Cross Website Data Protection

Ramdas Karthikeya | Yashwanth Reddy Anugu | Siva Sai Praveen Dantam

1 Introduction

Data protection from third-party systems has become a headache in recent years. Until and unless someone examines the source code of the software and how it works, the implementation of a third-internal party's system is unknown. Despite the fact that the corporation fulfills all data compliance laws, there is still some doubt regarding the software's trustworthiness. As a result, data security is critical, particularly when working with external sources.

All of the variables that the remainder of the web page has access to are accessible to Malicious JavaScript. This includes access to the cookies of the user. Session tokens are frequently stored in cookies. If an attacker obtains a user's session cookie, they can mimic that user, undertake actions on their behalf, and get access to sensitive information. JavaScript can read the DOM of a browser and make changes to it. Fortunately, this is only available within the JavaScript-enabled page. The XMLHttpRequest object in JavaScript can be used to make Requests containing any data to unexpected locations. HTML5 APIs can be used by JavaScript in contemporary browsers. It can, for example, access the user's location, camera, microphone, and even personal information.

Our project's purpose is to use data encryption and decryption methods to protect data sent from one website to another. The overall goal is to create two distinct websites as well as a server. It's referred to as the main website (ex: main.com) and the third-party website (ex: third-party.com). The main website features a form with a few fields that contain the user's personal information, such as First Name, Last Name, Date of Birth, Gender, and Email. The user next clicks the submit button after entering these details. After the user submits the form, we link them to a third-party website where they must enter their SSN number and a random score is generated for the use case of credit score. Once the user enters their SSN and the credit score is generated, the user clicks submit to send the information entered on the main website and the score generated in the third party website.

Protecting information entered on the main website and hiding plain text information from the third-party website is a crucial aspect of this implementation. We make a call to the server after the submit button on the main website is hit to send encrypted data of the personal information as a single string of all information. We redirect the user to the third-party website after the encrypted data is delivered back to the primary website. The encrypted data is saved in session storage or a cookie on the third-party website so that it can be transferred to the server once the score is retrieved. We are preventing unauthorized access to user data in this way from being accessed by the third party website.

2 Design

Our application has a Front end server and a backend Server.

2.1 Front End (Tech stack: React.js, React Router)

Front end application consists of 2 websites. React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript toolkit for creating UI components-based user interfaces. Meta (previously Facebook) and a community of individual developers and businesses maintain it.

React router is required to direct from one website to other website.

Info site

In the info website the user enters the required personal information and sends it to the third party website. It is the main landing page and this is the first page we

Credit Score Generator site

In the credit check website once the request from the info website is received , it asks the user for their SSN number and generates their credit score and displays it to the user

Front end server is run on React-Node server on Port 3000

2.2 Back End (Tech stack: Node.js, Express)

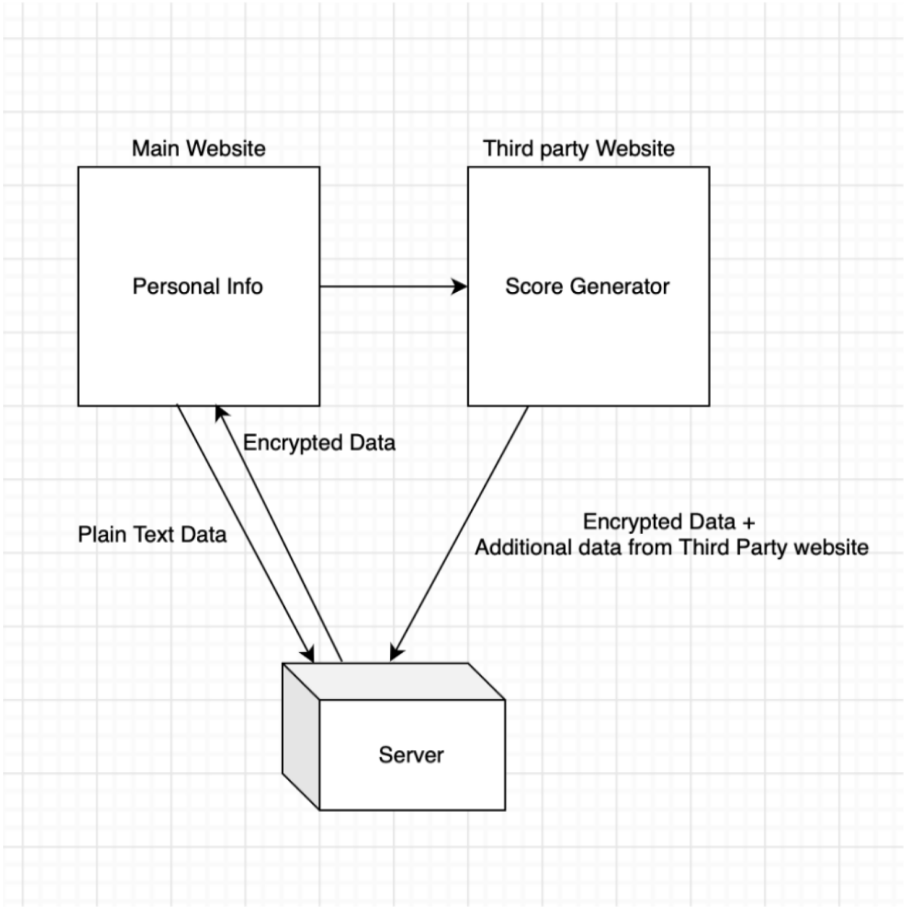
Node.js is a cross-platform, open-source back-end JavaScript runtime environment that uses the V8 engine to execute JavaScript code outside of a web browser.

Express is the HTTP middleware used in Node.js to connect make and connect Http requests.

The backend application consists of a server spun on Node.js on Port 5001. Server accepts requests from the front end for the purpose of encryption and decryption. For the encryption and decryption of the user data we have used the AES algorithm. We

developed the AES algorithm using the crypto module of node js , the module provides various cryptographic functionalities that include a set of wrappers for ciphers, deciphers, HMAC and verify functions.

3 Application Architecture



4 Cryptography

4.1 Encryption

The main aim is to encrypt user data entered in Info website form Credit Check website for this purpose we used AES Encryption, as AES proven itself to be a reliable and effective method of safeguarding sensitive information. For Encryption we used Crypto module in Node.js

We used an inbuilt application programming interface of the crypto module[2] which is used to create a cipher object, with the stated algorithm, key and initialization vector(iv). For generating the initialization vector we have used randomBytes

Crypto.createCipheriv(algorithm, key, iv[, options]):

Creates and returns a cipher object, with the given algorithm, Key and Initialization Vector Algorithm: aes-256-ctr, Initialization Vector : iv = crypto.randomBytes(16)

Encrypt function:

```
const AES256 = 'aes-256-ctr';
const key = 'vOVH6sdmpNWjRRIqCc7rdxs01lwHzfr3';
const initialisationVector = crypto.randomBytes(16);

const encrypt = (data) => {

    const cpr = crypto.createCipheriv(AES256, key, initialisationVector);

    const data_encrypted = Buffer.concat([cpr.update(data), cpr.final()]);

    return {
        initialisationVector: initialisationVector.toString('hex'),
        hashedData: data_encrypted.toString('hex')
    };
};
```

4.2 Decryption

Once Encrypted data is received from the Info website , the Credit Check website generates the credit score of the user and sends it along with the encrypted message in the form of a cookie session. The server decrypts the messages once the credit report is sent along with the encrypted data to get the information of the user

Crypto.createDecipheriv(algorithm, key, iv[, options]):

Creates and returns a decipher object, with the given algorithm, Key and Initialization Vector Algorithm: aes-256-ctr, Initialization Vector : iv = crypto.randomBytes(16)

Decrypt function:

```
const decrypt = (data) => {

    const dcpr= crypto.createDecipheriv(AES256, key, Buffer.from(data.initialisationVector, 'hex'));

    const data_decrpyted = Buffer.concat([dcpr.update(Buffer.from(data.hashedData, 'hex')), dcpr.final()]);

    return data_decrpyted.toString();
};
```

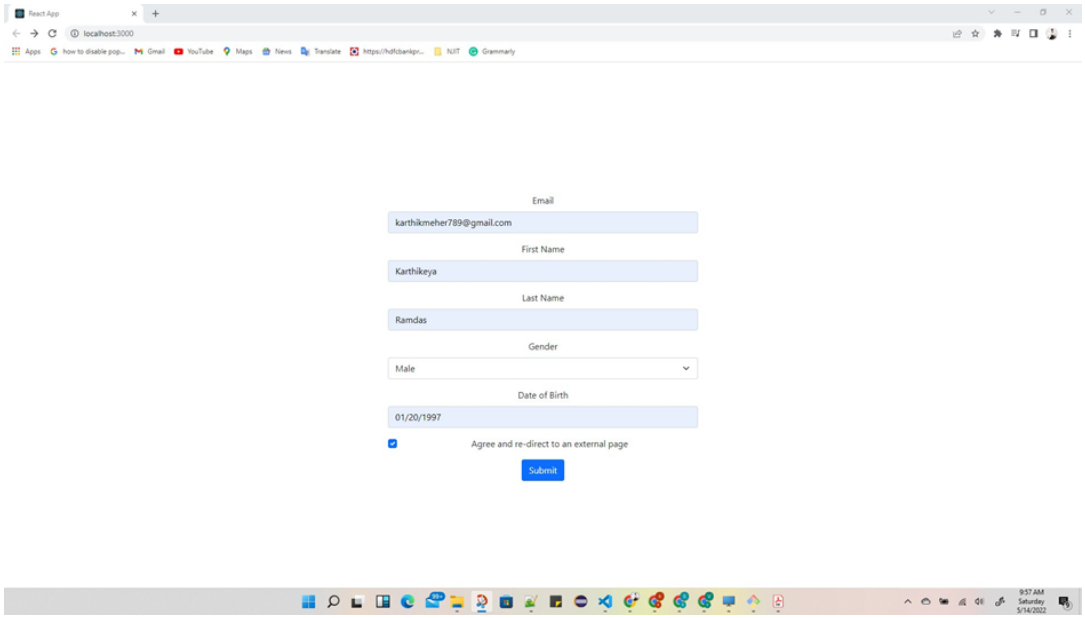
5 Application Flow

5.1 Info site

In the info website the user enters his personal information like Email, First name, Last name, gender, Date of Birth. The user has to select the checkbox that they agree to redirect to an external page. On clicking the submit button, a Validation is run on the Client side. The form data is validated.

The Validation rules we followed are:

- Email Format Validation
- Non Empty First Name
- Non Empty Last Name
- Valid Date of Birth (Valid Month, Date and Year values)
- Agreeing to the condition to redirect



Once the user clicks the submit button, the user details in the info page will be sent to the by calling the Encrypt API (localhost:5001/encrypt). The API returns the user data encrypted as a single string which we call as the token. This token will be sent to the credit score generator website as a query parameter.

An example encrypted token looks like this:

468f8f673fbd1df103c8afa87e4e4f3d211aa95984895cdedde503001d4054fb8eeb99a099e1d786e9cbccb
ff9594130111e74d9ef82b5452ae3cf354367dc3b814f9532dce1b2683374bb82ef2186d80c504634cee4e
6f2e7280f7980954595860801ee|14ee20d6c354820469f7046557f6194f

5.2 Credit Score Generator Site

The url of the website looks like this with the token as a query parameter in it.

http://localhost:3000/score/468f8f673fbd1df103c8afa87e4e4f3d211aa95984895cdedde503001d4054fb8eeb99a099e1d786e9cbccbff9594130111e74d9ef82b5452ae3cf354367dc3b814f9532dce1b2683374bb82ef2186d80c504634cee4e6f2e7280f7980954595860801ee|14ee20d6c354820469f7046557f6194f

In the credit check website the user is asked to enter SSN to generate credit score and a random credit score is displayed for the user.

SSN Id Input field

The Validation rules we followed are:

- The SSN Id entered Should be a number
- The Id should be of length 9.

Algorithm to generate random score:
Math.floor(Math.random()*700)
This generates a score between 0 to 700.

Once the user clicks submit we call the decrypt API (localhost:5001/decrypt) by sending the token and the SSN Id and the score included to the Backend.

The decrypt function takes the token from the payload and decrypts the data using the crypto function and displays the decrypted data.

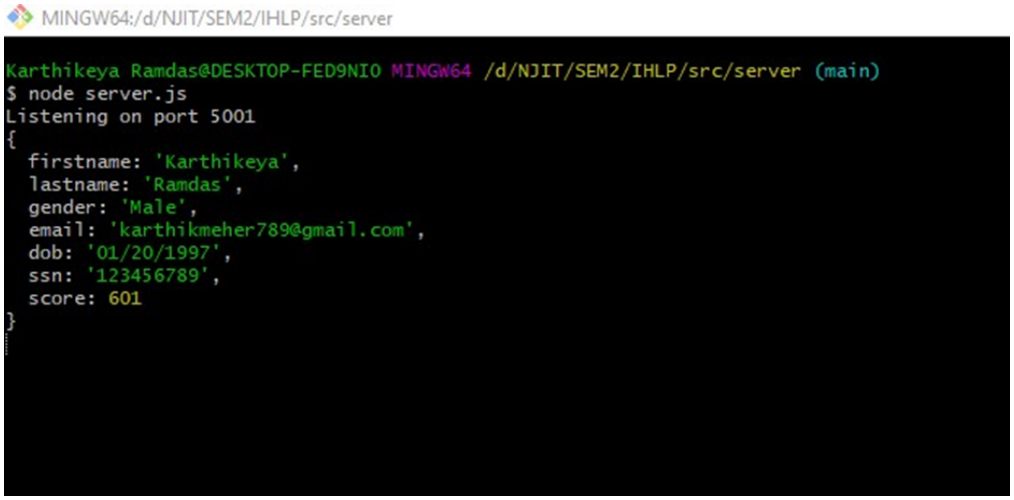
6 Results

The information encrypted is sent as a Query Parameter , the Query Parameter is as follows, as we can see below all the personal information of the user is encrypted and sent as a single query parameter

http://localhost:3000/score/0be52e5e0532cd2d7c4516ce313dfd0ccdc957d98
26be970d647f8d31ddb37da59e159b2bdea742bbe20679e02910093748b5a78
e868286aeb6595ca9553786ddd0473a7f6387f3389ab1c5de7e4566fc1169803
aebb943fe2%7Cabba72d9b7fe6663664bced7e32d2ed5

Once the server receives the encrypted information from the Credit Score generator website, the server decrypts this information and displays it.

In this way we are able to encrypt and decrypt the information by hiding the plain text information from other websites.



7 Conclusion

In this way we are protecting the Person Identifiable Information by encrypting the data when redirecting to 3rd Party web-pages. The user Information is safe at all times and in practice these standards are much easier to follow and adapt by every website.Applying these concepts in real time is relatively easy and has being employed by many applications in practice. AES algorithm effectively works for this purpose and is relatively stronger regarding the crypt-analysis compared to other algorithms like DES.

8 References

1.Node.js Documentation.
2.Crypto Module Documentation.