Homeworks

Computer and Network Security

Emilio Coppa

Basic rules

- 7 homeworks released during the course
- submit solution for at least 4 homeworks to get a bonus of 2 points in the final exam
- the bonus can be used only if your final grade is equal or higher than 24
- program valid until Sept 2021 (exam sessions: Jan, Feb, Jun, July, Sept); undergraduate/Erasmus/other master's students can participate
- Compilatio plagiarism detection, checking the similarity with web documents and with other consigned homeworks
- don't post or share to anybody the assigned homeworks

Homework submission

1. Report (PDF):
   - Discuss in detail the solution to the homework (max 15 pages, no need to be verbose if not useful)
   - PDF generated using LaTex
   - title (homework name/ID), subtitle (CNS Course Sapienza), Author (first name, last name, student ID), date (deadline), no abstract
   - use article class, 11 pt
   - language: English
   - if pictures needed use \includegraphics
2. ZIP archive: a compressed .zip archive containing
   - source code of the solution
   - auxiliary scripts
   - source code of the report (latex files, bib files, figures, etc.)

Homework submission (2)

Submission form:

https://forms.gle/pnUWoWunRe1YxRaH9

(to fill the form use the institutional mail from Sapienza or, if you do not have one yet, a personal mail from Google)

Homeworks will be evaluated in a binary form: accept/reject

A reject is notified via mail and can happen at any time during the year (even during the exam where you plan to use the bonus).

Homeworks: deadlines (subject to changes!)

ID	Released	Deadline
1	06/10/2020	Dynamic deadline: up to 31/12/2020
2	20/10/2020	30/10/2020
3	30/10/2020	20/11/2020
4	30/10/2020	20/11/2020
5	20/11/2020	30/11/2020
6	27/11/2020	10/12/2020
7	08/12/2020	20/12/2020

Homework #1: solve two crypto challenges in a CTF

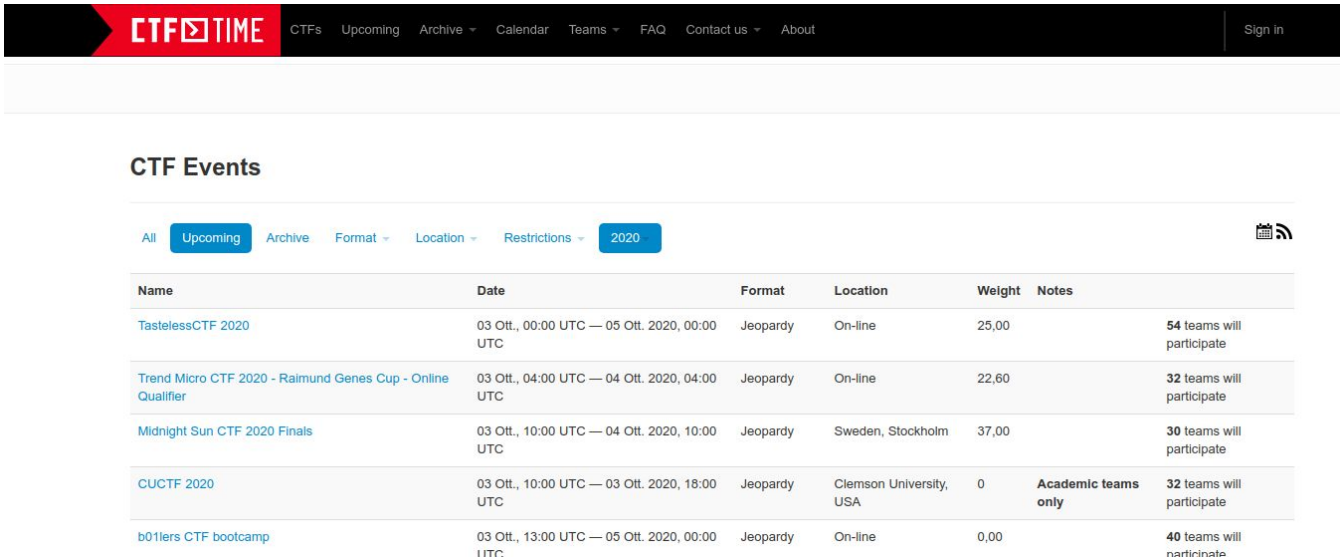
In computer security Capture the Flag (CTF), "flags" are secrets hidden in purposefully-vulnerable programs or websites. Competitors steal flags either from other competitors (attack/defense-style CTFs) or from the organizers (jeopardy-style challenges). Several variations exist, including hiding flags in hardware devices.

Security CTFs are usually designed to serve as an educational exercise to give participants experience in securing a machine, as well as conducting and reacting to the sort of attacks found in the real world (i.e., bug bounty programs in professional settings). Classic activities include reverse-engineering, network sniffing, protocol analysis, system administration, programming, cryptoanalysis, writing exploits

Homework #1: solve two crypto challenges in a CTF (2)

Rules:

- Choose one CTF jeopardy competition listed on CTFtime.org



Entry level CTF are OK.

Homework #1: solve two crypto challenges in a CTF (3)

- Register on CTFtime.org, register a team
- Register on the CTF website:
  - you should play alone using the team created on CTFTime.org
  - you should have confirmation email of the registration, showing your nickname and the event
- Play the CTF: solve at least two challenges that are “tagged” by the organizer as “crypto” or any other topic covered by the course (see syllabus)
- Write a write-up for each challenge and post it on CTFtime.org (see this FAQ)

Homework #1: solve two crypto challenges in a CTF (4)

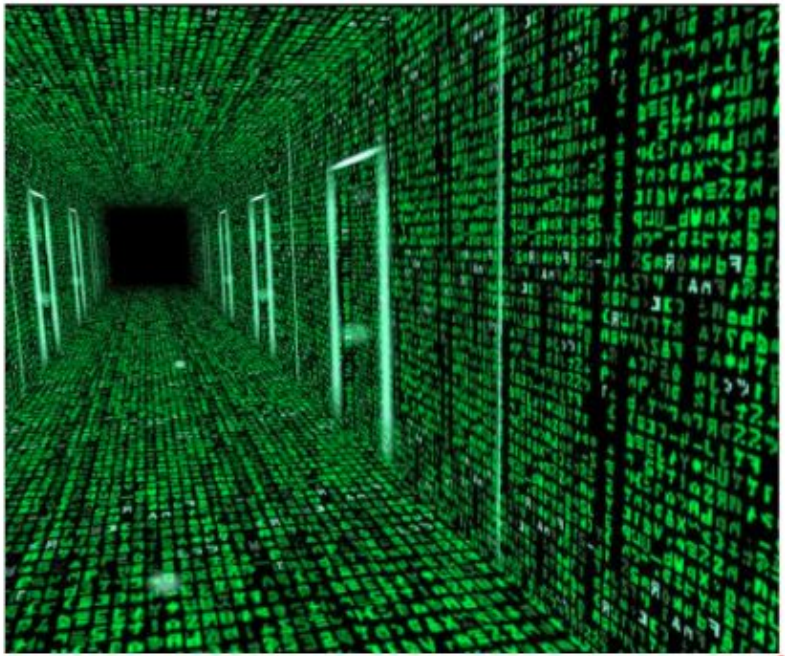
- Write a PDF report using LaTex, integrating the two write-ups, screenshot of the confirmation mail from the CTF event, and screenshot about correctness on the flag (points on the CTF website). At the beginning of the report, provide details about the CTF event: name, URL, URL on CTFTime.org, start date, end date.

- Submit as a homework #1:
  - the PDF report
  - a ZIP archive containing any source code/program/script related to the challenges

- Deadline: 48 hours after the end of the CTF competition

Homework #1: solve two crypto challenges in a CTF (5)

You think it will like this:



Homework #1: solve two crypto challenges in a CTF (6)

You think it will like this:

Homework #1: solve two crypto challenges in a CTF (7)

The goal of this homework is to have fun while learning crypto!

Playing CTF is the best way of learning cybersecurity!

Homework #2: AES implementation

Goals of the homework:

1. implementation of AES-128
2. extend implementation adding operation modes ECB, CBC, CFB, OFB, CTR
3. experimental comparison of your implementation wrt performance to a well-known implementation

The report should discuss how you have structured the implementation, motivate any design choice, and discuss the performance comparison.

Homework #2: AES implementation (2)

Implementation of AES-128:

- you can use any programming language (Python is suggested)

- different approaches (slower to faster, interesting to boring):
  - perform computations in GF(256) for SubBytes() and MixColumns(): you learn a lot but it will take a bit of time, you should use some library (e.g., SageMath/Python) to deal with GF.
  - use precomputed S-Box for SubBytes(), xtime() for MixColumns()
  - use precomputed S-Box for SubBytes(), precomputed lookup table for MixColumns()
  - single lookup table for all steps except roundKey()

Homework #2: AES implementation (3)

Implementation of AES-128:

- useful documents (Resource page in Piazza):
  - FIPS-197: Advanced Encryption Standard (AES): see examples in Appendix C
  - Example of MixColumn step
  - The Design of Rijndael by Joan Daemen and Vincent Rijmen: see section 4.1
- many public implementations of AES: it is ok to “look” at them to learn some details… but you should not do cut-and-paste code…
- you need to implement both encryption and decryption

Homework #2: AES implementation (4)

Extend implementation adding operation modes ECB, CBC, CFB, OFB, CTR:

- padding is needed: one easy approach is PKCS#7 padding, but also other approaches are fine (your choice).
- see document “Recommendation for Block Cipher Modes of Operation by NIST” on the Resource page in Piazza:
  - CFB can done in different ways (parameter s, choose one value)
  - check out examples in Appendix F

Homework #2: AES implementation (5)

Experimental comparison of your implementation wrt performance to a well-known implementation:

- choose one “standard” and “well-known” implementation of AES that provides at least three operation modes (3 out of 5: ECB, CBC, CFB, OFB, CTR). For instance, you can easily use PyCryptoDome (even if your implementation is not in Python!). Most languages have a library providing AES. OpenSSL (or LibreSSL) is another good option.

- try to validate the correctness of your implementation by comparing ciphertexts generated by the two implementations. Notice that the well-known implementation may use a different padding scheme and different “parameters”, hence it is fine if you do not get the same output for all modes but you should (try) to investigate the reason.

Homework #2: AES implementation (6)

Experimental comparison of your implementation wrt performance to a well-known implementation:

- perform an experimental comparison between your implementation and the the “well-known” implementation:
  - consider three different file sizes: 1KB, 100KB, 10MB
  - compare performance of different modes in encryption and decryption: a standard metric reported for ciphers is the throughput
  - use different charts/plots in the report to show the comparison

- It is OK if your implementation is way slower than the “well-known” implementation. The ultimate goal of this homework is to get an idea of how AES works, how different modes work, and have a bit of familiarity with one real-world AES implementation.