Padding Oracle Attack Lab

Due: February 27th by 11:59 PM

1 Lab Overview

In this lab, you will be exploiting the padding oracle in order to decrypt a message.

1.1 How padding works!

The preferred method of padding block ciphertexts is PKCS7. In PKSC7, the value of each padded byte is the same as the number of bytes being added. So if a block is 12 characters, you pad it with $[04, 04, 04, 04]$. If it is 15 characters, you pad it with $[01]$. If it is exactly 16 characters, you add an entire extra block of $[16] * 16$.

So a decrypted plaintext with a final block ending in $[..., 13, 06, 05]$ is not valid. The original cipher text therefore could not have been valid - there are no allowed plaintexts that would encrypt to that ciphertext.

1.2 The Padding Oracle Attack

It turns out that knowing whether or not a given ciphertext produces plaintext with valid padding is ALL that an attacker needs to break CBC encryption. If you can feed in ciphertexts and somehow find out whether or not they decrypt to something with valid padding or not, then you can decrypt ANY given ciphertext.

See additional resources for details: [1, 2].

2 Lab Environment

You can use any Python3 environment for this lab.

Three files are provided:

1. `poa.py`: A Python3 file where you will have to implement the incomplete function `padding_oracle_attack_exploit()`
2. `requirements.txt`: A Python3 dependency file. You can use

   ```
   pip install -r requirements.txt
   ```

   to install all the dependencies.

For this lab, you will have to perform the following:

1. Implement the function `padding_oracle_attack_exploit` function to decrypt a cipher text generated from using the `encrypt` function using only the calls to `oracle` function.
2. Answer the following questions in your report.
   (i) What is the padding oracle attack? (40 points)

(ii) Working implementation of the standard Padding oracle attack obtain the plaintext from the ciphertext. (100 points)

(iii) Description of your implementation with screenshot(s). (40 points)

(iv) **(BONUS.)** We have provided a slightly modified version of the padding function `pad` and `unpad` where the length of the plaintext is also prepended to the message before padding. It is possible to partially recover 1 block of plaintext. Can you find out how? Describe your technique with implementation (50 points).

3 Submission

Total points earnable: 180 points + 15 points for Writing quality + 50 bonus points

3.1 What to Submit

Your submission directory should contain:

- All source files written by you to perform the lab tasks.
- A README text file describing how to compile and execute your source code.
- **Report (Writing quality - 15points)**: A detailed report containing an explanation of the observations. Name this report *Analysis-lab2.pdf*. Be sure to put your name in your report.

3.2 How to submit

Please use Brightspace to make your submissions.

Submit every time you have a stable version of any part of the functionality. Make sure to submit early to make sure you do not miss the deadline due to any last minute congestion.

Note that resubmitting overwrites any earlier submission and erases any record of the date/time of any such earlier submission.

NOTE: Do not submit your virtual machine disk image. Copy the files for submission off of your virtual machine and into a directory.

References

[1] The Padding Oracle Attack. https://robertheaton.com/2013/07/29/padding-oracle-attack/

[2] Cryptopals: Exploiting CBC Padding Oracles https://research.nccgroup.com/2021/02/17/cryptopals-exploiting-cbc-padding-oracles/