CSci 4271W Developing Secure Software Systems (section 010)

Homework 5 **Due: April 8th, 2025**

Ground Rules. You may choose to complete these exercises in a group of up to three students. Each group should turn in **one** copy with the names of all group members on it. You may use any source you can find to help with this assignment but you **must** explicitly reference any source you use besides the lecture notes or assigned readings. No answers should come from people outside your group, or from AI tools like ChatGPT. If you use an AI tool to revise your writing, save a copy of the first draft you wrote yourself to provide it was originally your work. Electronically typeset copies of your solution should be submitted via Gradescope by 11:59pm on April 8th, 2025.

1. **Confidentiality and Integrity**. In class we learned about symmetric *encryption* schemes and symmetric *authentication* schemes. Symmetric encryption provides the classic security property of "confidentiality" against passive attackers, but if an attacker attempts to modify a ciphertext that is encrypted with a scheme that doesn't also provide authentication, and can observe how the recipient reacts to the modified ciphertext, they might be able to learn about the contents of the original plaintext. An example of an unauthenticated encryption scheme is so-called CTR, or Counter Mode, encryption using a block cipher such as AES. In this scheme, a symmetric key and a randomly chosen "initial counter" are used to produce a long and random-looking string, P that is combined with the plaintext M via XOR to produce the ciphertext C, so $C = M \oplus P$.

(a) Using your knowledge of ASCII encodings of English letters, describe how an attacker can convert a sequence of English letters encrypted with this encryption scheme from lower-case to upper-case or vice-versa, without knowing the letters themselves.

(b) Now suppose that Ape will send Bear a message using only English letters and spaces, and Bear's computer will drop messages that contain any other characters. If Mal, an on-path network attacker, wants to know where the spaces are in Ape's message, how could they use the technique from part (a) to accomplish this task?

(c) If Ape and Bear instead use an authenticated encryption scheme, then any change Mal makes to a message encrypted by Ape will result in a decryption error. Explain why this would prevent an attack like the one presented in part (b).

2. **Broken RNGs**. Let's do a few quick calculations about the potential impacts of various famous broken random number generators.

(a) The Netscape (the ancestral browser of Firefox) SSL random number generator in 1997 was found to be using a combination of the current time in seconds and microseconds since the Unix epoch (two 32-bit numbers), the browser PID (a 16-bit number on systems of the time), and the browser process's parent PID (also a 16-bit number) to generate symmetric keys. Let's assume that a network adversary can estimate the time of the key generation with millisecond accuracy, such as based on optional TCP timestamps or arrival times, so that there are 1000 possible values for the time input. Further, let's assume that the user runs the browser in a common way which causes the parent PID of the browser to always be 1.

In this version of Netscape, to connect to a server using SSL, the client would generate a symmetric key K (using the above inputs), encrypt K using the server's 1024-bit RSA public key, and send this value to the server; then the client and server would use K to symmetrically encrypt the rest of their session (using essentially the same record protocol as in TLS).

For an on-path network adversary under the assumptions mentioned above, how many possible symmetric keys would there be for a captured key exchange? Knowing that in this application of RSA, two encryptions of the same symmetric key with the same public key result in the same ciphertext, describe how an adversary could test a possible key. Now run the command `openssl speed rsa` on your VM (which runs as many cores – 1 – and at a similar speed as a nice workstation at the time) to see how many 1024-bit encryption (public key) operations can be performed per second. At this speed, how long would it take an attacker to determine the symmetric key, enabling them to decrypt the session?

(b) In 2007, a bug in the Debian package for OpenSSL caused ssh daemons to use just a 15-bit value as input to the public-key generation algorithm. How many possible public/private key pairs could a machine running this build of sshd have? You can generate one such public key on a CSELabs machine (or VOLE) using the command `time openssl dhparam 1024`. Estimate how long it would have taken to generate all of the possible public/private-key pairs for this build of sshd.

Mal-in-the-Middle Attacks

The next two problems are about examples of real crypto protocols that have been vulnerable to Mal-in-the-Middle attacks. Note: You can find descriptions of these attacks by searching online, but that will defeat the point of learning to look for ways that protocols can break, depriving you of skills you might find useful later in life (or on an exam).

3. **Mal-in-the-Middle attacks I: Replay**. The OCSP protocol allows a client to check if a certificate, with ID $CertID$ has been revoked. In the protocol,

- The client sends the server a query, $Hash(CertID)$
- If the certificate has not been revoked, the server sends $Sig_S(\text{“valid”}, Hash(CertID))$ to the client and if the signature verifies, the certificate is accepted as valid.
- Otherwise, the server sends the client $Sig_S(\text{“revoked”}, Hash(CertID))$ and if the signature verifies, the client rejects the certificate.

Describe a replay attack that allows Mal-in-the-Middle to convince a client that a certificate is valid, even after it has been revoked. (Note that Mal cannot forge the server's signature on previously unsigned messages)

4. **Mal-in-the-Middle attacks II: Downgrade:** In the SMTP protocol, a client connects via TCP (port 25) to a mail server. If the servers supports TLS-encrypted SMTP, it will respond to the client's hello message with a “250 STARTTLS” status. If the client also supports TLS, it responds with a STARTTLS message, and the client and server negotiate a TLS session. If the client does not support TLS, it will ignore the “STARTTLS” portion of the “250” response status and continue using unencrypted SMTP commands. (The server then also continues using unencrypted responses). Describe a downgrade attack that would enable Mal-in-the-Middle to prevent the client and server from using TLS, even if both support it.