Security in the web development process

Nguyen Canh Son
Abstract

Security in the web development process

Nguyen Canh Son

Today web applications have become very common on the Internet. One aspect that can cause a lot of headaches for the managers of these web applications are security problems. Some of them are due to programming mistakes, some due to the inexperience of the developer or in other ways, the developers failed to protect their systems against harmful threats. This thesis will talk about how a system should be developed in order to protect itself from these harmful threats that can be everywhere on the internet. The thesis will have four parts, in the first one we will look at the structure of an existing business web application. This includes details about the current structure of the system, how it is implemented and details about the technology used to build the system. Next, a number of file transfer protocols used for deployment will be discuss with focus on security. The thesis also includes an overview of common attacks that developers should be aware of when developing a web application and how to avoid them. Finally, a comparison between using cloud storage and self-hosted server storage will be given that will help developers make their choice of storage. The result of the thesis is a guide for anyone who wants to develop a secure web application. This guide will include the advice for using a secured file transfer protocol like SFTP in deploying a website, how to prevent some common attacks and the recommended storage option.
Contents

1 Introduction 7

2 Background 8
   2.1 Client-server architecture . . . . . . . . . . . . . . . . 8
   2.2 Authentication . . . . . . . . . . . . . . . . . . . . . . . . 9
   2.3 Encryption . . . . . . . . . . . . . . . . . . . . . . . . . . 9
   2.4 Sniffer attack . . . . . . . . . . . . . . . . . . . . . . . . . 10

3 Current system architecture of Skolrutiner system 10
   3.1 Overall design and technology . . . . . . . . . . . . . . . . 10
   3.2 .NET entity framework . . . . . . . . . . . . . . . . . . . . 11
   3.3 AngularJs . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

4 File transfer for deployment 13
   4.1 FTP . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
   4.2 FTPS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
      4.2.1 Implicit FTPS . . . . . . . . . . . . . . . . . . . . . . 15
      4.2.2 Explicit FTPS . . . . . . . . . . . . . . . . . . . . . . 15
   4.3 SFTP . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
   4.4 HTTPS file transfer . . . . . . . . . . . . . . . . . . . . . . . 16
   4.5 WEBDAV/WEBDAVS . . . . . . . . . . . . . . . . . . . . . . 17
   4.6 Recommendations . . . . . . . . . . . . . . . . . . . . . . . 18

5 Protection against common attacks 19
   5.1 SQL injection . . . . . . . . . . . . . . . . . . . . . . . . . . 19
   5.2 XSS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
      5.2.1 Stored XSS attack . . . . . . . . . . . . . . . . . . . 24
      5.2.2 Reflected XSS attack . . . . . . . . . . . . . . . . . 24
      5.2.3 Preventing XSS attacks . . . . . . . . . . . . . . . . 25
   5.3 CSRF . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27

6 File storage 29
   6.1 Local server storage . . . . . . . . . . . . . . . . . . . . . . . 29
   6.2 Cloud storage . . . . . . . . . . . . . . . . . . . . . . . . . 30
   6.3 Recommendation . . . . . . . . . . . . . . . . . . . . . . . . 32

7 Conclusion 34
1 Introduction

Sweden is a country with a highly developed educational system. In such environment, having a system that helps the schools and teachers to easily manage documents is an essential issue. Since I worked as a TA in Vietnam, I understand the need of having an efficient file management system and the Swedish company Skolrutiner\footnote{https://skolrutiner.se} gave me a chance to interact with such system.

The main purpose of the services provided by Skolrutiner is to make the works in school more efficient when it comes to distributing important documents. Most schools have important documents on a shared hard drive only accessible within their local network. The idea is to let the staff not to be bound to their computers and for easier and quicker access to their documents. Skolrutiner gives the staff access to crucial information, such as emergency exit, accident reports, equal treatment guideline, school rules, contemporary protocols and more. Classified information is published in the system at the schools’ own risk. The purpose of the system is to be lean and only the most up to date documents are published, which erases the hurdle of having to search for files in unorganized places.

In any web application, security plays such an important role that cannot be ignored. The goal of my thesis is to:

- understand the current architecture of the Skolrutiner system
- study about security in deploying a system
- how to secure the system against some common attacks such as SQL injection, XSS or CSRF
- find out the best way of storing files for such system.

The result of my work will be a set of recommendations on how to improve the security of the development process of the Skolrutiner system and the security of the Skolrutiner system itself. These recommendations can also be used by anyone who wants to build a secure web application.
2 Background

In order for everyone to understand the thesis easier, definitions of some technical terms used in the thesis will be provided in this part.

2.1 Client-server architecture

Client-server architecture is a producer-consumer computing architecture. In this architecture, a server acts as the producer, it delivers and manages most of the resources and services to be consumed by a client, which can be considered as the consumer. In a client-server web application, the client initializes a request and sends it to the server over the network connection. The server will then process this request, contact the database if necessary and then respond to the client with the processed data. After receiving the response data from the server, the client will use the data to make changes in the HTML pages and render the new HTML pages in the browser or make changes to data stored on the client. A server machine can manage several clients at the same time, whereas one client can also be connected to several servers simultaneously, each providing a different set of services.

In a client-server web application, both the client and the server have their own storage called cookie and session. When a user visit a website, a session will be started and a new file will be created in a temporary directory on the server to store session variables and their values. This data will be available on all pages on the site during that visit and the server can use these values to keep track of the user’s activities on that website. The server will terminate the session when the user closes the browser, leaving the site or after a predetermined period of time without user activity, commonly 30 minutes duration. Unlike sessions, cookies are data stored on the client computer under the form of key and value and they are normally used for tracking purposes, such as checking if the user is logged in or not. The server sends a set of data under the form of key and value to the browser. The browser stores this information on the local machine of the user for future use. The next time the client sends a request to a web server, it also sends the cookie information to the server with the request and the server uses this information to identify the user.
2.2 Authentication

In a secured computer system, authentication is the process of identifying an individual. In this process, users are normally authenticated based on something they know such as passwords or PIN codes [1, ch. 4.5, p. 234]. The definition of authentication is related to, but not identical to authority and identification. While identification is the ability to check if the user is a unique user in the system, authentication is the ability to prove that the user is genuinely who that person claims to be [1, ch. 4.5, p. 245]. For example, when an user logs in with userId and password, identification is the action to verify if the userId is uniquely existed in the system, whereas at the same time, authentication is the action used to verify if the password match with the userId stored in the system. Authorization usually takes place after users are identified and authenticated. Authorization is the action to verify the right to access the information or to perform some actions that the system grants to the user [2].

2.3 Encryption

Encryption is a process of encoding a message or information in such a way that only authorized parties can access it. If the data is encrypted, only authorized users have the ability to read it in the form of plain text. Normally encryption does not itself prevent the information from being leaked and stolen, however, it will make sure that if the private information is stolen by an unauthorized user, the leaked data here cannot be read normally as an authorized user can. There are two main encryption techniques:

- Symmetric key: in this scheme, authorized users will receive the same key as the data provider in order to achieve secure communication.

- Public key: in this scheme, the encryption key is published for anyone to use and encrypt messages. However, only the receiving party has access to the decryption key that enables messages to be read. Every public key needs a certificate, which is also called a digital certificate. The certificate includes information about the key, information about the identity of its owner and the digital signature of an entity that has verified the certificate’s contents signed by a trusted Certificate Authority (also called an issuer). If the signature is valid, and the
software examining the certificate trusts the issuer, then it can use that key to communicate securely with the certificate’s subject.

Unlike encryption which is a two-way function where we can get the plain text data from the encrypted one with the right key, hashing is a one-way function to scramble plain text with arbitrary size into a fixed size one. As all the hashed text have a fixed size, there always collisions that two different passwords have the same hashed value. The best hashing algorithms are designed so that it’s nearly impossible to find the collision. So if the attackers get the hashed data, such as the hashed version of a password, they have no other choice but to try all the possible inputs to the hash functions and compare the results to the hashed password to get the un-hashed one.

2.4 Sniffer attack

A sniffer is an application or device which is usually used by hackers to read, monitor, and capture network data exchanges and read network packets. When data is transmitted across networks, if they are not encrypted, the hacker can use a sniffer to read/capture this data.

3 Current system architecture of Skolrutiner system

In this section, we will look at how the current system is designed, which technologies are used. The detail of some frameworks which are used in the development process of Skolrutiner will also be given in this section.

3.1 Overall design and technology

The server of the Skolrutiner system is implemented in C# and it applies the .NET Entity Framework in order to make the structure of the system clearer. On the client side, the system is implemented with a very common framework, AngularJs. In the next sections, we will look at some information about the .NET Entity Framework and AngularJs which are the technologies used in the implementation process of the Skolrutiner system.
3.2 .NET entity framework

.NET is a software development platform developed by Microsoft. The framework was meant to create applications, which would run on the Windows Platform. The first version of the .Net framework was released in the year 2002. This version was called the .Net framework 1.0. The .Net framework has come a long way since then, and the current version is 4.8. The .Net framework can be used to create desktop applications, web-based applications or web services. The framework supports different programming languages such as Visual Basic, C#, C++ and F# for developers to choose between.

.NET Entity Framework was first released by Microsoft in 2008. It is an open source Object Relational Framework (ORM) that enables .NET developers to work with a database using .NET objects. It eliminates the need for most of the data-access code that developers usually need to write. The main purpose of the Entity Framework is to be a programming model that attempts to lessen the gap between database constructs and object-oriented programming constructs. In most ORM frameworks, single object are mapped directly to the database schema. The Entity Framework has a more granular mapping layer so that we can customize the mappings which means that we can map a single entity to multiple database tables or even map many different entities to a single table.

LinQ stands for Language Integrated Query, it is a library that is integrated in .NET Framework 3.5. It is integrated in C# or Visual Basic, thereby eliminating the mismatch between programming languages and databases, as well as providing a single querying interface for different types of data sources. LinQ to Entities is basically the Entity Framework with LinQ library supported which makes it easier to write queries.

3.3 AngularJs

AngularJs is a JavaScript framework that makes it easy to build applications for the web. HTML was originally designed for displaying static documents. This becomes an issue when building web applications with dynamic contents. AngularJs is used to fix this problem. HTML pages are fulfilled with several HTML elements. The AngularJs framework works by reading the values of these HTML elements. Angular renders these values as directives to bind input or output parts of the page to a model that is
represented by standard JavaScript variables. The values of those JavaScript variables can be manually set within the code or retrieved from static or dynamic resources. By changing the values of these JavaScript variables, the contents of the HTML pages are also updated as all the HTML elements of the HTML pages are bound to the AngularJs variables.
4 File transfer for deployment

An important factor that every website manager have to think about is how to manage their code on the server. File transferring is one of the most important things when speaking about this. In our development process, we have to work with our own machine, not the server machine. In order to transfer the files between the two machines, the one you are working with and the server machine, there must be a protocol to establish a connection between them. In Figure 1, we can see some of the most common file transfer protocols and their default ports. In this section, we will go through all of them and discuss about their strong and weak points. After that, we will see which file transfer protocol is the most secured on for deploying a project.

4.1 FTP

When it comes to file transferring for deployment, FTP is probably the first method that comes to mind. FTP itself stands for file transfer protocol, it is an application layer protocol used to send and receive files via TCP/IP.
using ports 20 and 21 [3, Ch. 3, p. 174]. FTP is built on a client-server architecture using separate control and data connections between the client and the server. In FTP exchanges, a host running the FTP server accepts commands from another host running the FTP client. FTP clients come with a set of simple commands that make up their user interfaces. To transfer data, the client depends on an FTP server that is always waiting for requests. After a client connects to the FTP server, FTP data is exchanged via TCP, which means that FTP provides some assurance of delivery.

Many developer may be most familiar with FTP but this is not a secure file transfer protocol:

1. The password is transferred in plain text so it can be a vulnerability that can be exploited by the hackers.

2. Many TCP/IP connection must be established to: manage the connection, upload the files, download the files, list the files and list the folders. The firewall needs to implement new logic to prevent the attacks that aim at these connections.

3. The exchanged data cannot be encrypted. This is a huge vulnerability for any FTP user that any data exchanged through FTP can be intercepted and read by a packet sniffer (as known as a network analyzer, packet analyzer, protocol analyzer, or a sniffer).

In conclusion, if a system needs to comply with data security/privacy laws and regulations like HIPAA [4], PCI-DSS [5], SOX [6], and the EU Data Protection Directive[7], FTP will never be a good choice.

4.2 FTPS

As the internet becomes more and more popular, sending data without encrypting is not considered safe anymore, many laws such as PCI-DSS or HIPAA has been passed. FTPS is an extension of FTP that solves this problem. The data exchanging method of FTPS is the same as in FTP but FTPS adds a layer of security on top of FTP.

FTPS (also known as FTPES, FTP-SSL, and FTP Secure) is an extension to the previously described File Transfer Protocol (FTP) that adds support for the Secure Sockets Layer (SSL) cryptographic protocols. SSL stands for Secure Sockets Layer. FTPS uses public key encryption and relies on
a SSL certificate to be digitally signed by a trusted Certificate Authority. The secure variants of FTP include Implicit FTPS and Explicit FTPS. Both utilize SSL encryption.

4.2.1 Implicit FTPS

Implicit FTPS is the first method created to encrypt the data exchanged via FTP. With implicit SSL mode, an SSL encrypted session must be established via port 990 between the local client and the server for exchanging data. Any connection attempt made by a client without using SSL would be immediately dropped by the server.

4.2.2 Explicit FTPS

Explicit FTPS is the newer method of FTPS data exchange and has generally overtaken implicit FTPS use, except for some legacy systems. In explicit SSL mode, the client and server negotiate with each other for the level of protection used. This is very useful as the server can support both raw FTP and encrypted FTPS sessions on a single port. In an explicit SSL session, the client first establishes a traditional FTP connection on the standard port of FTP. Before sending the user credentials, the client cannot send any authenticated requests. Prior to sending user credentials, the client requests that the server switch the command channel to an SSL encrypted channel by sending the AUTH command. Upon successful setup of the SSL channel, the client sends the user credentials to the FTP server. These credentials along with any other commands sent to the server during the FTP session are automatically encrypted by the SSL channel. However, if the user prefer sending unencrypted data for better transferring speed, only the user credentials will be encrypted and the data will be transferred like FTP. This feature is called negotiation.

Without the negotiation feature, Implicit FTPS is actually more strict than explicit FTPS when it comes to establishing a secure connection. From a developer point of view, I prefer using explicit FTPS over implicit FTPS due to the flexibility of explicit SSL.

4.3 SFTP

SFTP stands for SSH File Transfer Protocol. It is often confused with
FTPS and vice-versa, although these protocols share nothing in common except their ability to securely transfer files. While FTPS adds an encryption layer to the FTP protocol, SFTP is an entirely different protocol based on the SSH (Secure Shell) network protocol. The SSH Protocol is a protocol for secure remote login and other secure network services over an insecure network [8]. Unlike FTPS, SFTP uses only one connection and encrypts both authentication information and data files being transferred.

There are two methods for authenticating the connections with SFTP.

- Using userId and password like FTP. Nevertheless, unlike FTP, with SFTP, these credentials are encrypted, which brings them to a higher security level than FTP.

- Using SSH key. The SSH key has two parts, a public key and a private key. In order to use the SSH key, the server has to send the public key to its trading partners. After receiving the public key from the server, the client uses it to establish an SFTP connection with the server. When the client connect to the server, the public key are also transmitted for authentication. If the public key match with the server’s private key, the authentication will be successful, otherwise, the request for connection will be turned down.

SFTP use a single key for both encrypting and decrypting the transferring data. This type of encryption key is called symmetric key encryption. This key is only used for the encryption and decryption of data, not for authenticating the connection. In comparison with FTPS, SFTP is performed in-line over the main control connection. By re-using the main connection, no other connections are opened between the local machine and the server when using SFTP, this results in a single secure, efficient connection through firewalls.

4.4 HTTPS file transfer

HTTPS is a secure version of HTTP (using SSL), it inherits the limitations (and the advantages) of HTTP.

HTTP file transfer is typically enabled and managed by clients. The client is responsible for initiating and managing an HTTP connection between the sending and receiving device using HTTP commands. Once the connection is established, the files can be transferred between the client and server.
HTTPS file transfer uses the same authorization method as FTPS. All the connections to an HTTPS server must be established over a secure SSL connection that encrypts and decrypts both the requests and responses. A slightly plus point for HTTPS file transfer is that although it uses SSL as the authorization method, only one port is needed to accomplish both the file transfer and the authorization which is normally port 443.

Nevertheless, the weakness of this type of file transferring is significant. Unlike SFTP, both HTTP and HTTPS are stateless protocols, they may require new connections for each transfer. This may lead to a problem when sending a large number of files at the same time. Also, uploads via an HTML form are generally limited in size because browsers time out, as it took a long time to upload big file, so mostly this protocol is used for downloading files.

One advantage of HTTPS over SFTP is that it is easier to use. Any user with a browser can use HTTPS for downloading data. On the other hand, if the user wants to use SFTP, they will have to download an SFTP client. Even though there are many free SFTP clients available nowadays, they require installation and a degree of familiarity with an additional tool that is not necessary when sending someone an HTTPS link.

In summary, if we want a secured file transfer that is easy to use for naive users who mostly need to download only, HTTPS will be a good decision. But if we want a more sophisticated file transfer, SFTP should be considered as a more suitable choice. And for large files transfer, we also should use SFTP.

4.5 WEBDAV/WEBDAVS

WEBDAV stands for Web-based Distributed Authoring and Versioning, it is an extension of HTTP and WEBDAVS is its secure version which runs over HTTPS. These file transfer protocols are different from most of the protocols we went through so far. Through WEBDAV and WEBDAVS, user can not only perform all the actions we can do with FTP but also have one more featured, to edit the file without downloading it. These two file transfer protocols are mainly designed for collaborative activities. WEBDAV or WEBDAVS is probably best suited for organizations that wants to distribute authoring capabilities, for example schools or universities.

However, as an extension of HTTP, WEBDAV/WEBDAVS also has the same weaknesses as HTTP such as the limited size for uploads or the problems with sending multiple files at the same time.
4.6 Recommendations

After researching the FTP, FTPS, SFTP, HTTP/HTTPS and WEBDAV/WEBDAVS file transfer protocols I find that SFTP is the most secure option. Even though the speed of SFTP is slower than the other protocols, if security is our first priority, the decreased speed is probably not an important factor. For the Skolrutiner system and other similar systems, the most suitable file transfer protocol is SFTP. SFTP is more secure than FTPS as it only opens one port ON the server and reuses it. And in comparison with HTTPS, SFTP is not restricted by the file size and it does not have the problems while transferring a lot of files at the same time as it does not need to open new for each file.
5 Protection against common attacks

In reality, a safe file transfer protocol is not enough to protect our website from all attacks. In this part, I will describe some common attacks, how they work and how the system should be designed to get rid of those vulnerabilities.

5.1 SQL injection

SQL injection is a code injection technique. The hacker takes advantage of the vulnerabilities on the server while checking inputs from the client by inserting malicious SQL queries. If the attack is performed, the hacker can easily get all the information from the server, insert new data, update or delete all of them. To perform a SQL Injection attack, the hacker will input a malicious string into the input fields.

In Figure 2, we can see an example code that is vulnerable to an SQL Injection attack. If these inputs are not verified or checked if it contains any malicious code, a malicious input such as `UserName=';DROP TABLE User;--` will make an awful change. In the example in Figure 2, the new SQL command with this input will be:

```
SELECT UserId FROM User WHERE UserName=';DROP TABLE User;--'AND Password = ";
```

The characters `--` is the signal say that the rest of the command is just the comment, which means the part of the command after these characters will not be considered as a part of the SQL execution and the result will be the table `User` will be deleted.
using (SqlConnection connection = new SqlConnection(ConnString))
{
    // Build the query statement using dynamic data.
    string sql = "SELECT UserId FROM User WHERE " +
    "UserName = '" + UserName + "' AND " +
    "Password = '" + Password + "'";
    using (SqlCommand cmd = new SqlCommand(sql))
    {
        cmd.Connection = connection;
        try
        {
            cmd.Connection.Open();
            var userId = cmd.ExecuteScalar();
        }
        catch (SqlException sx)
        {
            // Handle exceptions before moving on.
        }
    }
}

Figure 2: Example code that is vulnerable against SQL Injection

Some major causes of SQL Injection are from the mistakes or the negligence in the development process. One of them is incorrectly escaping characters. This happens when the server does not check if the inputs contain any malicious content. The attacker can use this vulnerability to insert the malicious queries through the input and the server will execute it unintentionally. To prevent SQL Injection attacks, there are two efficient ways: validating or parameterizing all the inputs. For input validation, the server should check for all the inputs in the request received from the clients before processing it. If there are any input contain malicious SQL commands, an error should be dumped and a response should be sent immediately back to the client to prevent further losses such as the execution of malicious queries.

For parameterization of inputs, the order of executing the SQL queries will be changed. In this way, the query plan is constructed on the server before the query is executed with parameter values. In other words, the execution of written query is planned to insert/update/select data from the databases and this is unchangeable after this stage. And from this, if the
parameters contain malicious SQL commands, they will still be executed with these malicious characters as a parameter, not a command.

```csharp
public List<Document>
    GetDocumentsByCategoryIdAndFolderId(int categoryId, int folderId)
{
    var sql = skolrutinerDbContext
        .Documents
        .Where(document => document.CategoryId == categoryId && document.parent == folderId);
    return sql.ToList();
}
```

Figure 3: An example from the Skolrutiner system showing how the server retrieves data from the database using LingQ.

The Figure 3 above shows how the current Skolrutiner system contacts and makes changes to the database. The SQL query generated from the LinQ statement in this figure is shown in Figure 4. In the generated query, the WHERE clause is parameterized automatically. When the query is executed, parameter @_linq_0 will be replaced with the value of categoryId and parameter @_linq_1 will be replace by the value of folderId. Therefore, it's impervious to attack with conventional SQL injection attacks. No matter what values a user provides as input to the search page, this query is perfectly safe and will not allow malicious input to make changes to the query original purpose on the server. If we input a malicious string for the SQL injection exploit, it will be treated as its literal, a string. In fact, the most harmful attack that a user could get with this query is to perform a brute force attack, an attack in which the attacker tries all possible inputs to get users’ credentials, which cannot be prevented by changing the way to query data.

When I tried to perform an SQL injection attack to the Skolrutiner system on login screen. I inputted ’;SELECT * from Users; to the password field and it resulted as in figure 5. There are also some validation from the client side when I tried to input some malicious code into the email text field in login screen of the system. In Figure 6, I entered ’ OR '1'=='1';select * from
Users;– and the validation message appeared and said that an incorrect form of email is entered.

```sql
SELECT 
  [Extent1].[Id] AS [Id],
  [Extent1].[FirstName] AS [FirstName],
  [Extent1].[LastName] AS [LastName],
  [Extent1].[EmailId] AS [EmailId],
  [Extent1].[OrganizationNumber] AS [OrganizationNumber],
  [Extent1].[PhoneNumber] AS [PhoneNumber],
  [Extent1].[Address] AS [Address],
  [Extent1].[City] AS [City],
  [Extent1].[ZipCode] AS [ZipCode],
  [Extent1].[Password] AS [Password],
  [Extent1].[Role] AS [Role],
  [Extent1].[Active] AS [Active],
  [Extent1].[CreatedDate] AS [CreatedDate],
  [Extent1].[PriceListId] AS [PriceListId]
FROM [dbo].[UserAccount] AS [Extent1]
WHERE ([Extent1].[EmailId] = @p__linq__0) AND ([Extent1].[Password] = @p__linq__1)
```

Figure 4: Generated SQL Query by LinQ from the statement in Figure 3

5.2 XSS

Cross-Site Scripting (XSS) is one of the most common types of attack on the internet at the moment. But this does not make it less dangerous. XSS is a type of injection attack, in which malicious scripts are injected to a trusted website. In this type of attack, the hacker can implement malicious scripts to get the cookie, the session token or any other important information from the victim’s client and continue using this information on the trusted website. The goals of XSS attacks are always the execution of malicious scripts on the vulnerable websites. These scripts can aim at transmitting private data, like cookies or other session information, to the attacker. Sometimes, they are also used to deface a website by changing the content of it. Even though there is a limitless number of attacks based on XSS, they can be classified into two categories: stored XSS attack and reflected XSS attack.
Inloggning misslyckades. Kontrollera att ditt konto
är godkänt och att din e-post och lösenord är
korrekt

batmannhinho@gmail.com

' OR '1'='1';select * from Users;--

LOGGA IN

Glömt lösenord?

Figure 5: Result of a SQL injection attack on the Skolrutiner system

' OR '1'='1';select * from Users;--

⚠️ Please include an '@' in the email address. " OR '1'='1';select * from Users;--
' is missing an '@'.

LOGGA IN

Figure 6: Example of input validation in the Skolrutiner system
5.2.1 Stored XSS attack

From Figure 7, we can see how a stored XSS attack works. In this type of attack, the attacker inserts malicious scripts to the database of the server using a form submit or an API (1). When another user accesses the website and makes a request (2) for a page that now includes the malicious script, that script will be sent to the client (3) and be executed on the victim’s browser (4) without any notice and the attack is successfully performed.

5.2.2 Reflected XSS attack

With reflected XSS attacks, the injected malicious script is reflected off the web server. The Figure 8 show step by step how a reflected XSS attack works. First, the attackers craft a URL of the vulnerable website. The chosen URL must be an API that some of the value of its parameters will be returned after being processed on the server. After choosing that specific URL, the attacker will write some malicious scripts and include it in the URL as the reflected parameter. After crafting the URL, the attacker will send (1) this URL to the victim and trick him to click in that link to send a request (2) to the server. When the server response, the response contains the malicious scripts also (3). The browser will execute these malicious script (4) without any contest as it comes directly from the server, a trusted source. The scripts
can aim at sending the private information of the victim to the attacker. The consequence of the attack based on the purpose of the scripts.

5.2.3 Preventing XSS attacks

In order to prevent an XSS attack, we should escape all dynamic content coming from the data source, so the browser knows it is to be treated as the contents of HTML tags, as opposed to raw HTML. Escaping dynamic content generally consists of replacing significant characters with the HTML entity encoding such as replace ‘<’ with ‘&lt;’ or replace ‘>’ with ‘&gt;’. Escaping editable content in this way means it will never be treated as executable code by the browser. The technique to escape all dynamic content is also called sanitization.
In Figure 9, we can see an example of how the data are displayed on a normal HTML page in the Skolrutiner system. As mentioned in the first part of the thesis, this website, Skolrutiner, uses AngularJs on the client side as the tool to communicate with the backend. In Figure 9, to print out the title on the web page, a normal Angular script is used to bind the value of the title to the `<a>` tag: `{{x.Title}}`. In AngularJs, this way of binding the value of data into an HTML tag always escapes interpolated content such as script tag.

The code in the Figure 9 is the way the system display the list of existing categories. To test the ability to prevent an XSS attack of AngularJs, we can create new categories with JavaScript code as an input value to perform an XSS attack. Here I input a malicious value: `<script>alert('XSS')</script>`. If the attack is successful, there will be an alert on the browser and an image tag will be displayed at the position of categories. The result of the attack is shown in Figure 10. In order to view the page source, I tried using the Google Chrome Inspect feature but in the end I only see the HTML elements after it is read and translate into normal text by the browser. That is the reason why I have to use a chrome extension called View Rendered Source [9] to see if the output is sanitized or not. The result of the extension is shown in Figure 10b. From this figure, we can see that the script I injected has been escaped and encoded into HTML entities which is treated as raw text and the script is displayed as a normal HTML text in Figure 10a. From this figure, we can see that all the JavaScript codes are displayed on the HTML page instead of running without the permission of the user. From this, it is easy to see that AngularJs has escaped all the JavaScript code we inserted.
(a) What's rendered on the webpage

```html
<script>alert('XSS')</script>
```

(b) Inspected HTML element

```
<li ng-repeat="x in categories"
    class="item ng-scope">
    <a ui-sref="bulletin.kategori({id: x.Id})"
       class="list-group-item ng-binding" ui-sref-active="active" ui-sref-active-parent="parent" href="/bulletin/kategori?id=43">
        &lt;script&gt;alert('XSS')&lt;/script&gt;</a>
</li>
```

Figure 10: Result of an unsuccessful XSS attack

before so clear that using Angular. With this experiment, we can say that using AngularJS on the client for displaying information is a good way to escape JavaScript code in order to prevent an XSS attack.

## 5.3 CSRF

Cross-Site Request Forgery (CSRF) is an attack that forces the victim to send a request unintentionally to a website which they are authenticated. The request inherits the identity and privileges of the victim to perform an undesired function on the victim’s behalf. Normally, most of the user’s credential including the session cookie and IP address are automatically included in the header of the request. As a result, the browser will have no way to distinguish between the forged request sent by the victim and a legitimate request sent by the victim.

When using CSRF, forcing the victim to retrieve any data, which is similar to XSS, is meaningless for the attacker as the attacker doesn’t receive the response, the victim does. As such, CSRF attacks target state-changing requests, such as changing the victim’s email address and password or purchasing something, not theft of data, since the attacker has no way to see the response to the forged request. This contrasts with XSS which aim at sending the information of the user to the attacker’s site. In fact, in order to make a CSRF attack effectively, it is often combined with a stored XSS attack or social engineering.

To protect a website against a CSRF attack, the most secure method is
to use a hidden CSRF token which is generated uniquely by the server for each session or even each request. This token is a large unpredictable value generated by a cryptographically secure random character generator. It is normally stored in the session of each authenticated user. It is also transferred to the client and stored in a hidden field so that it can be included in each request sent from the client. A POST request which is sent to the server must include this hidden token value in the header or the body of it. And for a GET request, the token is placed within the URL or in the header of it. As the cookies are sent automatically with the requests, these tokens are not. That is why the attacker cannot fake this token to perform a request.

With Skolrutiner, any logged in user will receive a unique token that is stored in the session and only last for that session. This token is a random, unpredictable string and it is added with some more fixed character at the beginning. Any time AngularJs wants to send a POST or PUT request to the server, it will need to read this token and send it along with a custom HTTP header. Because these operations (reading the session cookie, setting the header) can only be done on the same domain of an AngularJs application, we can make sure that this can only be done by a real user who is using our AngularJs application. Figure 11 shows how a csrf-token looks like. In this figure, Authorization in the parameter name in the header which is used to place the CSRF token, the highlight part is the token generated and Bearer is just a mark for the type of the CSRF token [10]. This a typical way to avoid users to be the victim of a CSRF attack and also being used by many other websites at the moment.
6 File storage

With cloud computing well into the mainstream and gradually replacing roles currently filled by the traditional server architecture, it’s time for us to ask a question if cloud hosting could replace traditional file servers for storing and sharing files in a system. In this part of the thesis, we will take a look at benefits provided by each type of storage, not only from the security point of view but also from the perspective of cost, speed, maintenance, etc. In short, everything should be considered when choosing storage. Also, we will have a comparison between some common cloud storage at the moment and a conclusion about choosing storage based on the requirements of the system.

6.1 Local server storage

Local file servers have been the mainstays and workhorses of the business world practically since business networking has been a term. With local server storage, everything which is uploaded to a server from a client will be stored to a designed location which also lies in some part of the server itself. There are a significant number of advantages to keeping our files on an internal (or externally managed) server that we control and we will go through all of them one by one now:
• **Total control**: The biggest advantage that using a local server storage provides us is the ownership of the files on our server. If our server runs on our physical machine, no one can have access to these files without our permission. Even if we are on a shared hosting, we still control all of the contents on the server, and the hosting company cannot erase our data or otherwise breach the contract we signed before using their hosting service.

• **Speed**: As all of our files are stored on our server, no new connection to an external server would be created to get or save a file. As a result, the speed needed for downloading and uploading files is much shorter than using cloud storage where we have to establish a new connection to the cloud hosting service.

• **Compliance**: Regulatory and compliance laws may not allow certain types of sensitive data (medical, government, financial) to be stored in the cloud. In contrast, with our own in-house server, we can configure it the way we want: How we want files served, how credentials will be assigned, where people will be able to access files from, etc. Managing the file system ourselves gives us flexibility and allows us to create our own access rules.

On the other hand, using a self managed file server also has some disadvantages. As all of the files are stored on the same server as the implementation of the system, the system will need a high speed processor, a big amount of RAM and storage in order to manage all of the files. Whether our host is on a shared hosting or an in-house machine, it is always a huge problem to scale up and down as demand rises and falls. In fact, if we use local server storage, we will lose more money than what we need in the slow period and we can also see that we are running out of bandwidth, RAM, or storage space during a busy one. Another weakness of local server storage is that we will have to configure all the security by ourselves which will take us a lot of time. And if our server is under attack, the restore process is going to be a really big problem.

### 6.2 Cloud storage

In order to understand what cloud storage is, the first question we need to answer is what cloud computing is. According to the National Institute of
Standards and Technology, “cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be easily provisioned and released with minimal management effort or service provider interaction.” [11].

Cloud storage is a service model in which data is maintained, managed, backed up remotely and made available to users over a network. The cloud storage providers are responsible for keeping the data available and accessible, and also keeping the physical environment protected and running. Cloud storage users usually have to pay to the storage capacity of the providers by the amount of data consumption, monthly rate or yearly rate.

The places where the uploaded files are physically stored are called the data centres. Everything we uploaded to a cloud storage is stored on dedicated servers and storage volumes domiciliated in vast warehouses which are often placed on campuses full of such warehouses. Data centres are owned by cloud service providers, who are responsible for keeping the servers up and running. The job of these data centres is to keep our data safe from data thieves or the destruction of data. They always have at least one data backup system generator in case of power outage to make sure that our data is always under protection.

Recently, enterprises are increasingly adopting cloud storage options. We will look at some advantages of the new technology brings to us and let’s see if they are suitable to replace the traditional local server storage.

- **Security.** With the development of cloud computing nowadays, there is a large number of trusted providers with better knowledge of security than most of the developers. Whenever we decide to use cloud storage provided by a third party organization, they will determine how access is granted and where people can access the server from. Normally setting up the security for a server storage takes us a lot of time. By using a cloud service, we will save a lot of time as the cloud service provider will do that for us.

- **Low expense.** While using the cloud storage, we will not have to pay for the onsite hardware or capital expenses as mentioned in the disadvantages of local server storage. Even though we still need to pay for the amount of data we store in the cloud, but in my opinion, it is still a very tiny number in comparison with the amount of money we
have to spend on the onsite hardware or capital expenses if we choose to use local server storage.

- **Flexibility in the amount of storage.** Unlike local server storage, cloud storage does not have a limit on the amount of data stored. Normally, the user and the provider will sign a "Pay-as-you-go pricing" contract. And when the amount of data we stored dynamically grown, the cloud service providers will take care of that issue for us.

- **Data protection.** Most of the cloud providers have at least one data backup system so that we can use their service without thinking of the threat of losing our data. This will be, most of the time, included in the contract when we register for using the cloud service of the provider.

In spite of the fact that cloud storage has so many advantages, there are some disadvantages of this type of storage that make it not suitable in a number of cases:

- **Privacy.** All cloud storage users should be aware that they are giving sensitive business data to a third party cloud service provider. To mitigate these concerns, we should choose a cloud storage provider that encrypts our data before storing it on their cloud. However, even if these sensitive data are encrypted, it is still in the hand of a third-party company and nothing can make sure that the encrypted data cannot be decrypted by someone with bad intentions.

- **Dependency.** As the data is stored on another machine, the speed of the service using cloud storage depends heavily on internet speed. If there are some problems with the internet connection, the performance of our system will be reduced significantly.

- **Backup and restore.** Although most of the cloud storage providers include backup and restoring features as a part of their services, restoring a large amount of data is still very time consuming even with a strong internet connection.

### 6.3 Recommendation

When I look into the implementation of Skolrutiner, the system stores every document of the users on its local server storage which will result in
a very bad situation when the number of user grows and the amount of stored documents grows. We will have to purchase new storage for storing documents, make sure that our processor is strong enough for managing a large number of files.

From a developer point of view, a combination of both self-hosted storage and cloud storage would be the best choice as we can utilise the advantages of each methods and not be limited by the disadvantages of each method. My recommendation is using the cloud storage to store normal documents uploaded by the customers. And for some important and sensitive documents, local server storage would be a more suitable choice as these documents will not be reached by a third party company. A cloud storage that encrypt your data could also be a idea for storing this type of documents but it is still not as good as the local server storage. In this way, we will not have to care about protecting the non-sensitive documents of the customers, it will also save a lot of money as we do not have to purchase hardware for self-hosted server, and we do not worry about a third party company will reach the sensitive documents between the Skolrutiner system and the customers.
7 Conclusion

In this thesis, I have tested if the Skolrutiner web application is vulnerable to SQL injection, cross-site scripting and cross-site request forgery. My investigation has concluded that the technology used to implement the Skolrutiner system is completely safe against these attacks.

And for anyone who is developing a web application, the thesis provide an instruction on how to have a secured developing process. In order to have a secured deploying process, using a secured file transfer protocol is needed and I strongly recommend using SFTP for deploying a project onto the server. Also, some solutions such as parameterization, sanitization or using csrf tokens should be used in order to prevent common cyber attacks like SQL injection, XSS and CSRF. And last but not least, by combining both cloud storage and self-hosted storage, we can make full use of all the advantages of them for saving the cost of storing the documents by using cloud storage and saving sensitive documents on local server storage.
References


