14-741/18-631: Homework 2

All sections Due: Tuesday October 7th, 2021 by 10:00am EST

Name:

Andrew ID:

Total (100 pts max):

Guidelines (Please read before starting!)

- Be neat and concise in your explanations.
- You must use at most one page for your explanation for each Problem (code you wrote may be on additional pages). Start each problem on a new page. You will need to map the sections of your PDF to problems in Gradescope.
- To access the CTF problems, create the SSH proxy as per the guide on Canvas.
- For CTF problems, **you must use the following format in your explanation**:
  - CTF Username
  - Flag
  - Explain the vulnerability in the program, and explain conceptually how that vulnerability can be exploited to get the flag.
  - How did you exploit the vulnerability? List the steps taken and the reasoning behind each step. The TA grading should be able to replicate the exploit following the steps. Feel free to make references to your code! **Note that *"use XYZ online solver"* is not sufficient - you must explain how the online solver derived the answer for full credit.**
  - Append your source code in the same writeup. Your source code should be readable from the writeup PDF itself. Note that this does not count towards the page count above.

  Omitting any of the above sections would result in points being deducted.
- Some questions are marked as **Optional Team Work**. For those, you can work with another student. The maximum team size is 2. In your write up, please write that "completed as a team", followed by the andrew ids of you and your teammate. **Individual CTF username and flag need to be put in the write up for CTF questions.** Please refer to the problem write up for requirements of whether to submit individual or one joint write up (this may defer from problem to problem).
- **References:** List resources outside of class material that helped you solve a problem. This includes online video tutorials, other CTF problems on other platforms, etc. Remember that source code available online (e.g. stackoverflow) also needs to be cited. A quick guide on citing source code can be found here: `https://integrity.mit.edu/handbook/writing-code`. **Omitting the references section may result in an Academic Integrity Violation(s).**
- It is highly recommended that you use Python for your assignment. You may use other languages that you are familiar with, but the teaching team will not be able to support or debug language specific errors.
- Please check your English. You won't be penalized for using incorrect grammar, but you will get penalized if we can't understand what you are writing.

- Proofs (including mathematical proofs) get full credit. Statements without proof or argumentation get no credit.
- There is an old saying from one of my math teachers in college: "In math, anything partially right is totally wrong." While we are not as loathe to give partial credit, please check your derivations.
- Write a report using your favorite editor. Note that **only PDF submissions will be graded.**
- Submit to Gradescope a PDF file containing your explanations and your code files before 1:29pm Eastern Standard Time on the due date. You can find the timing for EST here: `https://time.is/EST`. Late submissions incur penalties as described on the syllabus (first you use up grace credits, then you lose points).
- If you choose to use a late day, you do not have to inform the instructors. We will calculate the number of late days used at the end of the semester based on the time of submission on Gradescope.
- Post any clarifications or questions regarding this homework to Piazza.
- **General allowed team work** Beyond designated team questions, you are encouraged to shared resources (e.g., TA's help, online resources you found helpful); you are encouraged to set up virtual study sessions with your teammate(s) to check each other's progress and discuss homework assignment solutions.
- **This is not a group assignment. Beyond your teammate, feel free to discuss the assignment in general terms with other people, but the answers must be your own.** Our academic integrity policy strictly follows the current INI Student Handbook `http://www.ini.cmu.edu/current_students/handbook/`, section IV-C.
- Good luck!

1 Using PGP (25 points)

In class we learnt about public key encryption and signing, time to use it! This problem will require you to create a public/private OpenPGP key pair, using, for instance the GNU Privacy Guard (*gpg*), or similar package and share it on a key-server. **You must provide us the fingerprint for the key you upload to the keys.mailvelope.com server.**

Our email address is ta-14741-18631@gmx.com. The OpenPGP keyID is 0xE0B3FB258A91EC72, the fingerprint is D7290F5BC5560B50183E47AFE0B3FB258A91EC72, and the public key is available on `https://keys.mailvelope.com/`. GPG may require using the HTTP Keyserver protocol address for the mailvelope server: `hkps://keys.mailvelope.com/`.

1. Based on this information, how do you verify that the public key you got from the key-server is valid, i.e., that no one has modified it?
2. Complete the PGPChatbot Challenge on the CTF Server. Include your key's fingerprint in the write-up. **Important note:** Make sure to use your **CMU @andrew email address** while uploading your public key to keys.mailvelope.com for this problem.
3. Imagine you were interacting with us using email and had to send us an attachment, would you use PGP/MIME or PGP/inline, if the email client(s) you and we use support both methods? Justify your answer.

Hint: These webpages contain useful resources:

- `http://www.gnupg.org/` - the GNU privacy guard homepage
- `https://www.devdungeon.com/content/gpg-tutorial` for installing gpg on your system and all commands you'll ever need (including this problem).
- `https://www.base64encode.org/` - Might come in handy to base64 encode your signed/encrypted messages.

In addition, please note:

- We recommend using **gpg**, a PGP utility by GNU. **gpg** is installed by default on WSL(ubuntu), ubuntu and MacOS. You will have to install **gpg** yourself on native Windows (fairly simple). You are free to use any other utiliy but TAs might not be able to help you debug issues.
- If you use **gpg** over SSH without a GUI, *"–pinentry-mode loopback"* flag might come in handy.
- For MacOS users, you might face issues while pasting large chunks of data to the CTF problem due to MacOS terminal restrictions. In such case, put your input in a file (one input per line), and pipe the content of the file to your netcat session:

```
cat input.txt | nc 192.168.2.83 XXXX
```

2 Hash extension (30 points) (Optional Team Work)

You can choose to form a team of 2 students or work individually. If you are working in a team, only one needs author the write up, and the other simply writes "*see [teammate andrewid]*". Each team member is expected to write their own code and individual CTF username and flag need to be put in the write up.

SHA256 is a commonly-used algorithm for hashing, based on Merkle–Damgård construction. What happens if it is used as a message-authentication code? Can you modify a message and regenerate its hash without knowing the original authentication key? Solve the Hash Extension problem on the CTF server.

Hint: Start with the file `pure_python_sha256.py` which is a simple Python 3 implementation of SHA256 which you can modify in order to implement the extension attack. This problem does **NOT** require any brute force, it can be solved just by connecting to the server twice (once to get the original cookie and its hash, and the second time to submit your modified cookie and re-calculated hash).

Preliminaries

SHA256 Pseudocode: The "Pseudocode" section of `https://en.wikipedia.org/wiki/SHA-2#Pseudocode` is a good reference for the SHA256 algorithm.

SHA256 Padding: The SHA256 algorithm operates on 64-byte chunks of the message. It will add padding in the following form (even if the message is already a multiple of 64-bytes long, it will still add padding): First it will append a single byte 0x80. Then, it will append 0x00 until the length is 8 bytes less than a multiple of 64 bytes (note that if the original length were, say, 63, then there would be one byte of 0x80 and then 56 bytes of 0x00 added). Finally, it adds the original (pre-padding) length of the message (in BITS, not BYTES). As an example:

Start with a 50-character string
"XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXABC"

In hex this is 50-bytes long:
5858...5858414243

Append a 0x80, making it 51 bytes long:
5858...585841424380

Append 0x00, until it is 8 less than a multiple of 64 bytes long (56 bytes):
5858...5858414243800000000000

Append the original length of the unpadded message (50 bytes = 400 bits = 0x190 in hex):
5858...5858414243800000000000000000000000190

Unlike the padding oracle attack, this attack itself is not based on padding, but you will need to manually pad the message before length-extending it.

SHA256 on Long Strings: The SHA256 algorithm is meant to operate on 64-byte blocks using Merkle–Damgård construction. For longer strings, it loops over itself multiple times, maintaining a running hash value consisting of 8 32-bit integers (often referred to as `h0, h1, h2, h3, h4, h5, h6, h7`). At the start of the hash, these are initialized to a fixed set of values (defined by the SHA256 specification), and during each loop iteration these 8 values are updated. At the end of the hash, these 8 values are converted to hex, concatenated together, and outputted as the hash value.

3 Validating Certificate Chains (45 points)

In class you learnt about how TLS secures network data by encrypting HTTP traffic. A key part of this process is validating the certificates, which prevents malicious actors from posing as trusted websites (as long as we trust the right people to begin with).

Part A: My certChainCheck

Challenge: In this problem, you will be writing your own X.509 certificate-chain validation program. Although there are a ton of nitty-gritties (which we hope you'll realise as you go) that goes into it, we do want to provide you with a proper amount of hands-on experience.

Scope:

- You do not need to parse the Certificates yourself (ASN.1 is not the simplest).
- All libraries that you will need are listed in your starter code. This is to save you some time, as well as to ease us with testing your code. You **cannot** use any other libraries than the ones included in the starter code. (ie. server will not have any other 3rd party library available. However, you may use any other native python3 library. Example, import re)
- You are expected and required to understand the documentation for the libraries listed and figure out what you need to use in order to get the feature working.
- **Python 3 is required**, due to how our testing infrastructure is set up and our TAs' ability to help with debugging. **Important note:** If you want to use another language, we are open to accommodate you only on a case by case basis. Please post a private question on piazza to discuss further.
- Your program should just be good enough to pass all the provided test cases in a proper way (without any hard-coded chunks that just get you through some test cases). You are **not** expected to implement the entire specification: `https://tools.ietf.org/html/rfc5280`
- Please watch first 30 minutes of `https://www.youtube.com/watch?v=gmYcsdXT3W8` for a huge hint for this problem.

Design: You will be required to fill in provided starter code and complete the function `x509_cert_chain_check()` in file `certChainCheck.py`. You may create any additional helper functions as you please.

Testing: Your Program will be tested against a set of domains with known good and bad certificates. You will have access the set of tested domains. Read the **README.md** file in the starter code to understand how to test your code locally.

Submission: For Part A, you will be 1) submitting your completed `certChainCheck.py` file on Gradescope and 2) including an explanation about your approach in the homework report.

Part B: Follow up

Wohoo! We wrote our own complete X509 Certificate chain checking program!

Wait, I take my words back I thus realized, the algorithm we wrote has a glaring flaw in it. Can you figure out the flaw and explain how it can be handled? NOTE: you do not need to write any code for this part, just explain the flaw and how the spec addresses it in your homework report to get credit for Part B.

Hint: The private key of my company website's SSL certificate got compromised. What do I do?