UNIVERSITY OF MARYLAND
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ENEE 457
Computer Systems Security
Instructor: Charalampos Papamanthou

Homework 4

Out: 11/09/18 Due: 11/16/18 11:59pm

Instructions

1. Strictly adhere to the University of Maryland Code of Academic Integrity.
2. Submit your solutions as a pdf document at Canvas. Include your full name in the solutions document. Name the solutions document as x-hw4.pdf, where x is your last name.

Problem 1 (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 2 (20 points)

Suppose you know that the passwords used by the students to authenticate to the university server consist of n uppercase letters of the English alphabet. Given the hash $h(p)$ of a password p describe three solutions to compute password p.

1. The first solution uses constant space, $O(26^n)$ query time and no preprocessing.
2. The second solution uses $O(26^n)$ space, constant query time and $O(26^n)$ preprocessing time.
3. The third solution uses $O(26^{\frac{2n}{3}})$ space, $O(26^{\frac{2n}{3}})$ query time and $O(26^n)$ preprocessing time.

Because of the above attacks, the administrator changes the way passwords are stored: Instead of storing $h(p)$ for password p, the administrator stores $(h(p||r), r)$, where r is some randomness that is chosen for each password. Which one of the above solutions will not work now? Why?

Problem 3 (20 points)

Given the confidentiality categories TOPSECRET, SECRET, CONFIDENTIAL, UNCLASSIFIED and the integrity categories HIGH, MEDIUM, LOW, indicate and explain what type of access (read, write, both, or neither) is allowed in the following situations.

1. Paul, cleared for (TOPSECRET, MEDIUM), wants to access a document classified (SECRET, HIGH).

2. Anna, cleared for (CONFIDENTIAL, HIGH), wants to access a document classified (CONFIDENTIAL, LOW).
3. Jesse, cleared for (SECRET, MEDIUM), wants to access a document classified (CONFIDENTIAL, LOW).
4. Sammi, cleared for (TOPSECRET, LOW), wants to access a document classified (CONFIDENTIAL, LOW).
5. John, cleared for (SECRET, LOW), wants to access a document classified (SECRET, LOW).
6. Robin, who has no clearances (and so works at the (UNCLASSIFIED, LOW) level), wants to access a document classified (CONFIDENTIAL, HIGH).

Problem 4 (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 5 (10 points)

Bob thinks that generating and storing a random salt value for each userid is a waste. Instead, he is proposing that his system administrators use a cryptographic hash of the userid as its salt. Describe whether this choice impacts the security of salted passwords and include an analysis of the respective search space sizes.

Problem 6 (10 points)

Most modern TCP implementations use pseudo-random number generators (PRNG) to determine starting sequence numbers for TCP sessions. With such generators, it is difficult to compute the ith number generated, given only the $(i-1)$st number generated. Explain what network security risks are created if an attacker is able to break such a PRNG so that he can in fact easily compute the ith number generated, given only the $(i-1)$st number generated.