excellent and Bad (see Table II).

Excellent	Good	Satisfactory	Unsatisfactory	Very unsatisfactory	Bad	No answer
9	8	4	3			1

TABLE II
Evaluation of the course by the students.

Most of the comments are extremely positive. Two negative reviews consider the lectures useless and would prefer to have only labs. This comment is also shared in positive reviews: the students would prefer to have more labs than lectures. The last negative review is about my own personality and not the content of the course: it is not relevant here.

VI. Conclusion

How to improve the course ? First, it was asked to transform the lectures into labs according to the wishes of the students (I actually agree with them). With 3 extra labs, the goal could be to add two networking labs and one lab on malwares and host-based intrusion detection system (`yara+inotify-tools` for instance). Another possible direction is to use the extra labs to make an in-depth forensics analysis of Windows or Linux system. The security of the exam could also be improved by randomizing the order of the challenges for each student (today only the answers are randomized).

Acknowledgment

The author wants to thank Jean Louis Roch for giving him the opportunity to teach at Ensimag and for giving him *carte blanche* for the content of the course. This work is supported by the French National Research Agency in the framework of the *Investissements d'Avenir* program (ANR-15-IDEX-02).

References

[1] Cory Altheide and Harlan Carvey. *Digital Forensics with Open Source Tools*. Syngress Publishing, 1st edition, 2011. Chapter 3 available for free at https://booksite.elsevier.com/samplechapters/9781597495868/Chapter_3.pdf.

[2] Gildas Avoine, Pascal Junod, Philippe Oechslin, and Sylvain Pasini. *Sécurité informatique, cours et exercices corrigés*. Vuibert, 2015.

[3] Margaux Canet, Amrit Kumar, Cédric Lauradoux, Mary-Andréa Rakotomanga, and Reihaneh Safavi-Naini. Decompression Quines and Anti-Viruses. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, CODASPY 2017*, pages 23–34. ACM, March 2017.

[4] Suman Jana and Vitaly Shmatikov. Abusing File Processing in Malware Detectors for Fun and Profit. In *IEEE Symposium on Security and Privacy, SP 2012*, pages 80–94. IEEE Computer Society, May 2012.

[5] Douglas Stinson, Serge Vaudenay, Gildas Avoine, and Pascal Junod. Cryptographie - Théorie et pratique / 2ème édition. 2003. French translation of: Cryptography Theory and Practice, 2nd edition, Chapman & Hall/CRC, 2002.

[6] Ari Takanen, Jared DeMott, and Charlie Miller. *Fuzzing for Software Security Testing and Quality Assurance*. Artech House, Inc., Norwood, MA, USA, 1 edition, 2008.

[7] Damien Vergnaud. *Exercices et problèmes de cryptographie*. Sciences Sup. Dunod, 2012.