Spring 2017 Cryptography and Network Security

Homework 3

Release Date: 5/21/2017

Due Date: 6/11/2017, 23:59

Instruction

- **Submission Guide:** Please submit all your codes and report to CEIBA. You need to put all of them in a folder named by your student id, compress it to hw3_{student_id}.zip. For example, hw3_r04922456.zip.
- You may encounter new concepts that haven't been taught in class, and thus you're encouraged to discuss with your classmates, search online, ask TAs, etc. However, you must write your own answer and code. Violation of this policy leads to serious consequence.
- You may need to write some programs in the Capture The Flag (CTF) problems. Since you can use any programming language you like, we will use a pseudo extension code**.ext** (e.g., code.py, code.c) when referring to the file name in the problem descriptions.
- You are recommended to provide a brief usage of your code in **readme.txt** (e.g., how to compile, if needed, and execute). You may lose points if TAs can't run your code.
- In each of the Capture The Flag (CTF) problems, you need to find out a flag, which is in `BALSN{...}` format, to prove that you have succeeded in solving the problem.
- Besides the flag, you also need to submit the code you used and a short write-up in the report to get full points.
- In some CTF problems, you need to connect to a given service to get the flag. These services only allow connections from 140.112.0.0/16 and 140.118.0.0/16.

Handwriting

1. SYN Cookie (8%)

(a) (2%) Explain why SYN cookies can mitigate SYN flooding attacks.

(b) (2%) Explain why the cookie needs to contain a timestamp.

(c) (2%) Explain why the cookie needs to contain the client IP address.

(d) (2%) If an attacker could forge the Message Authentication Codes (MAC) used in a SYN cookie, what could he do?

2. DDoS (12%)

(a) (2%) Can a traditional firewall like iptables mitigate DDoS attacks? Why or why not?

(b) (2%) Attackers can make use of DNS servers to launch an amplification attack, which can enlarge the traffic to more than 50 times. An HTTP server could provide a larger amplification fator. Please explain whether an attacker can leverage an HTTP server to launch a more severe amplification attack.

(c) (4%) We have talked about collisions in hash table as an example of the algorithmic complexity attack in class. Please give two other examples of the algorithmic complexity attack against webservers and explain how they work.

(d) (4%) Besides flooding attacks, there is another category of DDoS called the *low and slow attack*. Please give two examples of the low and slow attack exploiting the HTTP protocol and explain how they work.

3. SSL/TLS (14%)

(a) (8%) Please explain what features of SSL/TLS is used to defend the following attacks:

  (i) (2%) Spoofing attacks: Pretend a connected client to fool a host into accepting bogus data.

  (ii) (2%) Man-in-the-middle: Act as the client to the server and as the server to the client during the key exchange phase.

  (iii) (2%) Replay attacks: Replay a single SSL/TLS packet of application data.

  (iv) (2%) Replay attacks: Replay a whole SSL/TLS connection. Start from replaying a `"Client Hello"` message (the handshake phase).

(b) (2%) What is forward secrecy? Why forward secrecy is important?

(c) (2%) What is Rollback attack? How to prevent it?

(d) (2%) What is SSL Stripping attack? Explain how HTTP Strict Transport Security (HSTS) defends against SSL Stripping attack.

Capture The Flag

4. CBC Encryption Mode (10%)

We have talked about ECB mode of operation in homework2, and I think you have undertood that ECB mode is vulnerable. This time, we are going to discuss another encryption mode: `CBC mode`. In this mode, each block is XORed with the previous cipher plaintext block before encryption. Besides, an initialization vector `(IV)` is used to increase randomness. It seems that CBC mode is more secure than ECB mode, however, if used improperly, CBC mode is still vulnerable to a sort of attack. In this problem, you are asked to decrypt the flag through the service `CBC_Encryption_Mode/can_you_decrypt.py`. You can access the service by `nc 140.112.31.109 10006`. Explain how you solve the problem and `simply describe how to prevent this kind of attack` in your write-up. Save you script (if exists) as `code4.ext`.

Hint: May the padding oracle be with you.

5. Eavesdropper (15%)

Find the following flags in `eavesdropper/cns.cap`, which are 802.11 WLAN packets captured in CSIE building.

(a) (3%) A wireless network called `cns2017` was using the insecure WEP encryption. Please find out the 5-byte key it used. (Don't need to enclose the key in `BALSN{}`)
Hint: aircrack-ng

(b) (4%) In `cns2017`, find the flag at `http://servers.tw:8080/flag1`.
Hint: wireshark

(c) (4%) In `cns2017`, find the flag in the uploaded file of `http://servers.tw:8080/file`.

(d) (4%) In `cns2017`, find the flag at `https://servers.tw:10443/flag2` with the private key `eavesdropper/private.key`.

6. DoS on hash table(12% + 6%)

We have learned about hash tables in the class. It is common to use a hash table to store user input, like query strings in URL. But we are just too optimistic and expect it runs in constant time for each operation. Please design an algorithm that generate special testdata, such that our hash tables below would degenerate to $\Theta(n^2)$ for n operations. A simple input file **normal_input** is provided for reference of input format.

(a) **C++ (6%)** Here is a small program **dos.cpp** written in C++ that runs fast in most cases. Please write a program that generate 50000 numbers in $[0, 2^{30}]$, such that this program runs slowly. Your evil input should make it at least 10x slower than random input. Save your program as `code6-1.ext`, and your evil input as `input6-1.txt`. Please explain how you designed the evil input.

Here is the environment that will be used to judge the performance of your evil input:

- Ubuntu 16.04.1 LTS (64bit)
- gcc version 5.4.0 (Ubuntu 5.4.0-6ubuntu1 16.04.4)
- Compile with `g++ -std=c++11 dos.cpp -o dos`

(b) **Python (6%)** Here is a small program **dos.py** written in Python that runs fast in most cases. Please write a program that generate 50000 numbers in $[0, 2^{30}]$, such that this program runs slowly. Your evil input should make it at least 10x slower than random input. Save your program as `code6-2.ext`, and your evil input as `input6-2.txt`. Please explain how you designed the evil input. *(Hint: dictobject.c)*

Here is the environment that will be used to judge the performance of your evil input:

- Ubuntu 16.04.1 LTS (64bit)
- Python 2.7.12 (The builtin python version of Ubuntu 16.04)

(c) **Bonus: PHP (6%)** Here is a small program **dos.php** written in php that runs fast in most cases. Please write a program that generate 50000 numbers in $[0, 2^{30}]$, such that this program runs slowly. Your evil input should make it at least 10x slower than random input. Save your program as `code6-3.ext`, and your evil input as `input6-3.txt`. Please explain how you designed the evil input. *(Hint: zend_hash.c)*

Here is the environment that will be used to judge the performance of your evil input:

- Ubuntu 16.04.1 LTS (64bit)
- PHP 7.0.15
- Zend Engine v3.0.0 (with Zend OPcache v7.0.15)

7. SQLi 101 (14%)

The account name of the following questions are **admin**.
The table name is **account**.
The schemas are **name** and **password**.
The database of (a)(b)(d) is SQLite.
The database of (c) is MySQL.

(a) (2%) Login to `http://104.199.120.47:31337/b` to get the flag.

(b) (4%) Find out the password of the admin at `http://104.199.120.47:31337/b`.
Hint: Boolean-based SQL Injection.

(c) (4%) Find out the password of the admin at `http://104.199.120.47:31337/c`. This time, the single quote `"'"` would be escaped to `"\'"`.

(d) (4%) Find out the password of the admin at `http://104.199.120.47:31337/d`. This time, the following keywords (case insensitive) would be removed.
keywords: ABORT, ACTION, ADD, AFTER, ALL, ALTER, ANALYZE, AND, AS, ASC, ATTACH, AUTOINCREMENT, BEFORE, BEGIN, BETWEEN, BY, CASCADE, CASE, CAST, CHECK, COLLATE, COLUMN, COMMIT, CONFLICT, CONSTRAINT, CREATE, CROSS, CURRENT_DATE, CURRENT_TIME, CURRENT_TIMESTAMP, DATABASE, DEFAULT, DEFERRABLE, DEFERRED, DELETE, DESC, DETACH, DISTINCT, DROP, EACH, ELSE, END, ESCAPE, EXCEPT, EXCLUSIVE, EXISTS, EXPLAIN, FAIL, FOR, FOREIGN, FROM, FULL, GLOB, GROUP, HAVING, IF, IGNORE, IMMEDIATE, INDEX, INDEXED, INITIALLY, INNER, INSERT, INSTEAD, INTERSECT, INTO, IS, ISNULL, JOIN,

KEY, LEFT, LIKE, LIMIT, MATCH, NATURAL, NO, NOT, NOTNULL, NULL, OF, OFFSET, ON, OR, ORDER, OUTER, PLAN, PRAGMA, PRIMARY, QUERY, RAISE, RECURSIVE, REFERENCES, REGEXP, REINDEX, RELEASE, RENAME, REPLACE, RESTRICT, RIGHT, ROLLBACK, ROW, SAVEPOINT, SELECT, SET, SUBSTR, TABLE, TEMP, TEMPORARY, THEN, TO, TRANSACTION, TRIGGER, UNION, UNIQUE, UPDATE, USING, VACUUM, VALUES, VIEW, VIRTUAL, WHEN, WHERE, WITH, WITHOUT

8. GannaPsy (15%)

It's me, Tom. Our organisation was under attack. All of our top-secret files, `including the flags`, was encrypted by the ransomware. We already recover most of our files, except `the flag of this problem`. This is your final mission. Try to recover the flag. You can find the ransomware and the encrypted flags in `helpme.zip`. And I will give you the backup of the flags, except `flag_666`, in `BACKUP.zip`

9. Bonus: XSS with Google (10%)

Practice XSS on https://xss-game.appspot.com/ and write your own writeups. Please do not lookup the answer directly. You will get zero point if you just write down the answer. We will grade your writeups according to how you approach the probelms and your understanding of the techniques shown through the report.