TTM4135 - Information Security
Web Security Laboratory Report

Spring Semester 2013

Norwegian University of Science and Technology

Department of Telematics

Group 09:
Pavel Arteev
Marius Münch
Bjørn Tungesvik

1 Introduction

Nowadays web security is an important area. It is necessary to secure the own web application against people with malicious intent. Eavesdropping of communication in the internet is pretty easy for plain HTTP but much more difficulty with secure HTTP. Furthermore HTTPS brings a lot of other useful mechanisms for web applications. To impart the corresponding knowledge to students of the course ttm4135 at NTNU in the spring semester 2013 three practical laboratory weeks had to be done. This report summarizes the results and insights of one student group .

2 Results

2.1 Part 1: Certificate authority

As result of part 1, we generated personal certificates for the server. Then we established the group CA in the certificate hierarchy of NTNU and generated the certificate for the apache web server.

Q1. Comment on security related issues regarding the cryptographic algorithms used to generate and sign your groups web server certificate (key length, algorithm, etc.).

The SSL certificate has a 2048-bit length private key. This is the same as the recommended minimum key length advised by NIST for asymmetric encryption. This key length is expected to be sufficient until 2030[1]. This implies a key length that is sufficient for securing the certificates in this assignments.

Unfortunately the default message digest algorithm, MD5, was used to sign our certificates. Previous research has shown that is possible to produce collisions with relatively ease. A group of researchers described how a pair of files having the same MD5 hash could be created.[5] Later other researchers showed with using this technique how someone could generate a fraudulent SSL certificate derived from a valid one. They showed that the certificate will be accepted as valid. [3] Due these advances MD5 is considered cryptographic broken and unsuitable for further use. We need to emphasize that no wild attacks has been reported using this technique, but it should still be avoided.[13, 7]

Further analysis of the web page also reveals that we support more than we should. The site will accept weak cipher suits. Note that during the SSL handshake, a cipher suite that is appropriate for both client and server is chosen. In our case we allow the client to choose an cipher suite with encryption lower than 128 bits, which is consider insecure. Normally modern browsers don't support weak ciphers, but on our site hackers could force a lower encryption session. However, we should emphasize that the risk is minimal since the browsers won't support weak ciphers.[6] This is however resolved now, we forced the use of higher encryption in our config file.

2.2 Part 2: Access control and Apache

In order to be able to develop and run a simple webpage it is necessary to install and configure a webserver. We are using the Apache HTTP Server, release 2.4.3. This software is downloadable as tar archive from mirrors which are listed on the official apache website.[10] To ensure that the downloaded archive is neither manipulated nor corrupt a PGP signature is offered on the Apache website.

Q2. Explain what you have achieved through each of these verifications. What is the name of the person signing the Apache release?

The verification of this signature proves the integrity and the authenticity of the downloaded data, whereby the check for the latter shows, that this release is signed by Jim Jagielski, one of the founders

of the Apache Software Foundation.

The installation of the server software is described in the official Apache Documentation.[11] After the installation it is necessary to configure the server and to set the right file permissions. We configured the server in a separated file group09.conf which gets included by the common httpd.conf. We are using two virtual hosts, the first one is listening on port 8141 and arranged for plain HTTP access to our application. The second one is listening on port 8142 and requires HTTPS. Furthermore some parts of the application should only be accessible for group members and the ttm4135 staff. To be able to realise this the web server is configured to check the used certificate of a visitor. If this certificate refers to group or staff member access is granted, otherwise the visitor would get an SSL error page. Furthermore we installed and configured a fully operational SVN repository during the completion of this part. Using the apache servers built-in authentication methods, the repository is only accessible to users that can provide valid usernames and passwords. The repository is also only accessible over an https connection, which protects the integrity of the communication. This authentication method is vulnerable for bruteforcing but a sufficient password strength makes this kind of attack inefficient. Since we use a SSL encrypted connection to submit this data eavesdropping is also hard to accomplish. The users and passwords are created with the htpasswd tool with the -s flag. This flag enables the use of SHA encryption for the passwords instead the default used MD5.

Q3. What are the access permissions to your web server's configuration files, server certificate and the corresponding private key? Comment on possible attacks to your web server due to inappropriate file permissions.

All files for this task are owned by the generic user of our group, gr09. This disallows attackers to get a root-shell just by exploiting the apache software. The specific permissions on the files are also important since unwanted reading, writing or even executing of the files are a security harm. It does not make sense to execute or write on neither the certificate nor the private key after they have been created. Clearly nobody else then the owner should be able to read the private key, so only he should have reading permissions. In contrast the server certificate have to been readable for other people both inside and outside the users' group. The configuration files have to been readable and executable for the gr09 user because the apache webserver gets started with his permissions. Since this file possibly have to be altered it is an good idea to make it writable as well. It can not be assured that other users are not needing this file so it should be okay to grant them read permissions. All in all it ends up to the following permissions, denoted numerically: private key 400, server certificate 444 and configuration files 744. [4]

Wrong file permissions are a risk when the web application running on the server is vulnerable to file inclusion or file disclosure attacks. In the case of a successful attack and wrong file permissions an attacker could as an example read the private key and fake our identity. It would be even worser if the attacker is able to write to files, then the attacker is free to manipulate our certificates or configurations.[12]

Q4. Web servers offering weak cryptography are subject to several attacks. What kind of attacks are feasible? How did you configure your server to prevent such attacks?

Web servers using weak cryptography are subject to these, and other, attacks:

- Factorisation of RSA modulus. Using a man in the middle attack to in order to get the publicly known values an attacker may be able to factor out the secret key. For this attack to be feasible, the private key must be small, e.g. less than 1024 bits. To protect against these kinds of attack it is sufficient enough to use recommended key lengths. [1]

- Bruteforce attack. Here the attacker tries every possible key inside a given key space. The length of the key determines the practical feasability of performing the attack. The longer the key, the more exponentially difficult is it to break.
- Chosen ciphertext attack. The attacker gathers informations by sending choosing ciphertexts, to the victim to obtain the encryption under an unknown key. By repeatedly doing so the victim may leak information that can help the attacker to figure out the key. The best counter measure for this attack is to switch to an cipher that is know to be resilient against these attacks.

2.3 Part 3: Writing a PHP application

In part 3 we have made a small PHP application with several functions. It is possible to login to the site using username and password and sessions are used to remember logged in users. Users can be added by the staff our by group members. Furthermore it is possible to download this report in an restricted area.

In order to realise this website we had to install PHP on the apache web server. We used the official PHP manual to do this and tested our PHP engine with the $phpinfo()$ function.[2]

Afterwards we created a user table with username, password and an ID as primary key in the MySQL database for our project. After these small preparations we were able to start with programming the PHP application.

On the login page we used a cookie to store the username which is helpful for the user when he wants to login several times from the same device. After a successful login we are setting up the session variables. The use of sessions is also comfortable for the user since he has not to login again when he is visiting the site again from the same device without a previous logout. Logged in users are allowed to download the report file. The signup page is only accessible for users with valid SSL certificate. On this page certified group and staff members and can add or delete users.

3 Discussion

The lab work introduced many tools and techniques to secure the web server. In this section we will discuss the most critical ones in order to achieve desired security.

3.1 Certificates and SSL

The resulting web server relies heavily on the SSL protocol and the x.509 standard in order to provide access control and authentication. In order for this to be properly secured, it must be ensured that sufficient key lengths, strong ciphers and signing algorithms are used during signing and generation. [6]

3.2 Our application

Our PHP application which is connected to the MySQL database builds the surface for users visiting our site. Since we developed it by ourself and webapplications are often the first attack target for people with malicious intent we had to care especially here about the used security mechanisms.

Q5. What kind of malicious attacks is your web application (PHP) vulnerable to? Describe them briefly, and point out what countermeasures you have developed in your

code to prevent such attacks.

Since the application uses a MySQL database to save certain user data, the application is naturally vulnerable against SQL injection attacks, which are used to read, manipulate or delete entries from the database. To prevent those attacks we are using prepared SQL statements. If an attacker somehow is able to get our database all user passwords are salted and hashed with SHA-512 to make it difficult to use this data. Furthermore we are sanitizing all user input via *preg_match*() with a self-defined whitelist. The allowed signs are digits, all normal characters and the special characters @,!,.,-,_,. and +. This whitelist offers also protection against cross site scripting (XSS) attacks since we are not allowing signs which are mandatory to start such an attack. XSS means that malicious javascript code is injected to the webapplication and gets executed in the browser of a client. [8, 9]

With protection against XSS we have also improved our protection against session hijacking since this kind attacks often rely on a XSS vulnerability. During a session hijacking attack the information which are used to remember a certain user are stealed and used by an attacker to access the application with the rights of the user. Unfortunately we cannot offer protection against other client side attacks resulting in a leak of the session information. Nevertheless we are saving the users' IP within the session and validate this on every page to prevent hijacking via the internet, however attacks within the same network are still feasible.

Further common vulnerabilities in PHP applications are file inclusions and file disclosure vulnerabilities allowing an attacker to execute, add or read files on the server. Our application is secure against them since we are using hardcoded HTML links instead of the PHP include function and also a hardcoded filename for the download function.

3.3 SVN repository

Certainly a solution with authentication via SSL certificates would have offered a stronger security but our group decided to use the mentioned method with good passwords. The reason for this is that one of the group members is travelling a lot and often works on other PCs than his own. Therefore he would have to carry his certificate to any of this devices which probably could be malicious since they are operated by other people. Furthermore in case of using a flash-device containing the certificate there is the danger of losing it. So basically there would be in our case a higher risk to disclose the certificates unintentionally. However a weak protection of the subversion system is a huge threat to the webapplication itself. If an attacker would be able to get the access he can read and alter the sourcecode of the application. Reading it allows the attacker to find vulnerabilities much easier and altering could be used for a lot of harmful attacks and things like backdoors. Nevertheless the changes being commited by an attacker have to be updated on the server to affect the website and hopefully the updating person would check the file.

4 Conclusion

It is necessary to be aware of the used algorithms. We had to ensure that our tools are properly configured, since most tools uses legacy encryption standards per default. Most of the legacy standards are now considered to weak to provide the protection needed by a modern application. The problem with the MD5 algorithm is an example of security weaknesses that should be avoided. All in all we achieved securing the web server with appropriate security mechanism.

References

[1] Nist recommendation. http://www.keylength.com/en/4/. Online, accessed 16 - march - 2013.

[2] PHP install. http://www.php.net/manual/en/install.php. Online, accessed 17 - march - 2013.

[3] Alexander Sotirov, Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, Benne de Weger. MD5 considered harmful today - creating a rogue ca certificate. http://www.win.tue.nl/hashclash/rogue-ca/. Online, accessed 17 - march - 2013.

[4] B. Keys. File permissions and access control. http://www.linuxsecurity.com/content/view/133913/171/. Online, accessed 17 - march - 2013.

[5] J. Black, M. Cochran, T. Highland. A study of the MD5 attacks: Insights and improvements. http://www.cs.colorado.edu/~jrblack/papers/md5e-full.pdf. Online, accessed 17 - march - 2013.

[6] Networking4All. Cipher suit. http://www.networking4all.com/en/support/tools/site+check/cipher+suite/. Online, accessed 16 - march - 2013.

[7] Networking4All. Weaknesses in SSL certificates with a MD5 signature algorithm. http://www.networking4all.com/en/support/tools/site+check/md5+algorithm/. Online, accessed 16 - march - 2013.

[8] OWASP. SQL injection prevention cheat sheet. https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet. Online, accessed 17 - march - 2013.

[9] OWASP. XXS (cross site scripting) prevention cheat sheet. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet. Online, accessed 17 - march - 2013.

[10] Apache software fundation. http://httpd.apache.org/download.cgi. Online, accessed 16 - march - 2013.

[11] Apache software fundation. Compiling and installing. http://httpd.apache.org/docs/2.4/install.html. Online, accessed 16 - march - 2013.

[12] WikiBooks. Web application security guide/file inclusion and disclosure. http://en.wikibooks.org/wiki/Web_Application_Security_Guide/File_inclusion_and_disclosure. Online, accessed 17 - march - 2013.

[13] Wikipedia. MD5. http://en.wikipedia.org/wiki/MD5, 2013. Online, accessed 16 - march - 2013.