SECURING AN E-COMMERCE WEBSITE

BCI3004 - Security of E-Based Systems

FACULTY: PROF. ANURADHA D

SLOT: A1

by

18BCI0179 - Shourya Maheshwari

18BCI0190 - Laksh Gupta

18BCI0197 - Nimish Shah

18BCI0214 – Mihir Srivastava

18BCI0236 - Kakad Aryaman Sandesh

School of Computer Science and Engineering

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

May 2021

Contents

1. Abstract

We have built an e-commerce website and have included many security algorithms to secure our website. In this website customers will be easily able to buy products and can compare the price of different sellers and can buy according to price in which they find more comfortable. Supervisors and admin will have separate login ID and pin through which they can access their account to modify their inventory, which can be done in real time giving the customer an accurate view of their inventory.

2.1 Introduction

We plan to build an e-commerce website and then secure it using the many security methods we have learned in this course.

2.1.2 Website purpose

The project allows users to check the inventories of nearby shops. Easyshop is the place where customers can view their day to day products and order them online instead of going there physically. In this project, operators enter the Website and buy some product, we then look at their cart and compare it to all nearby shops available, we then generate the bill for every shop and allow the customer to choose which shop they want to order from. Supervisors and admin will have separate login ID and pin through which they can access their account to modify their inventory, which can be done in real time giving the customer an accurate view of their inventory.

2.1.2 Website scope

We wish to show the users all the shops close to them and allow them to search for the products that they want through the options provided on our website. They should be able to create a cart that contains all the items they wish to buy. Then the website should show the user a comparative analysis of the prices at each shop and the approximate final price for the whole cart for every shop available nearby. The website should also notify the user if a product is not available.

There is also a feature for the customer to send their list to the shop so that the shopkeeper can keep the items ready so that the customer can only pick the groceries quickly.

3.1 Methodology

Implementation can be done in any language of choice. Only requirement is that you will be handling data in JSON format.

1. Initially that client and server will have constant Public Key Pair. When a client enters the authentication data (email, password), the data can be encrypted with the public key on the client side.
2. The server now decrypts the data and will generate a lucky number based on it (private key for server in diffie hellman protocol). Server will also generate a JSON Web Token which will be used later to authorize client requests.

   eg Response:

   {

   “serverPublic”: 123,

   “token”: “aE12.dhhh.0wdfE” ,

   “timestamp”: 1600000000

   }
3. The client stores it in the session and for every subsequent request made by the client, this data will be used.
4. Client now generates a Nonce (Number Only used Once), this nonce will be private key for client in Diffie hellman protocol.

   Client then calculates its public key and shared secret key,

   eg: clientPri = Nonce

   clientPub = generateDiffiePub(clientPri)

   sharedKey = (serverPublic^clientPri) % p
5. The shared secret is now used in PR-Pass to generate Symmetric Private Key.

   eg,

   symmetricKey = prpass(token, sharedKey)
6. The symmetricKey is used in AES to encrypt the request body altogether.

   eg,

   req.body = AES(req.body, symmetricKey)
7. Generate the HMAC of the information with the symmetric key generated by PRPass.
8. The data is then sent to the server, with clientPublic and token as headers in the request
9. Server then further does the same operations i.e. generating sharedKey, symmetricKey and will then decrypt req.body and verify HMAC and will further process it according to business logic and give appropriate response.

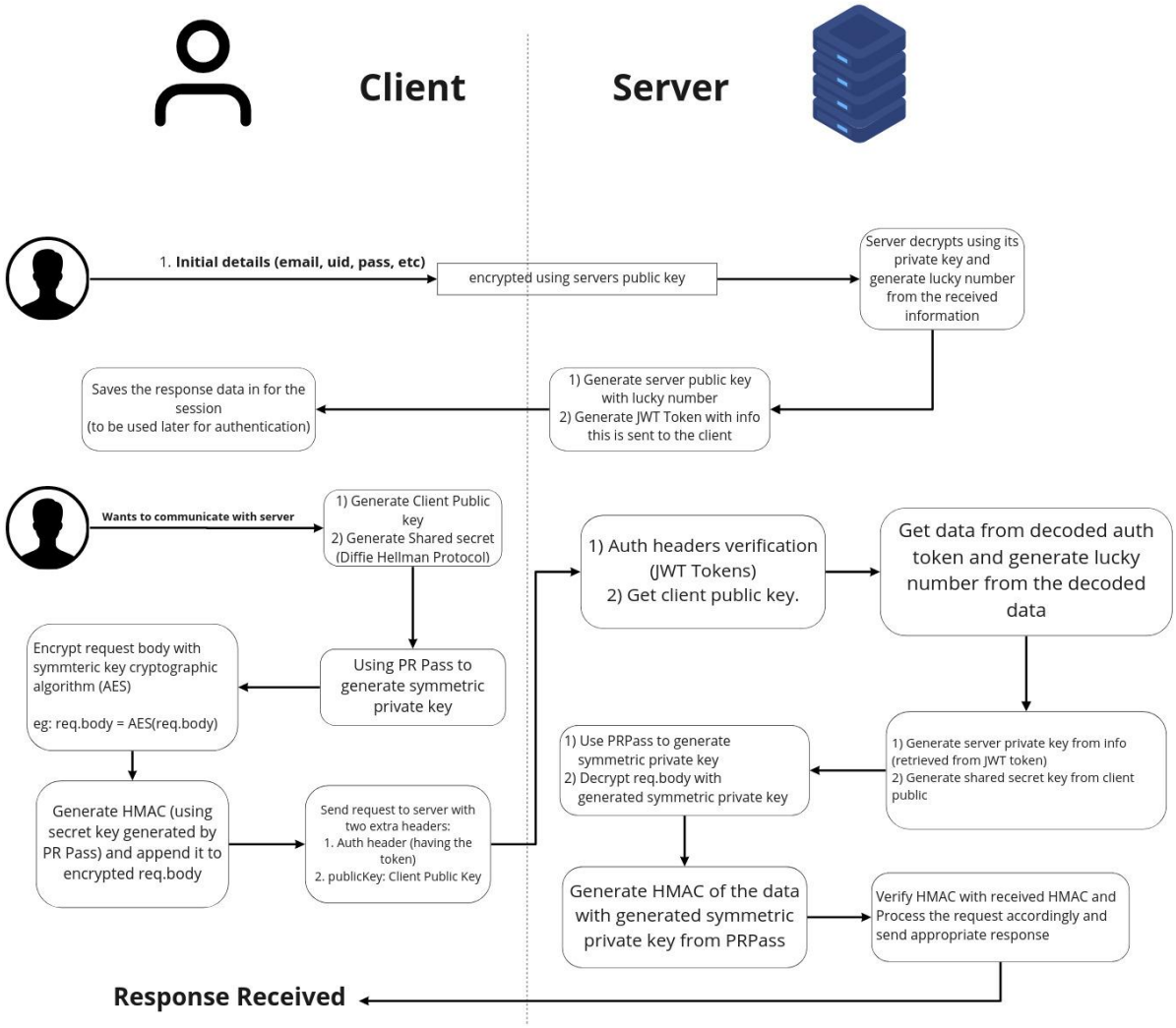


Fig. 3.1 Security Architecture of the website

Website design

Home Page of the website:

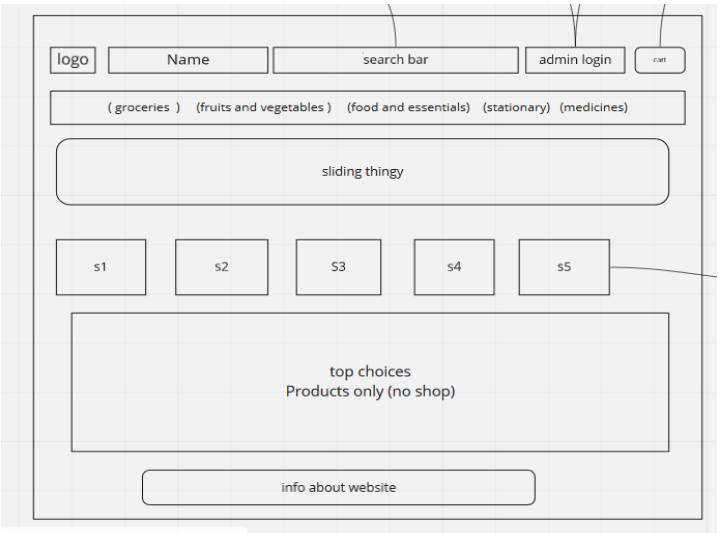


Fig. 3.2 Home Page

This is homepage of the website, this is where user will come on entering the website

Product popup page:

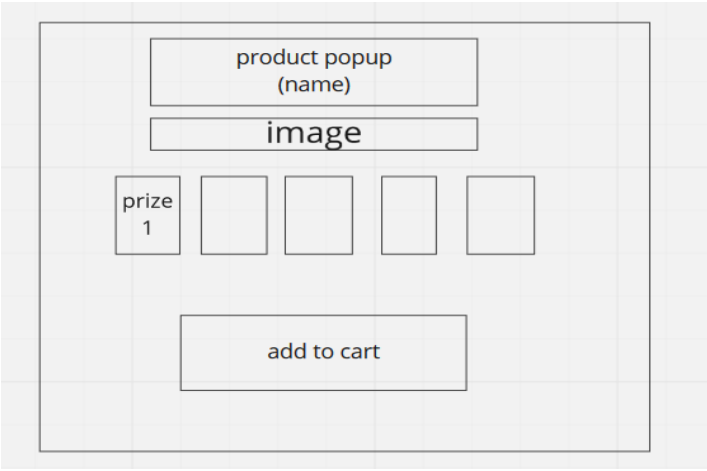


Fig. 3.3 Pop-up Page

On searching for a product, this page will come. Here, users can select products to add in the cart.

Shop page:

Fig. 3.4 Shop Page

The page shows information about the shop. It will have highlighted products and such.

Cart page:

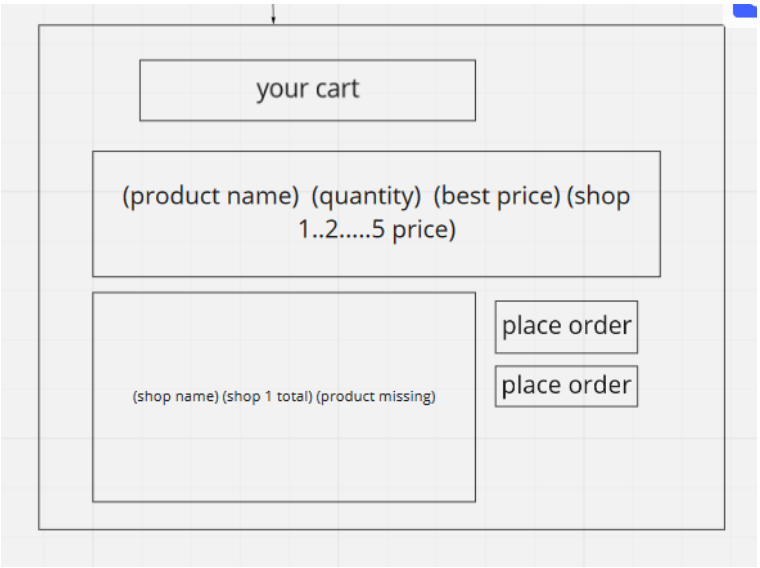


Fig. 3.5 Cart Page

The shopping cart, where user will see selected products

Search Page:

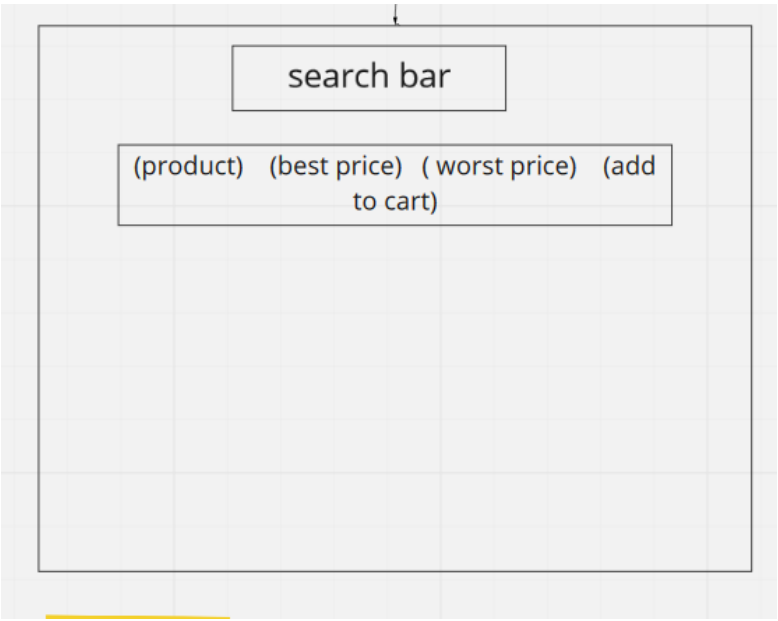


Fig. 3.6 Search Page

Searched results show up on this page line by line in tabular manner like on a e-commerce website.

Admin Login:

Fig. 3.7 Admin login Page

This page is restricted to only shopkeepers. They can manage their inventory with this page

Admin edit inventory page

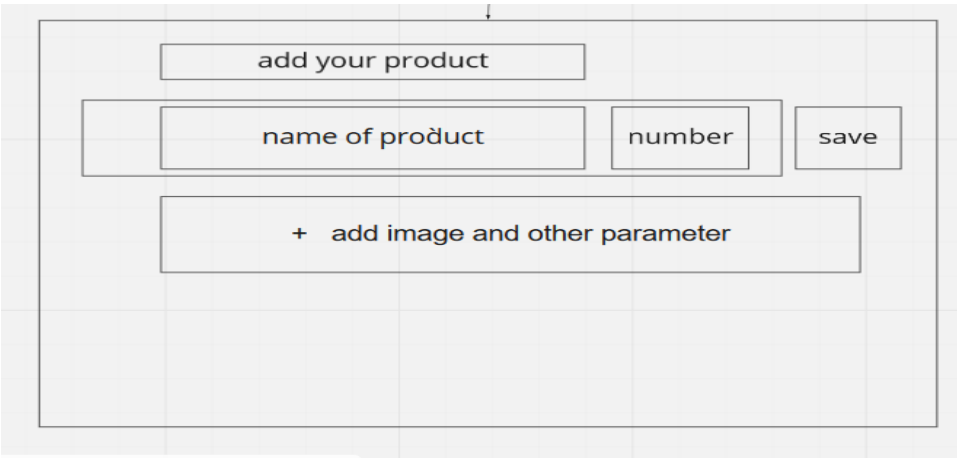


Fig. 3.8 Admin Panel Page

Shopkeepers can edit the amount of product left in the inventory.

Overall design

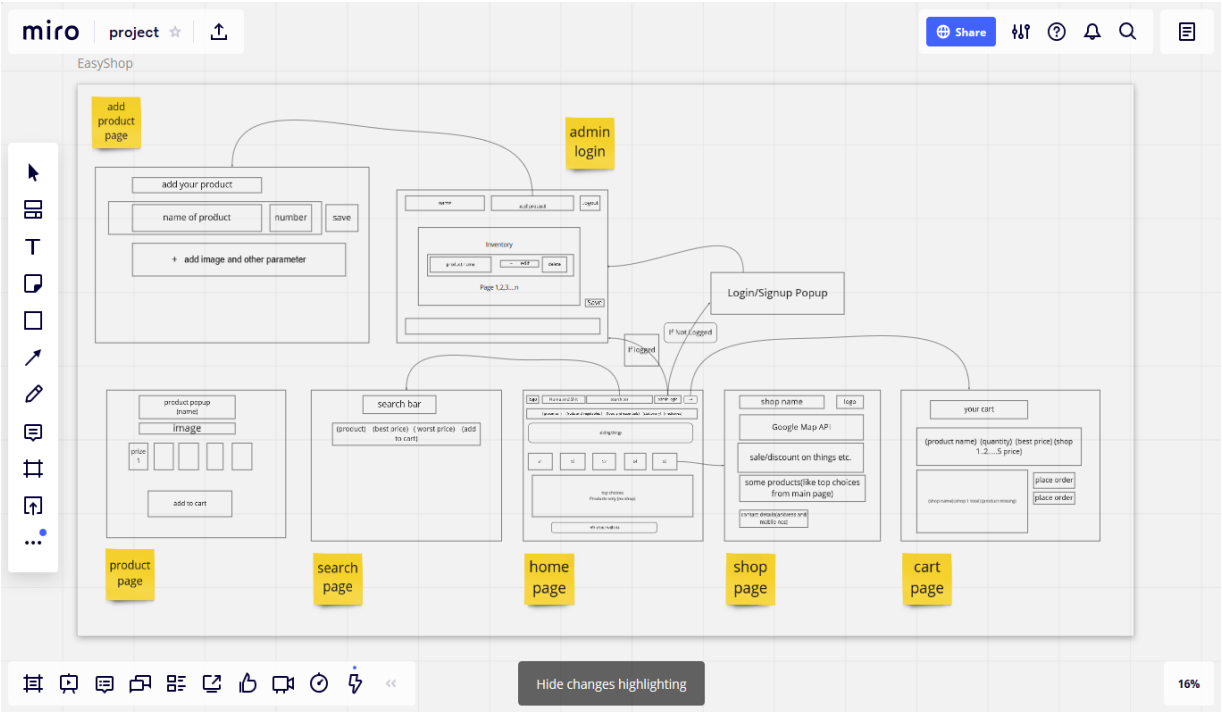


Fig. 3.9 Overall Website Flow

This is the overall flow of the website.

4. Implementation

Hardware requirements

To host the website, min. HW requirement is that you need to run minimal linux flavour.

As the product is a website it requires no special hardware interface except for the device the customer/admin is using to access the website.

Software requirements

We are using blakeJS,cors,express,node-rsa for modifying the transmission between client-server architecture.

We are using body-parser,cookie-parser,dotenv,jsonwebtoken,path for back-end development.

We are using reactJS, material-UI, axios, mdb icons, react-router-dom for managing front-end .

We will be using google maps embedded API (free one) to locate the shops natively on the website.

Code Implementation:

1. **Fast Modular Exponentiation:**

```
module.exports = (a, b, n) => {
  a = a % n;
  var result = 1;
  var x = a;
  while (b > 0) {
    var lsb = b % 2;
    b = Math.floor(b / 2);

    if (lsb === 1) {
      result = result * x;
      result = result % n;
    }

    x = x * x;
    x = x % n;
  }
  return result;
};
```

2. **PR Pass:**

```
const generatePass = (initPass, lucky) => {
  var luck = parseInt(lucky);
  luck1 = luck % 5;
  luck1 = `${luck1}`;
  var ans = "";
  for (let i = 0; i < initPass.length; i++) {
    a = initPass[i];
    if (/[a-zA-Z]/.test(a)) {
      ans += data[luck1][a];
    } else {
      ans += a;
    }
  }
  ans = wrap(ans);
  luck += 100;
  luck %= 72;
  ans1 = sample[luck].toString();
  for (let i = 1; i < 12; i++) {
    ans1 += sample[ans1[i - 1].charCodeAt(0) % 72];
  }

  for (let i = 0; i < ans.length; i++) {
    ans1 = add(ans1, ans[i]);
  }
  luck += initPass.length + ans.length * initPass.length;
  luck %= 72;
  char1 = sample[(luck % 26) + 26] + sample[((luck + 111) % 10) + 52];
  char3 = sample[((luck + 222) % 10) + 62] + sample[(luck + 300) % 26];

  luck = (initPass.length * luck) % ans1.length;

  ans1 = ans1.slice(0, luck) + char1 + ans1.slice(luck, ans1.length);

  luck += initPass.length + ans.length;
  luck %= ans1.length;

  ans1 = ans1.slice(0, luck) + char3 + ans1.slice(luck, ans1.length);
  return ans1;
};
```

3. **Diffie Hellman Generation:**

```
module.exports = (serverPub) => {
  const clientPri = Math.floor(Math.random() * 900000) + 100000;
  const clientPub = modularExp(17, clientPri, 541);

  const sharedKey = modularExp(serverPub, clientPri, 541);

  return {
    clientPri,
    clientPub,
    sharedKey,
  };
};
```

4. **Submitting Request:**

```
const submit2 = (e) => {
    e.preventDefault();
    const { name } = e.target.elements;
    console.log(JSON.parse(currentUser));
    console.log(name.value);
    const { clientPri, clientPub, sharedKey } = generatePriPub(
      JSON.parse(currentUser).serverPub
    );
    const timestamp = Date.now();
    console.log(clientPri, clientPub, sharedKey);

    const payload = generateCipher(
      {
        prod: name.value,
        operation: -1,
        timestamp,
      },
      sharedKey,
      JSON.parse(currentUser).token
    );

    const hmac = generateHmac(
      payload,
      generatePass(JSON.parse(currentUser).token, sharedKey)
    );
```

```
axios
  .post(
   server + '/store/update',
   {
    payload: `${payload}|${hmac}`,
   },
   {
    headers: {
     authorization: JSON.parse(currentUser).token,
     publickey: clientPub,
    },
   }
  )
  .then((res_) => {
   if (res_.data === 'success') {
    alert('updated');
    window.location.href = '/admin';
   }
  })
  .catch((err) => {
   console.log(err);
   alert('Error Occured');
  });
};
```

Website

This is the final website that was developed for this project it can be accessed using the given link

Link: http://easyshopindia.herokuapp.com/

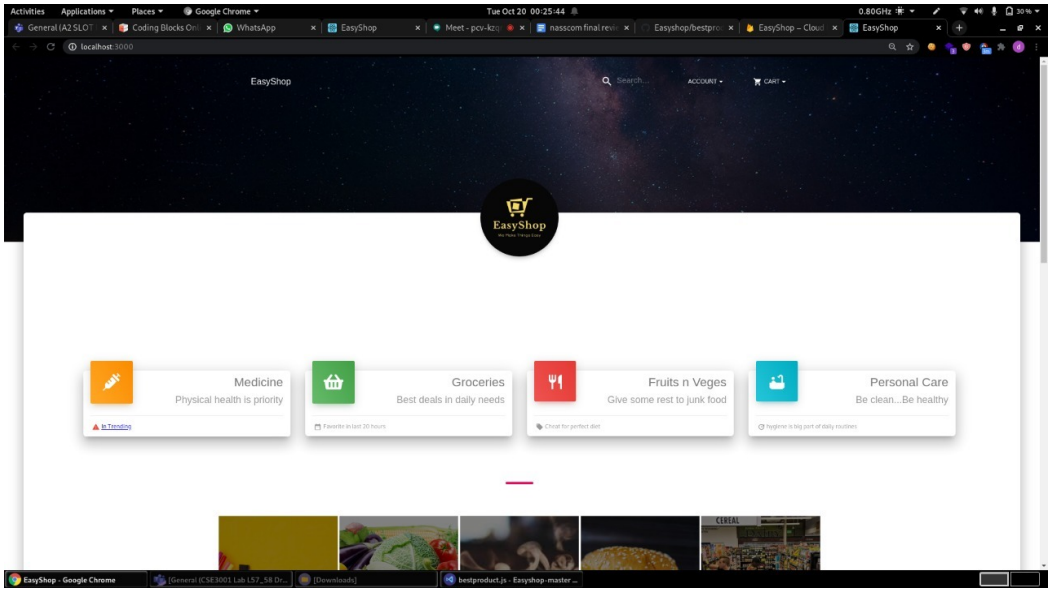


Fig. 5.1 Home page

This is the homepage of the website that we made

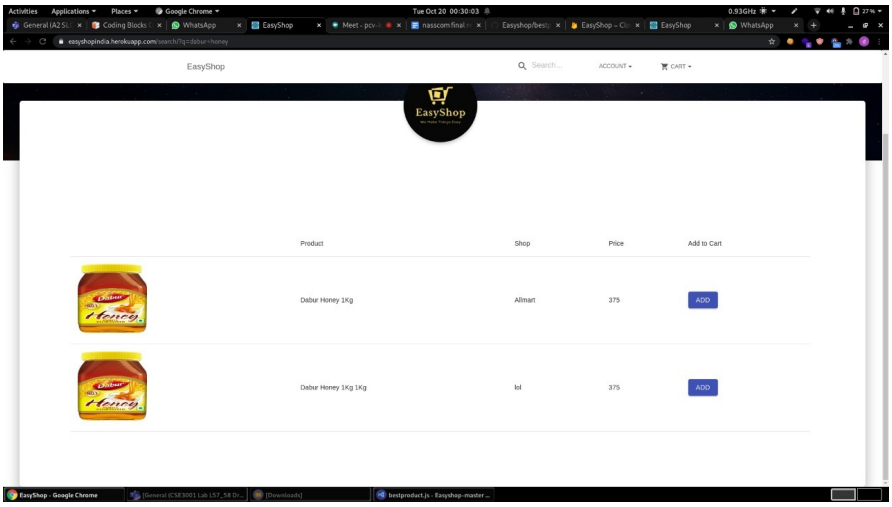


Fig. 5.2 Search page

This is the search page. Upon searching for the product, this page will show up with the matched products.

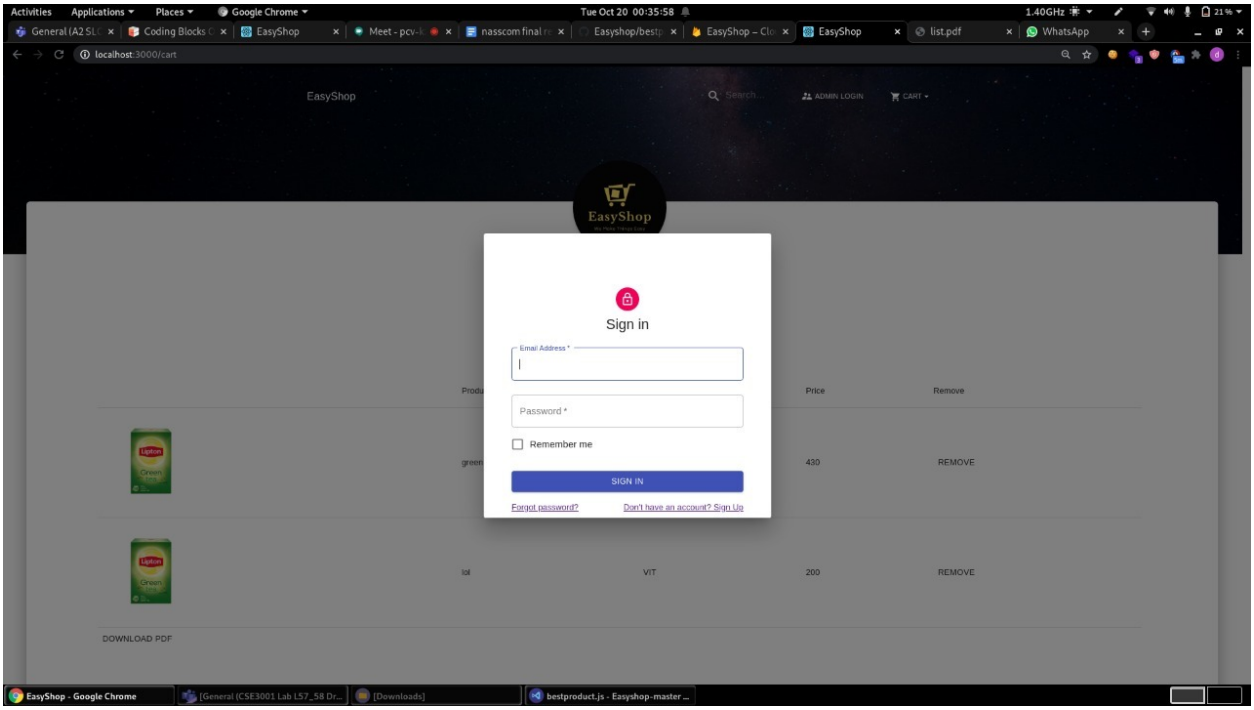


Fig. 5.3 Admin Login

The login page where shopkeepers can login to the website to manage their inventory.

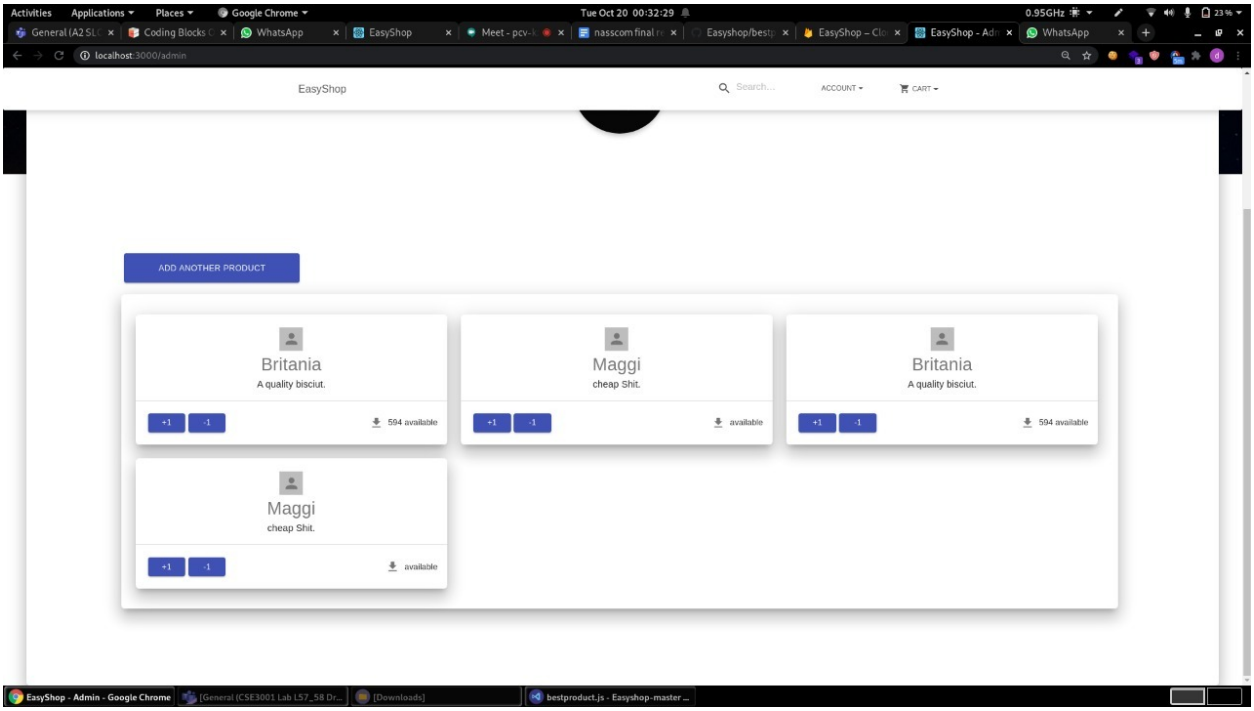


Fig. 5.4 Admin Inventory panel

After the shopkeeper logs in, he will see this page. He can update his inventory with this page.

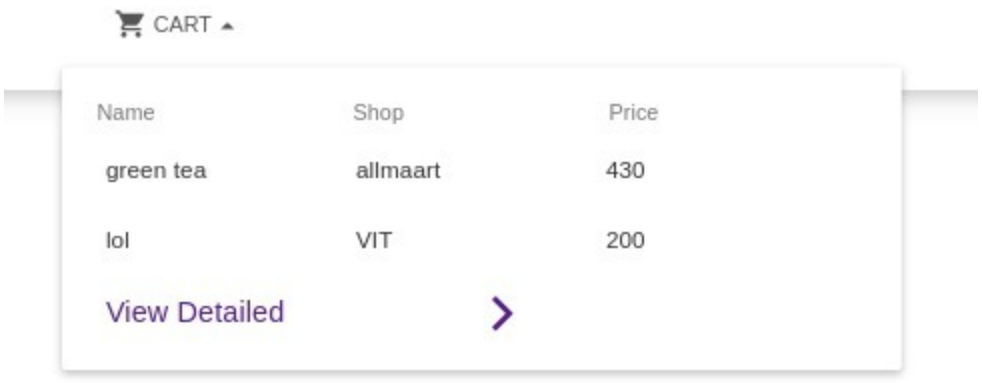


Fig. 5.5 cart

This is a quick cart, the user can quickly checkout his cart here instead of having to go to a different page to see the cart.

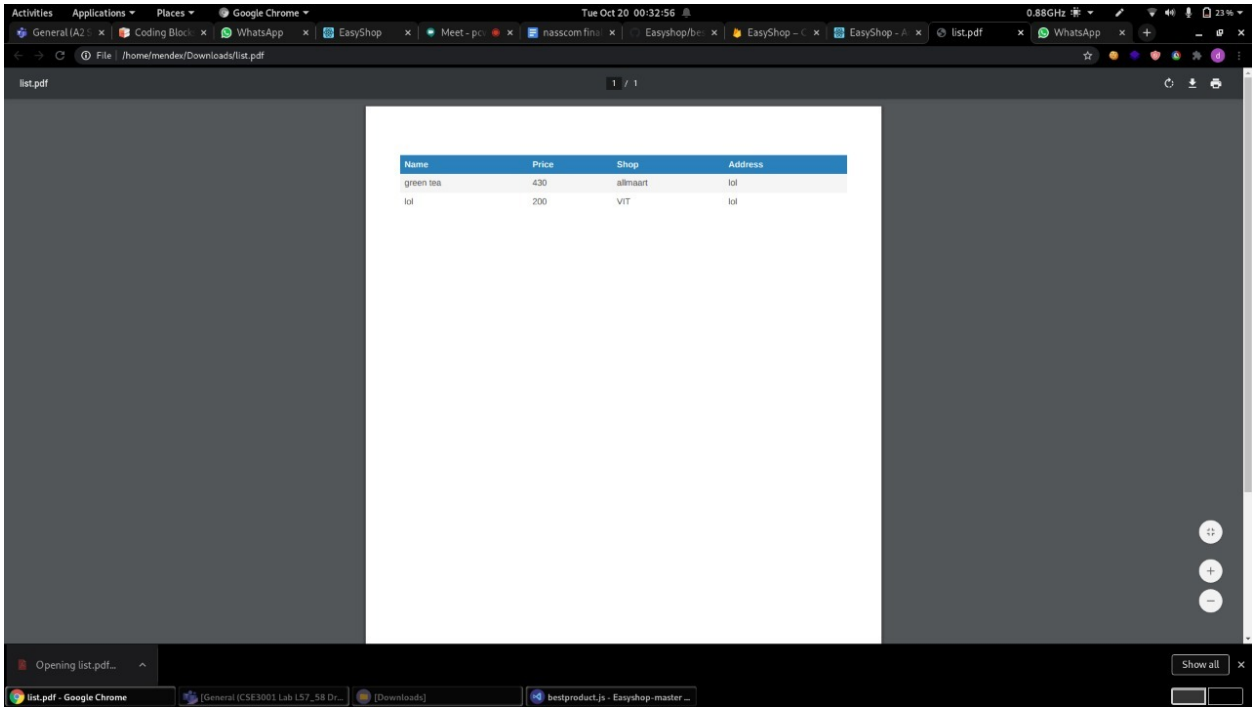


Fig. 5.6 Checkout

After the user "checks out", a pdf will be downloaded giving the approx amount needed for the purchase and other things.

5. Vulnerability analysis

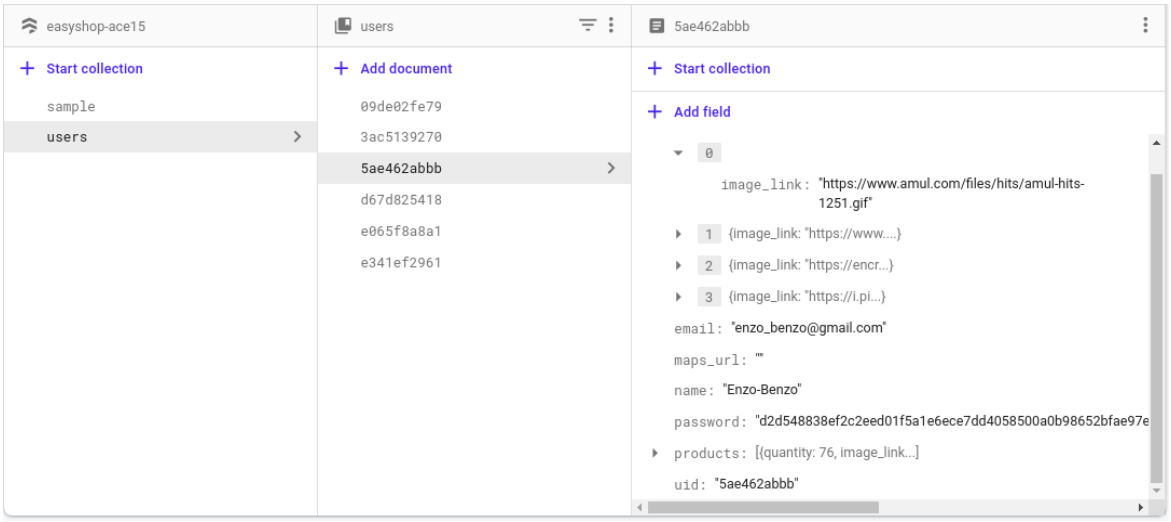


Fig. 5.1 Hashed Password Stored in Database

We store our users' data in hashed format. So if in any case there is database breach, hacker can't see the details of our user

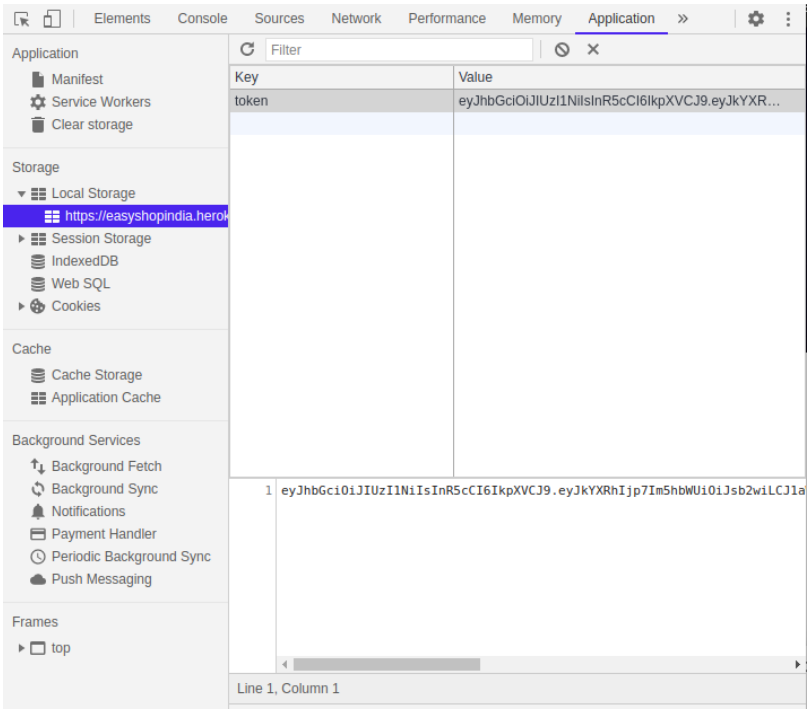


Fig. 5.2 Encrypted JSON web Token

We also take care of the request that our website makes. Our website will all request in just one line of "payload". So it will be very secure in case the hacker tries to use a burp suite then the hacker can see the only "payload" that is our request to the server.

- Google Lighthouse

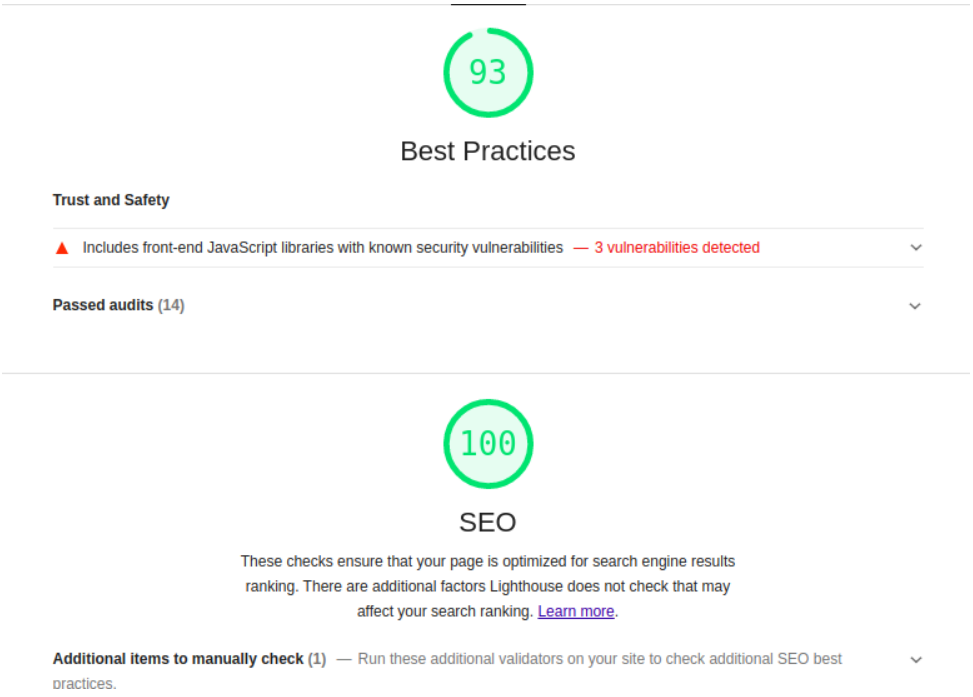


Fig. 5.3 Google Lighthouse testing

Here you can see we have achieved 100 in Search Engine Optimization without any paid plugin like the other websites and we have also achieved 93 in best practice which denotes that our code is well optimized for the compilation.

- Burp Suite Intercept testing

In burpsuite we use fuzzer to acknowledge all the vulnerabilities in our website. Via fuzzer we can see the all post request that will be made by the website to the database or server. So we intercept all the traffic for our particular website and we can see all the requests have been made to the database server via our website. And we can clearly see our website made a login post request to the database server and it shows “payload” instead of login-id and password information. And you can’t see the exact data that the user has entered, instead the raw data penetration tester only sees a hash of the request. So you can’t see the exact data nor you can see what type of data request has been sent. It will send all types of requests in just one line of “payload”.

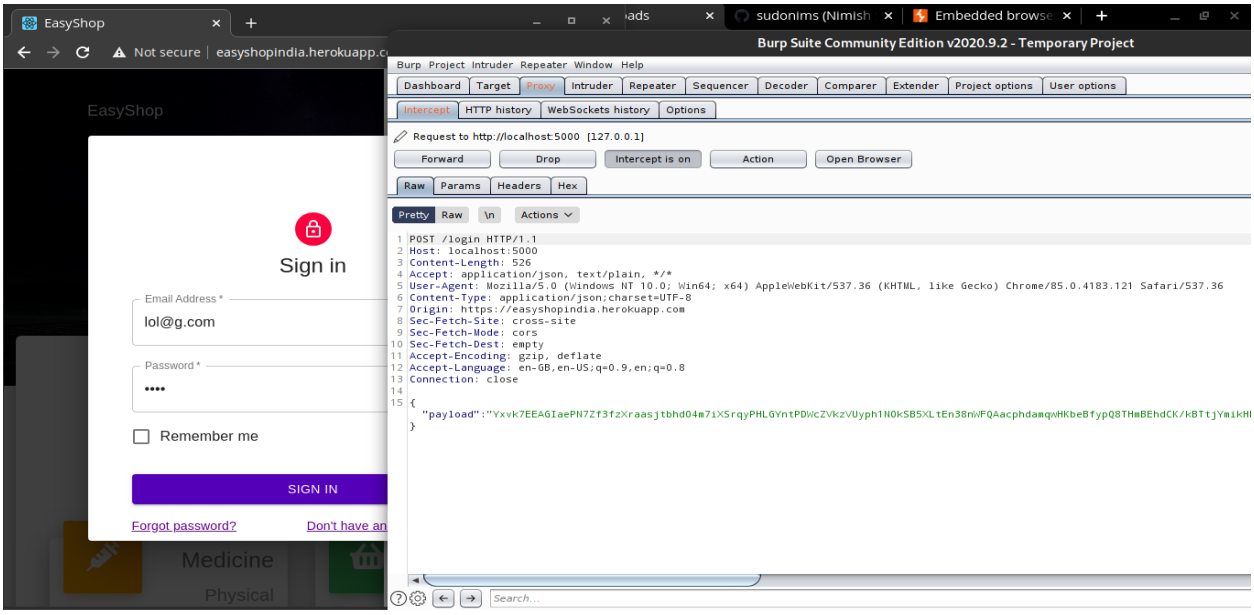


Fig. 5.4 Burp Suite intercept

- Nessus testing
  Nessus is a remote security scanning tool, which scans a computer and raises an alert if it discovers any vulnerabilities that malicious hackers could use to gain access to any computer connected to the network. We tested our website on Nessus Tool and got 2 medium level vulnerabilities and 27 negligible vulnerabilities. Both the Medium level vulnerabilities are not for our website, but are present on the platform we have used to host our website - Herokuapp

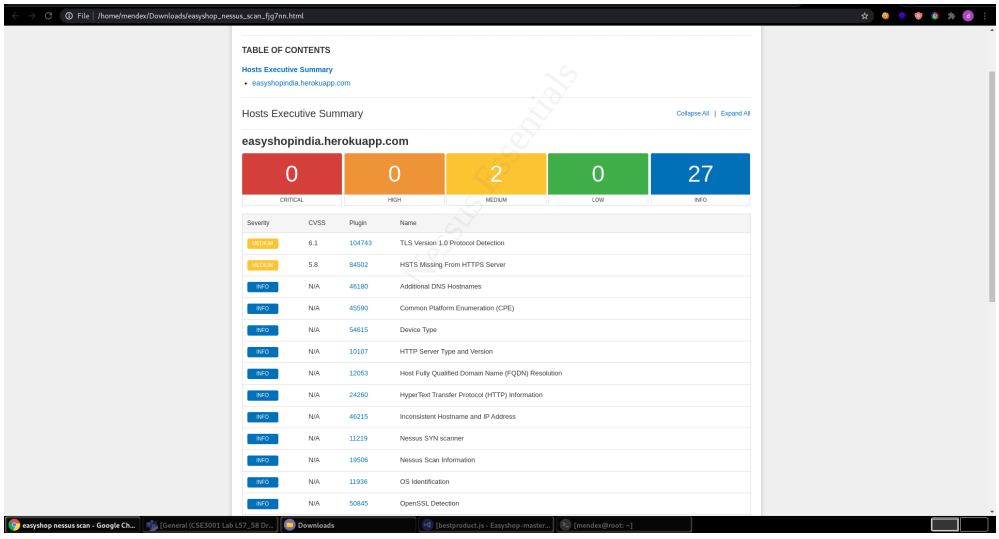


Fig. 5.5 Nessus test

This image lists the vulnerabilities found in our website by Nessus

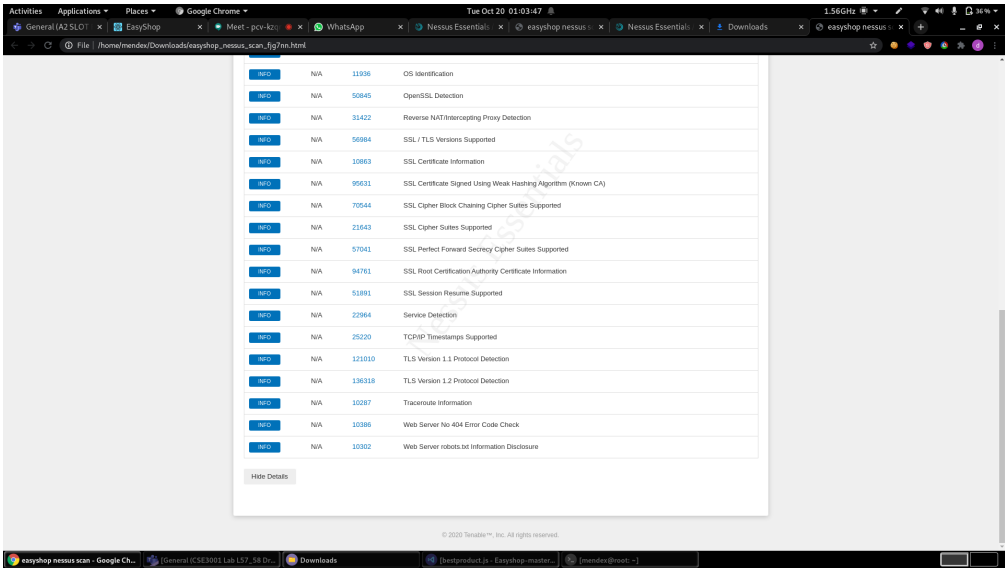


Fig. 5.6 Vulnerabilities found by the Nessus test

6. Preventive measures

Some of the security measures taken are

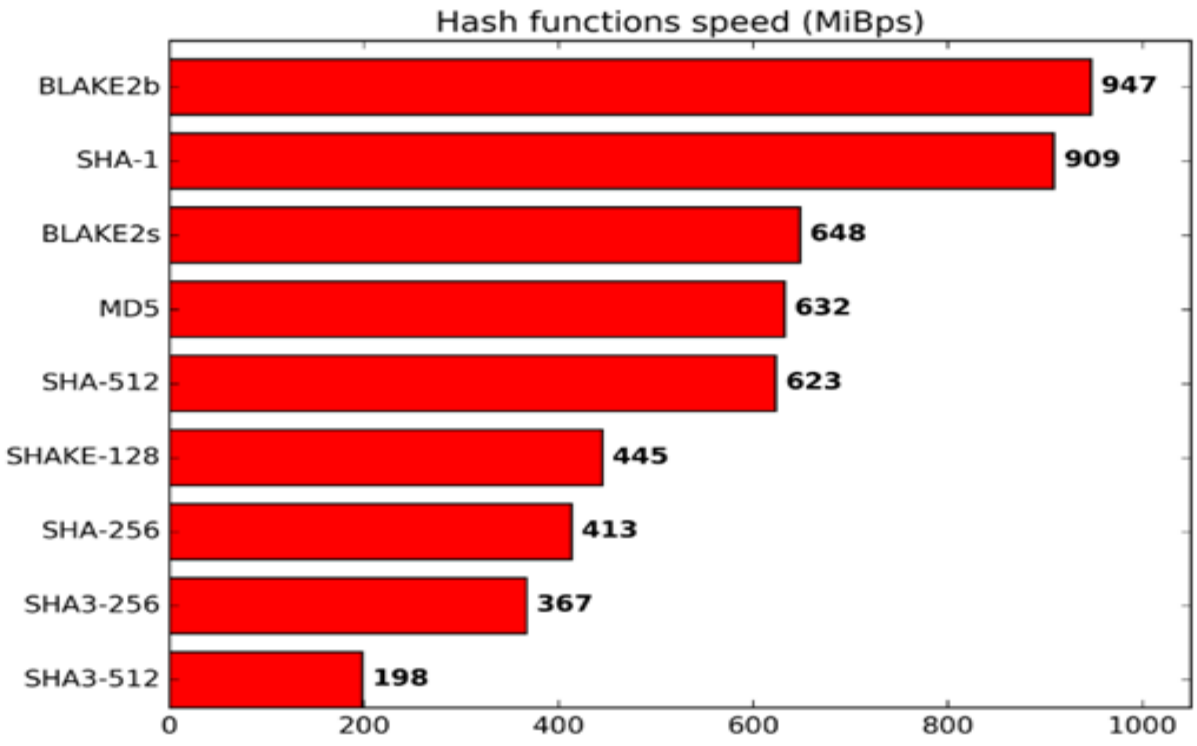
- The passwords for the users are stored in an hashed manner before being sent to the firebase database which then encrypts it before storing.

- All the requests made are first encrypted using AES algorithm and then sent so then any attacker cannot access the data in transit, and even if they intercept it will be useless for them as they will not be able to decipher it.

- Also we generate new key pairs for every request making it difficult for any attacker to try to break keys as breaking so many keys in such a short period of time is not possible.

- We use new hashing algorithms like blake2b instead of SHA as they are faster and more secure.

- We updated our hashing algorithm from sha256 to a faster and secure blake2b hashing algorithm. While both have the same algorithmic time complexity, runtime complexity of blake2b is far better than sha256. You can see that in picture below,



Src : https://blake2.net/

- We also use RSA 3072 bit encryption which is better than standard encryption algorithms.

- The public and private keys used in the RSA algorithm are created with “ssh-keygen” which is industry standard for cloud computing. Sowe can ensure the robustness of private keys that will be used in our application.

- The site hosted is also on HTTPS which is more secure than HTTP.HTTPS is not stateless protocol, so it can be used to store user sessions across the complete web application.

- The site is immune to most common attacks like Cross-Site Scripting (XSS), SQL injection, path traversal, local file injection and distributed attacks like DOS and DDOS.

7. Conclusion

This project has successfully made a web app for the given problem statement. And has met most of the requirements stated at the beginning of the project.

The language used is HTML, CSS, and JS and the database used is Google Firebase. It is a web application useful for any person looking to buy groceries. The application is flexible, easy to use with interactive UI.

So the e-commerce website we have developed is immune to most common web attacks namely: Cross-Site Scripting (XSS), SQL injection, path traversal, local file injection and distributed attacks like DOS and DDOS. and we have also successfully tested it using many tools like Google Lighthouse, Burp-suite and Nessus.

8. References and literature survey

[1]

E-Commerce refers to the exchange of goods and services over the Internet. The shopping through e-commerce has penetrated all segments of goods ranging from groceries to electronic goods and even vehicles. Rapid growth in mobile computing and communication technologies has facilitated the popularity of e-commerce. The main impediment in growth of e-commerce is cyber fraud and identity theft. Hackers are people who carry out cybercrime. Hence, poor security on e-Commerce web servers and in users computers is a core issue to be resolved for rapid growth of e-commerce. This paper provides directions for e-commerce security so as to improve customer confidence in e-commerce shopping.

[2]

With the rise of the Global Economy, and with an ever-expanding level of purchasers doing their business fundamentally by means of on the web or cell phones, electronic trade, internet business, is quickly being viewed as the best approach worldwide at the bit of a catch. Subsequently, building up a successful Web based business model is getting essential for any cutting edge business. In any case, an organization must address diverse new security challenges and be sure to keep up the best expectations of web based business security, to ensure both themselves and their clients. An inability to hold fast to severe internet business security can bring about lost information, traded off exchange data, as well as the arrival of the client's monetary information. This can lead to lawful and budgetary risk, just as a negative effect on the organization's notoriety. These new security challenges are the consequences of the utilization of the new innovation and correspondence medium, and the progression of data from big business

to undertaking, from big business to purchasers, and furthermore inside the undertaking. This paper presents the distinctive innovation and calculated parts of the web based business as a rule, and recognizes and orders the various sorts of security challenges confronting online business organizations specifically.

[3]

This paper deals with various security issues faced by ecommerce websites that are hosted on public networks. The core issue discussed is security of E-commerce transactions and two aspects of it, which are the security of the system and the security of information. The two major issues discussed are Computer network problems and The security issues of business transaction. There are also some proposed security control requirements at the process of E-commerce transaction. They are: The validity of the information, The confidentiality of information transmission, The integrity of the transaction information, The integrity requests when storing the data should prevent illegal destruction or change on the site, Non-repudiation of information, The authenticity of the traders identity. The two traders do indeed exist, not fake, The information can not be amended. The message on the network can not be modified.

[4]

This paper deals with privacy and security issues faced by modern ecommerce websites and how these issues faced by the customers are restricting them from engaging more with these websites. It also states that Web e-commerce applications that handle payments have more compliance in issues, are at increased risk from being targeted than other websites and there are greater consequences if there is data loss or alteration. The paper then also goes on to discuss web security in general and how rapidly growing e-commerce is a challenge online security as a whole.

[5]

Encrypting and decrypting data: Encryption works at the byte level, so almost anything can be encrypted. Once you have a key and a cipher, you’re ready to go. It should be noted that the same algorithm must be used for both the key and cipher. You cannot have a key initialized with DES and a cipher initialized with

RSA. The Cipher object uses the same methods to encrypt and decrypt data, so you must initialize it first to let it know what you want done with the data:

rsaCipher.init(Cipher.ENCRYPT_MODE, publicKey); //Initializes the Cipher object.

This call initializes the Cipher object and gets it ready to encrypt data. The simplest way to encrypt data is invoking the doFinal method on the Cipher object passing in a byte array:

byte[] data = Hello World! .getBytes();//Calculates the ciphertext with a plaintext string.

byte[] result = cipher.doFinal(data);

The result will now contain the encrypted representation of the passed-in data. It's just as easy to decrypt the same data. But before we can do that, we must reinitialize the Cipher object and get it ready for decryption

[6]

This paper aims at speeding up RSA decryption and signature. The performance of RSA decryption and signature has a direct relationship with the efficiency of modular exponentiation implementation. RSA 3072 is equal to 128 bit symmetric and 2048 is equal to 112 bit symmetric then 128-112= 16 and 2^16= 65,536. This paper proposes a variant of RSA crypto systems by reducing modules and private exponents in modular exponentiation. The experimental result shows that the speed of the decryption and signature has been substantially improved and the variant can be efficiently implemented in parallel.

[1] *Cyber Security; Issue and Challenges in E-Com*

[2] *E-Commerce Security Challenges: A Taxonomy*

[3] *Y. Wen, C. Zhou, J. Ma and K. Liu, "Research on E-Commerce Security Issues," 2008 International Seminar on Business and Information Management, Wuhan, 2008, pp. 186-189, doi: 10.1109/ISBIM.2008.168.*

[4] *The Study of E-Commerce Security Issues and Solutions*

Sangeetha M K Prof. Dr. Suchitra R Jain University, Bangalore Jain University,Bangalore

[5] *Kefa Rabah , 2006. Implementing Secure RSA Cryptosystems Using Your Own Cryptographic JCE Provider. Journal of Applied Sciences, 6: 482-510.*

[6] *Design and implementation of an improved RSA algorithm, Yunfei Li ; Qing Liu ; Tong Li,2010 International Conference on E-Health Networking Digital Ecosystems and Technologies (EDT),10.1109/EDT.2010.5496553*