UNIVERSITY OF MARYLAND
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ENEE 459C
Computer Security
Instructor: Charalampos Papamanthou

Homework 5

Out: 11/27/14 Due: 12/11/14

Instructions

1. Strictly adhere to the University of Maryland Code of Academic Integrity.
2. **Type** and submit your solutions as a pdf document at Canvas. Include your full name in the solutions document. Name the solutions document as x-hw5.pdf, where x is your last name.
3. No extensions will be given. Everything will be due at 11.59pm on 12/11/14.

Problem 1 (20 points)
As we discussed in class, a switch is a device that allows multiple computers to connect to each other in such a way that each computer only receives data intended specifically for it. Suppose we have a switch that works as follows. When a computer is connected to one of its ports, it adds a tuple (x, y) to a data structure called the Server Source Address Table, where x is the MAC address of the certain computer and y is the certain port that is assigned to the computer. This new entry will overwrite any other entry with the same MAC address. Assume an attacker can choose her own MAC address. Describe an attack that the network connected by this switch is vulnerable to.

Problem 2 (20 points)
The TCP three-way handshake works as follows: When a client wants to establish a TCP connection with a server, it first sends a SYN message with sequence number x. Then the server replies with a SYN-ACK message that has a fresh sequence number y along with $x + 1$, and finally the client replies with an ACK message that contains the sequence number $y + 1$.

In class we talked about the SYN-flooding attack, which works as follows: A malicious client could send a very large number of SYN messages (say M such messages) to the same server from M different spoofed IPs. When the server receives SYN message i, it would allocate space in memory for this connection i, i.e., it would store the fresh sequence number y_i that it chooses for the respective SYN-ACK message i as long as the respective client IP and port ($i \in \{1, \dots, M\}$). With SYN flooding, a server's resources will become exhausted since eventually all of its resources will be allocated for these half-open connections.

One of the proposed solutions to this problem is an approach called "SYN cookies". SYN cookies work by constructing a sequence number y for the SYN-ACK message that encodes information about the connection: Namely, when a server receives a SYN message, it replies with the appropriate SYN-ACK message (with the sequence number constructed as below) and then discards the state of the connection. If the client responds with the appropriate ACK message (i.e., with sequence number $y + 1$), then the server can reconstruct the state of the connection from its encoding in the sequence number. The SYN cookie has the following components:

- (5 bits) A timestamp t that equals $current\_time$ modulo 32 (where $current\_time$ is in seconds);
- (3 bits) The maximum segment size m that the server would have stored in the SYN queue entry;
- (24 bits) A $\mathsf{MAC}_k(.)$ of (i) the server IP address and port number for the connection; (ii) the client IP address and port number; and (iii) the value t. Note that only the server knows secret key k.

1. Explain why it is important that the sequence numbers x and y are chosen at random.
2. Even if x and y are indeed chosen at random, what might be a problem when the connection is not over SSL?
3. Explain why SYN cookies help to mitigate SYN flooding attacks.
4. When SYN cookies are used, explain what the server needs to do when he receives an ACK message from the client. Recall that without SYN cookies, the server just needs to decrease the sequence number by one and compare it to the locally stored sequence number.
5. Explain why the cookie needs to contain the counter t.
6. Explain why the cookie needs to contain the client IP address.
7. What could an attacker do if he could forge the MAC used in a SYN cookie, namely if he could compute arbitrary MACs without having access to the secret key k?

Problem 3 (20 points)

Suppose a new computer virus, called H1NQ, just got released. Mike has a new malware-detection program, QSniffer, that is 95% accurate at detecting H1NQ. That is, if a computer is infected with H1NQ, then QSniffer will correctly detect this fact 95% of the time, and if a computer is not infected, then QSniffer will correctly detect this fact 95% of the time. It turns out that the H1NQ virus will only infect any given computer with a probability of 1%. Nevertheless, you are nervous and run QSniffer on your computer, and it unfortunately says that your computer is infected with H1NQ. What is the probability that your computer really is infected?

Problem 4 (20 points)

In this assignment you are going to write a program (in the programming language of your preference) that implements Merkle hash trees (originally introduced by Ralph Merkle in this paper), as we discussed in class. Suppose n is a power of 2. Your program should implement the following functions:

1. $T \leftarrow \mathsf{setup}(f_1, f_2, \ldots, f_n)$. On input n files $f_1, f_2, \ldots, f_n$ (1 KB each) it outputs the Merkle hash tree T. Recall that a Merkle hash tree is a binary tree such that for each internal node v (that has children u and w) we store a hash $h_v = \mathsf{HASH}(h_u, h_w)$. If v is a leaf corresponding to file f_i we set $h_v = f_i$. You can use an existing implementation of SHA-2 for the implementation of HASH.
2. $\pi_i \leftarrow \mathsf{prove}(i, T)$. On input an index i and the Merkle hash tree T, it outputs a proof π_i for file f_i. Recall the proof is a collection of hashes along the path from i to the root r.
3. $\{\texttt{ACCEPT}, \texttt{REJECT}\} \leftarrow \mathsf{verify}(f_i, \pi_i, h_r)$. On input a file f_i, a proof π_i and the hash of the root h_r, it outputs either ACCEPT or REJECT.

4. $T' \leftarrow \mathsf{update}(f_i', T)$. On input an updated file f_i', it outputs the updated Merkle hash tree T'.

Run experiments and plot the time it takes to execute the algorithms $\mathsf{setup}()$, $\mathsf{prove}()$, $\mathsf{verify}()$ and $\mathsf{update}()$ for $n = 2^{10}, 2^{11}, 2^{12}, \ldots, 2^{20}$.

Problem 5 (20 points) Please find the fifth problem at:

`http://enee459c.github.io/homeworks/buffer-overflow/`