School of Information and Physical Sciences

SENG 2250 System and Network Security

Assignment 2

REFLECTIVE QUESTION:

1. What other potential security issues are there with this program? Identify and discuss at least three.

Here are 3 potential security issues with the current program:

- Lack of client authentication:
  - The server only verifies the client's certificate against the Certificate Authority; yet the client does not have to present its own certificate for authentication. A malicious actor can assume the client's identity to intercept on and access communications between the client and server. Mutual SSL authentication can be executed by requiring that both the client and server present their certificates *(How Does SSL Mutual Authentication Work?, 2022).*
- No data integrity mechanism:
  - The program uses SSL to secure communication but does not explicitly check the transmitted data's integrity. Even though SSL offers confidentiality, an attacker could change data in transit undetected. To ensure message integrity, we can utilize cryptographic message authentication codes (MACs) or rely on SSL's built-in message integrity protection (HMAC).
- Man-in-the-Middle:
  - The client's line 'context.check_hostname = False' permits connections without validating the server's hostname. A malicious actor could masquerade as the server without the client's knowledge, enabling a Man-in-the-Middle (MITM) attack. Activate hostname verification in the client to ensure that the certificate corresponds to the expected server.

2. If Barry were to secretly be the leaker, would the program remain secure? Explain why or why not.

If Barry (the CA) were secretly leaking the communications, the program would not remain secure because of the following reason:

The CA's reliability is the foundation of the entire SSL/TLS trust mechanism. The issuance of certificates that confirm the communicating parties' identities is under the authority of the CA. In this instance, Jill (the server) and Chris (the client) are authenticated using Barry's CA. If Barry was malicious, he could give fraudulent certificates to other organizations so they could pretend to be Chris or Jill. By using the certificate Barry issued, he can access to private key or create a man-in-the-middle attack (MITM)). Barry could set up a MITM attack by generating and distributing a fraudulent server certificate, allowing him to intercept and read communications between Chris and Jill. Because Chris trusts Barry's CA, he would accept the fraudulent certificate.

The program relies on Barry being a trustworthy CA. If Barry were the leaker, the trust model would be broken, and the SSL/TLS communication would not remain secure.

REFLECTION

Task 1

For Task 1, I used several sources, such as the official Python SSL library documentation (*ssl — TLS/SSL wrapper for socket objects*), to learn more about how SSL/TLS works technically to allow secure socket contact. This paper explained how to set up SSL contexts, load certificates, and control encrypted communication between clients and servers. I also went over ideas we talked about during lecture again, especially those from week 5, when we learned about Public Key Infrastructure (PKI) and digital keys, in particular the X.509 standard. This information was very important for correctly setting up the SSL certificates and knowing how the authentication process worked.

In the lab sessions week 6, I learned how to use OpenSSL, which allowed me to create and sign digital certificates. This hands-on experience was directly helped me a lot with Task 1, where I set up an SSL environment to prevent eavesdropping on client-server communication. By configuring Barry as the CA, I was able to establish a trusted certificate chain that Chris and Jill could use. These lab exercises reinforced the theoretical aspects of PKI and highlighted the importance of certificate authorities in establishing secure connections. Additionally, they highlighted the importance of the CA in verifying identities and maintaining trust in a communication network.

By using these ideas in Task 1, I discovered a few possible real-life uses. SSL/TLS, for instance, is the basis for safe web transactions and is used by many e-commerce and online banking sites to keep users' private data safe. It's also becoming an important part of IoT interactions, which is the reason why Tesla built SSL/TLS into their cars (Ciberforma Lda, 2024). The task's client-server setup cloud was expanded to support multiple users, imitating a secure chat app or secure email service. As part of improvements, mutual SSL could be used, in which both the client and the server verify each other to stop people from impersonating each other.

Task 2

For Task 2, I used a lot of what we learned in week 3 of the lecture, especially the RSA algorithm and how it works to keep contact safe. I also learned more about the RSA algorithm through the GeeksforGeeks document (GeeksforGeeks, 2024). I also reviewed my lab exercises in week 3, which involved implementing the fast modular exponentiation algorithm. This algorithm, which I implemented from scratch, allowed for fast calculations required by RSA, especially for operations involving large prime numbers. The RSA algorithm itself, which hinges on the difficulty of factoring large integers, served as the foundation of the password manager's security by encrypting passwords on the client side before sending them to the server.

In the practical setup, I extended the client program to create my own RSA including key generation, encrypt password and decrypt password, built interface for users to store, retrieve and end the session. It was essential to ensure that all passwords were encrypted before being sent to the server, aligning with the requirement that neither the server nor any potential interceptors should ever see the cleartext passwords (I verify that by using Wireshark to observe sent packets and just seen encrypted password). By securing the transmission with RSA, I guaranteed that only the client, who knows the private key, could encrypt and decrypt the passwords. This encryption model is similar to how modern password managers work, where encryption is applied locally on a user's device, preventing any server from accessing unencrypted data.

The completion of this task highlighted several real-world applications of RSA and password managers. RSA prevents MITM attacks in real-world applications by ensuring that private information like passwords, credit card numbers, and personal information stays encrypted. This project showed me how encryption techniques, such as RSA, may keep passwords safe from hackers even in the case of a compromised server. To further strengthen user data against sophisticated attacks the application might be improved by adding more security measures such salt hashing before encryption.

REFERENCE

How does SSL mutual authentication work? (2022). Alert Logic Support Center. https://support.alertlogic.com/hc/en-us/articles/360002999732-How-does-SSL-mutual-authentication-work

ssl — TLS/SSL wrapper for socket objects — Python 3.7.2 documentation. (2018). Python.org. https://docs.python.org/3/library/ssl.html

Ciberforma Lda. (2024, January 6). *Transport Layer Security (TLS): Beyond the Basics*. NETWORK ENCYCLOPEDIA. https://networkencyclopedia.com/transport-layer-security-tls-beyond-the-basics/#Common-Myths-and-Misconceptions

GeeksforGeeks. (2024, September 11). *RSA Algorithm in Cryptography*. GeeksforGeeks. https://www.geeksforgeeks.org/rsa-algorithm-cryptography/