Web security lab

Edgar Vedvik

Group 42

March 31, 2017

Since this is an individual report, it follows that the individual lab exercises and this report were completed by myself alone.

1 Introduction

In this lab we were tasked with exploring many different aspects of information security. There were four main categories that we were to explore and solve challenges in. The first category was PGP, and the tasks included creating certificates, uploading them and signing and verifying messages. The second category was certificates and Certificate Authorities(CA's). The tasks here were to create personal and group certificates and using them to access websites. Our third task was to configure access control for an Apache webserver, setting up SSL and client authentication. Lastly we were tasked with securing a PHP application.

All individual tasks were completed for the lab, this includes all the PGP tasks, and all CA tasks up until the group CA request.

2 Questions

Q1: What are examples of a Certificate Hierarchy graph and Web of Trust graph in use in the world today? Are there specific organizations that prefer one or the other?

A1: The strong set is the largest collection of strongly connected PGP keys, which means any two keys have a path between them.

Q2: How does the .p12 file work? Explain what this file is and its relationship to the NTNU CA

A2: .p2 or PKCS #12. is an archive format that can story multiple cryptography objects in the same file. Such as private key with its X.509 certificate. The NTNU CA is the root CA for this.

Q3: Comment on security related issues regarding the cryptographic algorithms used to generate and sign your group's web server certificate (key lengths, algorithms, etc.).

A3: The key length chosen was 4096 bits RSA. The key is within the recommended length.

Q4: List and explain each of the verifications. What is the difference between the two files? Who has signed the Apache release? What is the point of this process?

A4:

1. PGP signature. To verify this we need the public key from a key server. Once we have this we can verify the download with the key. If it matches, we know the file has not been tampered with. However, in order to trust the key we either need to meet the person face to face, or enter a web of trust.

2. MD5 hash can be used to verify that the file is the same as the the one you intended to download. It does not say anything about who gave you this file or that the file is from apache.

Q5: What is the purpose of a certificate chain? Describe, on a high-level, the steps that your browser takes when verifying the authenticity of a web page served over HTTPS.

A5: Certificate chains are used to verify the authenticity of a certificate. The signature for each certificate is created with the private key of the next certificate in the chain, and a browser can then verify the certificate against the public key of the next certificate until it reaches a root CA.

Q6: Web servers offering weak cryptography are subject to several attacks. List 3 specific feasible attacks that are possible due to weak cryptography and explain them briefly. How did you configure your server to prevent such attacks?

A6:

1. Factorisation of RSA modulus: RSA relies on the difficulty of factorizing large primes. In 2009, researches were able to factorize 768-bit RSA Kleinjung et al. (2010).

2. Brute force attack: With more and more computational power, GPUs and FPGAs specifically designed to crack hashes, hashes with a low amount of keys, can be cracked very quickly.

3. Chosen-Ciphertext attack (CCA). An attacker chooses a ciphertext to send to the server and receives a decrypted plain-text. By doing this enough times, the server leaks information that can be used to obtain the private key.

Q7: Cookies can be a potential security risk if not handled properly, especially if they contain sensitive information. Two important flags can be set on cookies: HTTP-Only and secure. What does the HTTP-Only flag do and how does it work? What does the secure flag do and how does it work?

A7: A HTTP-Only cookie cannot be accessed by client side APIs, such as Javascript and is therefore safe from XSS. It is however, still vulnerable to

CSRF attacks. The secure cookie can only be transmitted over HTTPS, and can therefore not be eavesdropped upon.

Q8: What kind of malicious attacks is your web application (PHP) vulnerable to? Describe four briefly, and point out what countermeasures you have developed in your code to prevent such attacks.

A8: This task was not completed, but I have described them anyways:

1. SQL injections: Wherever user input is used to query a database we are vulnerable to SQL injections if we do not handle the input correctly.Do not escape and concatenate / interpolate data that goes into an SQL query yourself. Use prepared statements.
2. Cross Site Request Forgery (CSRF): Execute unauthorized commands from a user that the website trusts. For example placing a malicious link in an ¡img¿ tag will cause the browser to open this site and potentially execute some action that the user did not authorize. To prevent this, use CSRF tokens or re-authentication.
3. Cross site scripting (XSS): Inject malicious code in user supplied data on a webpage. If this is not escaped properly, the code will then be run whenever a user views the user supplied data. The solution is to sanitize the user input or to not allow certain characters or patterns.
4. Code/Shell injection: When running user input through shell commands or eval() function a malicious attacker can supply code executes additional commands that you did not expect. Using eval is seen as bad practice and should be avoided. If shell commands are necessary, make sure the input to these functions are not tainted.

Q9: Passwords should be salted and hashed. Why are both of these steps necessary? Explain how you have implemented these for storing passwords in your database.

A9: Hash algorithms are one-way functions, therefore a hashed password cannot be reversed to the original password. This adds an additional security feature should the password database be exposed. Lookup tables and rainbow tables can still be used to crack passwords relatively quick since every hash of a password is the same. To mitigate this we introduce salting, which is the process of appending a random string of bytes to the password before hasing so that every hash of the same password will be different.

I have not implemented this, but a good implementation would be to select a slow hash algorithm, such as bcrypt. The salt should be generated by a crypographically secure pseudo-random number generator. Add the salt to the

password and then hash it. Save the hashed password and the salt in the database

References

Kleinjung, T., K. Aoki, J. Franke, A. K. Lenstra, E. Thomé, J. W. Bos, P. Gaudry, A. Kruppa, P. L. Montgomery, D. A. Osvik, H. Te Riele, A. Timofeev, and P. Zimmermann (2010). Factorization of a 768-bit rsa modulus. In *Proceedings of the 30th Annual Conference on Advances in Cryptology*, CRYPTO'10, Berlin, Heidelberg, pp. 333–350. Springer-Verlag.