

CHAPTER 1

INTRODUCTION

Today's world is developing at a rapid speed. Everyday a new Application is developed. At the same time, there are so many security issues are being faced by people in the Important Applications that we use in the day to day life. There are so many Security issues that are been discussed , but one major Security issue is Cross-Site Request Forgery, which comes under the top 10 Security Issues.

1.1 OVERVIEW

India is the 58 most competitive nation in the world out of 140 countries ranked in the 2018 edition of the Global Competitiveness Report published by the World Economic Forum as referred in [28]. Most of the Indian people use Net-Banking as the part of their daily life. In the recent times, there are so many banks that come up with their net banking sites. It confuses the common people whether to believe it or not, as it comes under the Money Matters.

Cross-Site Request Forgery is a Security Attack that is being done on the Net-banking Sites. Now a days, it is a very common attack made by the hackers not only in the net banking sites, but also in the developing E-Commerce Sites. CSRF is an attack that forces an end user to execute unwanted actions on a web application in which they are currently authenticated.

CSRF attacks specifically target state-changing requests, not theft of data, since the attacker has no way to see the response to the forged

request. With a little help of social engineering, an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.

CSRF vulnerabilities have been known and in some cases exploited since 2001. Because it is carried out from the user's IP address, some website logs might not have evidence of CSRF. Exploits are under-reported, at least publicly, and as of 2007 there were few well-documented examples for the CSRF attacks.

The Netflix website in 2006 had numerous vulnerabilities to CSRF, which is referred in [27], could have allowed an attacker to perform actions such as adding a DVD to the victim's rental queue, changing the shipping address on the account, or altering the victim's login credentials to fully compromise the account.

The Online Banking web application of ING Direct was vulnerable to a CSRF attack that allowed illicit money transfers. Popular video website YouTube was also vulnerable to CSRF in 2008 and this allowed any attacker to perform nearly all actions of any user. McAfee was also vulnerable to CSRF and it allowed attackers to change their company system.

New attacks against web-enabled devices were carried out in 2018, including attempts to change the DNS settings of routers. Some router manufacturers hurriedly released firmware updates to improve protection, and

advised users to change router settings to reduce the risk. Details were not released, citing "obvious security reasons".

There are so many steps taken by the Government in order to come out of the CSRF Attacks, but those were able to control only some of the Major Attacks. Still, there are attacks happening in the world that mandatorily needs a proper solution to the End Users, to use the E-Commerce websites and the Net-banking websites with Trust and without Fear.

1.2 PROBLEM STATEMENT

The Applications contribute to the Major part of Human Lifestyle in Today's technology oriented life in Digital India. We can't see anybody without a mobile that has an Internet Connection to access Mobile Applications.

The Major problem about the existing E-Commerce Applications is the Security issues to the End Users. For an instance, the Net Banking Applications may have clients all over the world and the clients may have to access their account in different Computers, Laptops, Mobile phones or other devices.

The Clients may have meetings and dealing of their business all over the world and hence the dealing of the currencies matters a lot for them. They can rely on the Net Banking Web applications with more security. In this case, the clients are sometimes forced to do some operations or even sometimes at the worst cases the Client's Login Credentials are forced to reveal by the Client, without the Clients' knowledge.

Attackers who are mainly intended to acquire the Client's data make a web application that is typical to the Net Banking Application and are ready to enter into the Client's Account without the Client's knowledge. Once the Client refreshes the page or has some Networking Issue, he has to login from the first and that is the time for the Attacker to introduce his Forged web application as the Original Net Banking application.

The End users does not know the difference between the Original Web Application and the Forged Web Application and enter the details to get the login once again. The Hacker gets the Login Information and acquire the full access to the Client's account without their knowledge and can do anything in the acquired account prior to the client.

Attacks are also done based on the saved cookies to the frequently accessed web applications, Unauthorized links that reach the users through the spam mails and unfortunately the user is happened to visit those links and unable to come out of it or the user's personal data is completely accessed by the attackers behind the Unauthorized link who are intended to stole the users data.

In the recent years, the Trust is a main factor to the Web Applications and due to the Security issues the users have a major hesitation to use the available technological developments. In 2006, Bangladesh had a top issue in one of its major Web Applications and because of the Cross-Site Request Forgery, it has lost about \$100 million. After this issue, the threat has become serious and people are in need to get a proper solution to get rid of the various attacks made by Cross-Site Request Forgery.

1.3 EXISTING SYSTEM

1.3.1 SECURED CONNECTION

In order to overcome the Security issues based on the Cross-Site Request Forgery, initially the http pattern is changed into https pattern, in which 's' denotes the Secured Connection. For the applications that uses the transfer of money or any other data that is confidential, the url contains https connection and now it is being implemented for all the common connections also.

Based on [3], the https secured connection contains a Lock Symbol on the url, which has the settings or the pre-requisites that are to be done in order to avoid the Third-party entry in confidential web applications. The symbol has the onClick() function which disables all the pop-ups that arises in between the transactions and it asks the user to allow or block the required features of the browser.

1.3.2 HTTPS POST METHOD

Data security is achieved by the secured methods that are used to pass the data. Initially, the GET() method is used to pas the data in the url, but this method is proved to be unsafe due to the attacks made in the url, that revealed the passed data, the POST() method is used in the Confidential data transfers, like passing the password as the argument to validate a login page which is referred in [1].

1.3.3 SAME-SITE COOKIES

Whenever a user is very frequent to access a website, the habit of saving the username and password as a cookie in the browser is done. This creates a problem with regard to the attack made by CSRF and hence the

same-site cookies are removed and the users are asked to remember their login credentials whenever they are suppose to login which is referred in [15]. By this, the login credentials of the user are safe-guarded.

1.3.4 TOKEN AUTHENTICATION AND CAPTCHA

Attackers continue to make the CSRF attack through all the loop holes available and hence to authenticate the user, a random number is generated as a token by the server and the token is validated for the user, to confirm the user attempt to make the login. The Token authentication stopped most of the tries to make the CSRF attack and later the end users felt that the authentication is not enough to identify the user and the in between attacks made by the computer. Hence, the Captcha scheme is introduced so that the user can see the captcha and type it, but the system fails.

1.3.5 AVOIDING TRANSITION BETWEEN WEB PAGES

CSRF attacks become very intense and it become to evolve by forcing the users to visit another web page, that the user have not visited before. That forged website may contain and piece of code to collect the user's personal data. Hence, based on [19] the transition to another web page from the existing web page is instructed to be avoided and no other tabs apart from the existing tab should be active.

1.3.6 TIME-OUT AND RE-AUTHENTICATION

In order to make the Token Authentication more stronger, the Time-Out concept came into existence, in which the token is set a time from the time of generation from the server and the time-out happens when the user takes more time than the threshold time, that is being set. Based on [25],

the concept enhances the random attacks by the attacker for the long time and getting the random number correct at one point of time.

1.4 PROPOSED SYSTEM

The existing attempts made in order to get rid of Cross-Site Request Forgery attacks were not capable to resist the Red-Hat hackers and the Shadow hackers. They constantly tried to make the attack from the existing drawbacks that can allow the intermediate user to come and join the existing user, without their knowledge. Hence, this issue is considered serious and the following proposal is made to overcome the existing attacks.

The Cross-Site Request Forgery attack mainly focuses on the user's diversion to indulge into the website. So, the proposal gives more importance to identify the user during the login session and also confirms that the same user makes the transaction. The existing approach contains the generation of a random number as a token for authentication and confirms token in the hidden form fields. In order to confirm the user, the proposal takes the MAC Address of the system during the login session in the hidden form field and uses the Mathematical BCX Hash Algorithm to generate the hash value for the MAC Address.

1.4.1 LOGIN VALIDATION

Whenever the user logs in, the username and the password of the user is verified by the existing Login Information, that is available in the corresponding Server. After the validation of the login information the BCX Algorithm uses a new table in the database to record the login activity of the user, for every new login that has been done. The new table makes an entry of the username, password, the hash value produced by the BCX hash Algorithm of the MAC Address. The server later generates an Anti-Forgery

Token by the same BCX Hash Algorithm by combining the current IP address taken from the hidden form field and the password of the user. The generated Anti-Forgery token is saved in the newly created table and the Anti-Forgery Token is being carried out to the Transaction page when the user makes the transaction. After all these procedures, the user enters into the Home Page.

1.4.2 BCX HASH ALGORITHM FOR MAC ADDRESS

The BCX Algorithm expands to Binary Conversion, Combination and X-OR Operation. To generate the hash value for the MAC Address, the algorithm gets the input as the MAC Address in the hidden form field and add each number in the Mac address and have it as Sum_Of_Mac. The Random Threshold value is set between 0 and the Sum_Of_Mac. Two values A and B are set such that A equals the Sum_of_Mac divided by the Random Threshold and B value equals to the Sum_Of_Mac modulo Random Threshold. If the value of A is greater, then compute AC_B else compute the value of BC_A . Initialize an Integer array of size 6, and put on the ASCII values of the MAC Address. The ASCII values are X-ORed with the Combination values and store the resultant values in the new array. The new array's values are subjected to Binary conversion and after each string the next binary value is appended. The number of 1^s and 0^s are counted and concatenated as a Integer and named as Pre-hash value. The hash value of the MAC Address is obtained by computing the value of ${}^{Pre-Hash_value}C_{Sum_Of_Mac}$. This value is stored as the hash vale of the MAC Address in the database in the form of Big Integer. The BCX Algorithm for MAC Address is computed at the time of login and during the transaction to verify the origin.

1.4.3 BCX HASH ALGORITHM FOR ANTI-FORGERY TOKEN GENERATION

The Anti-Forgery Token is produced by combining the password of the user and the current IP Address of the Client's System. The password is made up of Letters and Numbers. The Non-Numeric terms in the password are identified and are converted into Numeric terms by their corresponding ASCII value and now all are Integers. Now, the non-binary integers are converted into Binary values to obtain a big Binary value and stored into the string called 'pwd_ip_hash'.

The IP Address is subjected to the String Tokenizer to remove the '.' and the non-binary terms are converted to Binary terms. This binary term is appended to the pwd_ip_hash. The string in the pwd_ip_hash is subjected to a split of 3^s and then they are converted into Decimal values and passed into the BCX hash algorithm to get the Anti-Forgery Token.

1.4.4 TRANSACTION VALIDATION

The User enters into the Home Page after login validation, where the user has a lot of options about the Customization, Personalization of the Account and Transactions that can be made. Whenever the user wants to make the transaction, the user clicks on the Transaction Icon.

In the transaction page, the Beneficiary and the Amount to be transferred is fixed and when the user clicks 'OK', the MAC address is again taken from that page and given to the BCX algorithm. The hash value of the MAC Address is prepared. Then, the Anti-Forgery Token is passed to the server and it is verified.

If the AFT is matched, then the hash value of the MAC Address is compared with the hash value of the MAC Address when the user logs in, in order to double confirm that the same user who login, is now making the transaction in the same session. If both the AFT and the hash value of the MAC Address is matched, the transaction is proceeded, otherwise the session is made time-out and logged out.

CHAPTER 2

LITERATURE REVIEW

Yameng Haung et. al., (2018) proposed Binary Multidimensional Scaling for Hashing, which demonstrates Unsupervised hashing, which aims at learning binary hash codes for the original features so that the pair wise distances can be best preserved. Based on [13], While several works have targeted on this task, the results are not satisfactory mainly due to the over-simplified model. In this paper, a unified and concise unsupervised hashing framework, called binary multidimensional scaling is used which is able to learn the hash code for distance preservation in Batch mode and Online mode.

G.Parimala et. al., (2018) proposed Efficient Web Vulnerability Detection Tool for Sleeping Giant-Cross Site Request Forgery, which implements a real time scan of CSRF vulnerability attack in given URL of the web applications as well as local host address for any organization using python language. Based on [20], where the Client side detection of CSRF is depended on Form count which is presented in the given web site.

Akash Agarwall et. al., (2017) proposed Modeling and Mitigation of Cross-Origin Request Attacks on Federated Identity Management Using Cross Origin Request Policy, which depicts the construction of formally checkable models and design laboratory simulations to show that Federated Identity Management (FIM) is susceptible to cross origin attacks. Based on [22], Further, the proposal employs the Cross Origin Request Policy (CORP) to mitigate such attacks.

Prathiba Yadav et. al., (2017) proposed a Report on CSRF Security Challenges and Prevention Techniques which include Sea Surfing, Session Riding, One-Click attack, XSRF, Application Vulnerability, Cross Origin Request, Security Tokens, Origin Header, malicious URL and the Referrer Header. This system proposes the defense of the Web Application by One-Time Pass Code, Re-authentication and the CAPTCHA techniques. Based on [3], It also enforces the user not to use parallel tabs to access websites and surfing when the Banking sites credential opened in another TAB, not to save the Username and Password, do not allow the sites to remember the Login Credentials.

Fatma Kahri et. al., (2017) proposed An Efficient Fault Detection Scheme for the Secured Hashing Algorithm SHA-512, which interprets a number of counter measures based on the Hybrid Redundancy to be an efficient Fault Detection Scheme. The proposed Detection Scheme has been implemented on Xilinx Virtex-II Pro FGPA. It is fault coverage, area degradation, frequency, throughput and efficiency overhead have been compared and based on [8], it is shown that the proposed scheme allows a trade-off between the security and the implementation cost of the SHA implementation.

Tanjila Farah et. al., (2016) proposed Assessment of vulnerabilities of web applications of Bangladesh in which \$100 million has been stolen from Bangladesh's account. Cross Site Scripting and CSRF are involved in this attack. Because of the severity of these vulnerabilities, the users trust is violated. This proposal [5] discusses about XSS, Stored XSS, Reflected XSS and DOM based XSS with respect to the CSRF Attacks. The injection of the CSRF forgery web page is done by the DOM and Reflected XSS and

the user is forced to visit the Forged web application and reveal his Login Credentials Unknowingly to make a Transaction.

D.Kavitha et. al., (2016) proposed Prevention of CSRF and XSS Security Attacks in Web Based Applications, which generates the Anti-Forgery Token and passes it to the MD5 Algorithm. Thus based on [19], the hash value will be considered as the final Anti-Forgery Token, thereby enhancing the security on the Client Side.

Madhumitha Panda et. al., (2016) proposed Performance Analysis of Encryption Algorithms for Security which emphasis on encryption and decryption of different types of files like binary, text and image files by the Cryptographic Algorithms like AES, DES and Blow Fish and the Asymmetric algorithms like RSA Algorithm. Based on [7], A comparison has been conducted for these Algorithms using the Evaluation parameters such as Encryption time, Decryption time and Throughput.

Abhishek Bhardwaj et. al., (2016) proposed Study of Different Cryptographic Techniques and Challenges in Future, which emphasis on all the Data Hiding Techniques from the World Wars and the evolution of all the Cryptographic Algorithms from Caesar Cipher to DES and then to 3DES, AES and all the Current Algorithms. The proposal [11] explains all the Pro's and Con's of the Algorithms and the reason for which the shifting of the Existing Algorithm to the New Algorithms began.

Wasim Akram Shaik et. al., (2015) proposed Avoiding Cross Site Request Forgery (CSRF) Attack Using Two Fish Security Approach, where the Two Fish security is an enhanced way to validate the web page and performs authentication in two phases; Firstly MD5 encryption is performed

in order to calculate the hash values for url and secondly image based authentication is provided to validate the image of respective url. By using this strategy, hence based on [23], the user can easily recognize whether a website is a genuine website or vulnerable website.

Purnima Khurana et. al., (2014) proposed Vulnerabilities and Defensive Mechanism of CSRF, which proposed the flaws in web applications that lead to the CSRF attacks such as Cookies, Browser Authentication and Client Side Certificates to Authenticate Users. Based on [2], The system explains the various CSRF attacks on ING Direct, YouTube, Meta Filter and The New York Times. The defenses are provided by the Tokens which is a Randomized String communicated by the POST Method, OWASP CSRF Guard to build protection into the application by mapping into a file and simply updating it. The Stateless CSRF Protection is used to maintain the User Token in the Server-side Session State, which creates a cookie with the CSRF Token and include the token in the Request Parameter.

Dudhatra Nilesh et. al., (2014) proposed a New Cryptography Algorithm with high Throughput which includes the Arithmetic and Logical Mathematical operations performed based on the existing cryptographic algorithms like 3DES, AES, DES and Blow Fish. The complexities of these algorithms are being discussed in [6] and the New Algorithm is converting the plain text of 128-bit into the binary stream of two parts of 64-bits, it is reversed and merged. Later the left and right four bits are X-ORed and divided into 2-bits and 4-bits respectively.

Bhawna Mewara et. al., (2014) proposed Browsers Defense against Reflected Cross-Site Scripting Attacks, which evaluates Three Browsers –

Internet Explorer 11, Google Chrome 32, and Mozilla Firefox 27 for reflected XSS attack against different type of vulnerabilities. The paper [14] concludes that none of the above is completely able to defend all possible types of Reflected XSS vulnerabilities. Hence, this paper emphasis on add-on integrated defense on the client side, instead of adding it as an Extension.

Marco Rocchetto et. al., (2014) proposed Model-Based Detection of CSRF, which describes how a web application should be specified in order to facilitate the exposition of CSRF-related vulnerabilities. The proposal [18] use an intruder model, a la Dolev-Yao and discuss how CSRF attacks may result from the interactions between the intruder and the cryptographic protocols underlying the web application, to demonstrate the effectiveness and usability of the proposed technique with three real-world case studies.

Anak Agung Putri Ratna et. al., (2013) proposed Analysis and Comparison of MD5 and SHA-1 Algorithm Implementation in Simple-O Authentication based Security System, which is an Automated Easy Grading System , uses MD5 + Salt Algorithm to perform protection for authentication password of users in the Database. The experiments include time measurements and estimation of brute force attack for each algorithm. Based on [9], processing time and CPU usage were also measured. In the brute force hash code scenario, it was tried to find plaintext from the cipher text. In this scenario, both MD5 and SHA-1 was implemented and tested using Hashcat tool.

Vikas K. Malviya et. al., (2013) proposed On Security Issues in Web Applications through Cross Site Scripting (XSS), which aims to study and

consolidate the understanding of XSS and their origin, manifestation, kinds of dangers and mitigation efforts for XSS. Based on [21], Different approaches proposed by researchers are presented and an analysis of these approaches is performed.

Yin-Chang Sung et. al., (2013) proposed Light-Weight CSRF Protection by Labeling User-Created Contents, in which paper presents a light-weight CSRF prevention method by introducing a quarantine system to inspect suspicious scripts on the server-side. Instead of using script filtering and rewriting approach, this scheme is based on a new labeling mechanism which enables the web server to distinguish the malicious requests from the harmless requests without the need to modify the user created contents. Based on [24], consequently, a malicious request can be blocked when it attempts to access critical web services that was defined by the web administrator.

Rupali D. Kombade et. al., (2012) proposed CSRF Vulnerabilities and Defensive Techniques, which compared various defense mechanisms to analyze the best defense mechanism, to build strong and robust CSRF protection mechanism based on [24].

Hossain Shahriar et. al., (2010) proposed Client-Side Detection of Cross-Site Request Forgery Attacks, which testes the Current browser-based detection methods are based on cross-origin policies that allow white listed third party websites to perform requests to a trusted website and concludes that they are not Effective. Based on [15], to overcome an attacker's attempt to circumvent form visibility checking, the proposal compares the response content type of a suspected request with the expected content type,

for which a prototype plug-in tool for the Firefox browser is being implemented and evaluated the approach on three real PHP programs vulnerable to CSRF attacks.

Xiaoli Lin et. al., (2009) proposed Threat Modeling for CSRF Attacks which illustrates the tree model for attacking. This proposal makes the Threat model for exploring, understanding and validating security protection features in Realistic Web Applications Scenarios. The different categories of vulnerabilities including HTTP Post Handling Error, Off-By-One-Error, unrestricted upload attachments and other unknown vectors. The Threat Model has two perspectives, Adversarial and defensive. Based on [4], the defensive perspective identify and remove as many vulnerabilities as possible. The Adversarial Perspective identify the holes and vulnerabilities and exploit them to gain access to the Objective.

Adam Barth et. al., (2008) proposed Robust Defenses for Cross-Site Request Forgery, in which the proposal mainly focused on the Login CSRF attacks by CSRF Threat Model. Based on [1], even though the https for the home page provides enough security to the vulnerable content, this system proposes to use the Reference Header in the Network Layer to enhance the security by protecting the Original Header. If the website has any images and Third party content, they are validated by a Session token on Ruby on Rails Framework. The session token generation is managed by the HMAC. The Session Initialization related vulnerabilities and defenses for Open ID, PHP cookie less sessions and HTTP Sessions are managed for the Users.

H.Mirvaziri et. al., (2007) proposed a New Algorithm of Hash Function based on the Combination of the Existing Digest Algorithms,

which implements MD5, SHA-1 and RIPEMD Algorithms. The proposal [10] present several simple message pre-processing techniques and show how the techniques can be combined with MD5 or SHA-1 so that applications are no longer vulnerable to the known collision attacks.

Chris Karlof et. al., (2007) proposed Dynamic Pharming Attacks and Locked Same-Origin Policies for Web Browsers, which demonstrates hijacking DNS and sending the victim's browser malicious JavaScript, which then exploits DNS rebinding vulnerabilities and the name-based same-origin policy to hijack a legitimate session after authentication has taken place. As a result, the attack works regardless of the authentication scheme used. Based on [12], dynamic pharming enables the adversary to eavesdrop on sensitive content, forge transactions, sniff secondary passwords, etc. To counter dynamic pharming attacks, this proposal suggests two locked same-origin policies for web browsers.

Omar ISMAIL et. al., (2004) proposed an Implementation of Automatic Detection/Collection System for Cross-Site Scripting Vulnerability, which depicts a client-side system that automatically detects XSS vulnerability by manipulating either request or server response. The system also shares the indication of vulnerability via a central repository. The purpose of [16] is twofold: to protect users from XSS attacks, and to warn the web servers with XSS vulnerabilities.

CHAPTER 3

SYSTEM DESIGN

In this chapter, the various UML diagrams for the BCX model for Cross-site request forgery is represented and the various functionalities are explained.

3.1 UNIFIED MODELLING LANGUAGE

Unified Modeling language (UML) is a standardized modeling language enabling developers to specify, visualize, construct and document artifacts of a software system. Thus, UML makes these artifacts scalable, secure and robust in execution. It uses graphic notation to create visual models of software systems. UML is designed to enable users to develop an expressive, ready to use visual modeling language. In addition, it supports high-level development concepts such as frameworks, patterns and collaborations. Some of the UML diagrams are discussed.

3.1.1 USE CASE DIAGRAM OF BCX MODEL FOR CROSS-SITE REQUEST FORGERY

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analysed the functionalities are captured in use cases. So it can be said that uses cases are nothing but the system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system. The actors can be human user, some internal applications or may be some external applications.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly

design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified.

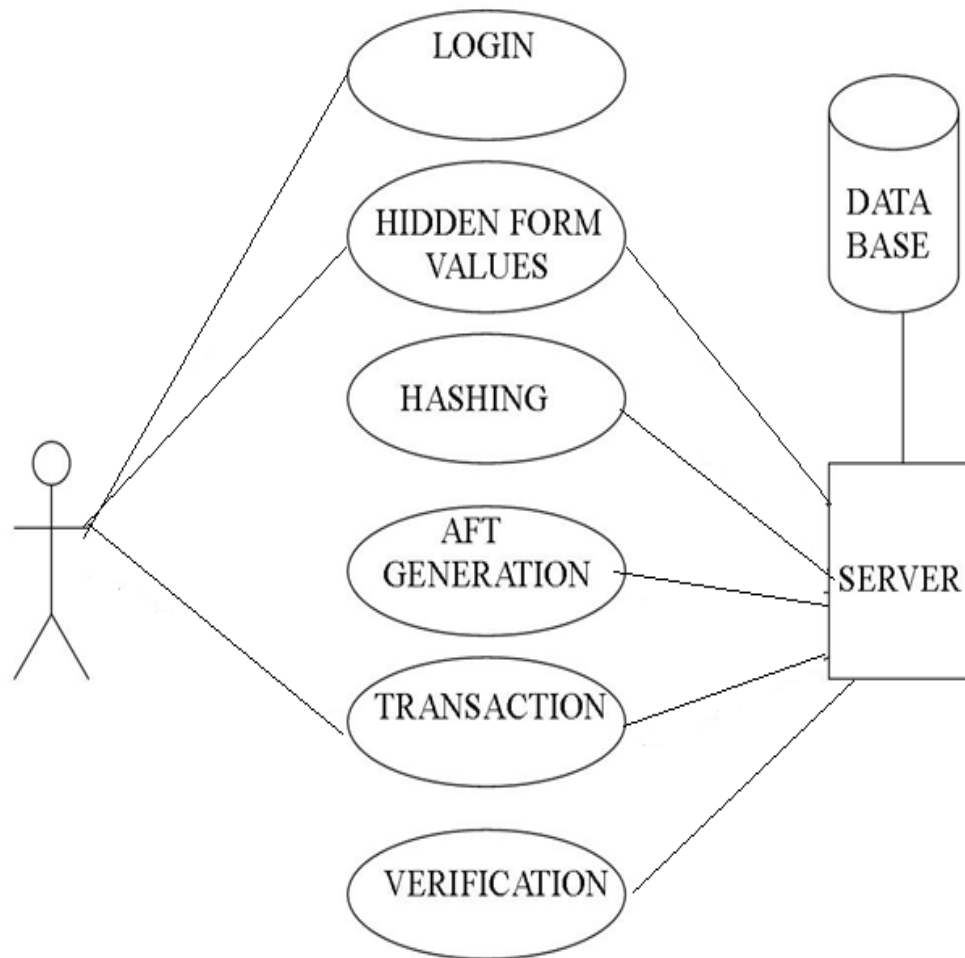


Figure 3.1 Use case diagram of BCX Model for Cross-Site Request Forgery

The Functionalities are to be represented as a use case in the representation. Each and every use case is a function in which the user or the server can have the access on it. The names of the use cases are given in such a way that the functionalities are preformed, because the main purpose of the functionalities is to identify the requirements.

To add some extra notes that should be clarified to the user, the notes kind of structure is added to the use case diagram. Only the main relationships between the actors and the functionalities are shown because all the representation may collapse the diagram. The use case diagram as shown in Figure 3.1 provides details based on the Cross-site Request Forgery application.

3.1.2 CLASS DIAGRAM OF BCX MODEL FOR CROSS-SITE REQUEST FORGERY

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system.

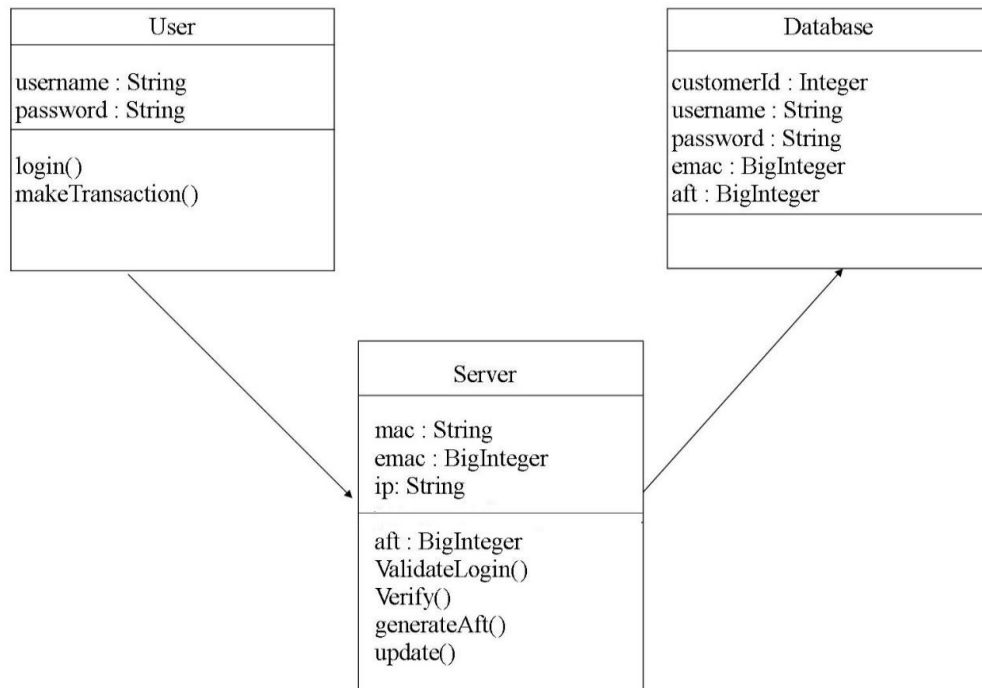


Figure 3.2 Class diagram of BCX Model for Cross-Site Request Forgery

The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified. All of these specifications for the system is displayed as a class diagram in Figure 3.2.

3.1.3 SEQUENCE DIAGRAM OF BCX MODEL FOR CROSS-SITE REQUEST FORGERY

UML sequence diagrams model the flow of logic within the system in a visual manner, enabling to both document and validate the logic, and are commonly used for both analysis and design purposes.

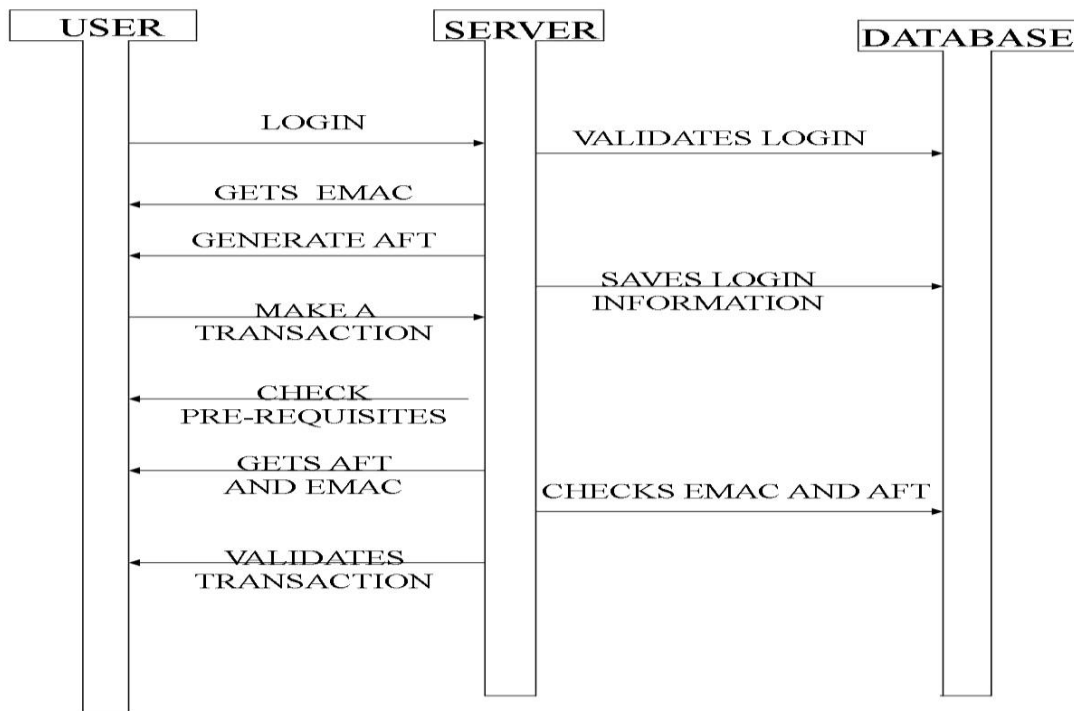


Figure 3.3 Sequence diagram of BCX Model for Cross-Site Request Forgery

The various actions that take place in the application in the correct sequence are shown in Figure 3.3 Sequence diagrams are the most popular UML for dynamic modeling.

3.1.4 ACTIVITY DIAGRAM OF BCX MODEL FOR CROSS-SITE REQUEST FORGERY

Activity is a particular operation of the system. Activity diagram is suitable for modeling the activity flow of the system.

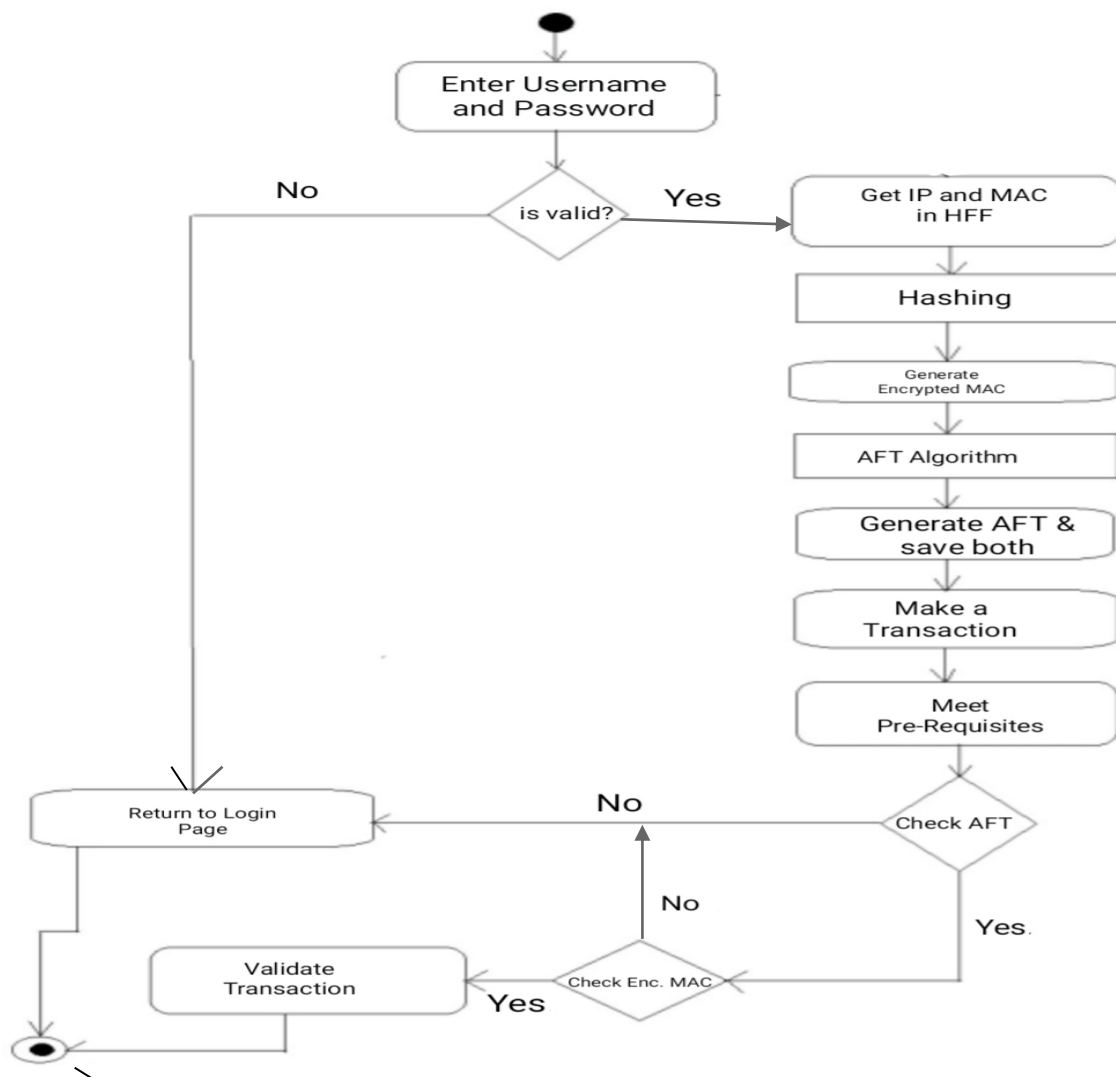


Figure 3.4 Activity diagram of BCX Model for Cross-Site Request Forgery

Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part.

An application can have multiple systems. Activity diagram also captures these systems and describes the flow from one system to another. This specific usage is not available in other diagrams. These systems can be database, external queues, or any other system. Activity diagram is suitable for modeling the activity flow of the system.

It does not show any message flow from one activity to another. Activity diagram is sometime considered as the flow chart. Although the diagrams looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single. The Figure 3.4 shows the activity diagram of the developed application.

3.1.5 COLLABORATION DIAGRAM OF BCX MODEL FOR CROSS-SITE REQUEST FORGERY

The next interaction diagram is collaboration diagram. It shows the object organization. Here in collaboration diagram the method call sequence is indicated by some numbering technique. The number indicates how the methods are called one after another. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization. The various objects involved and their collaboration is shown in Figure 3.5.

Now to choose between these two diagrams the main emphasis is given on the type of requirement. If the time sequence is important then sequence diagram is used and if organization is required then collaboration diagram is used.

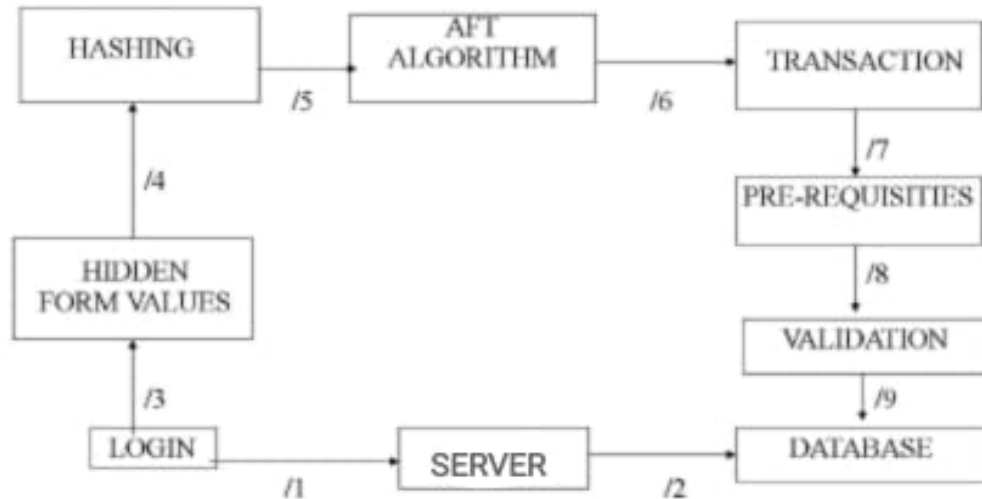


Figure 3.5 Collaboration diagram of BCX Model for Cross-Site Request Forgery

3.1.6 COMPONENT DIAGRAM OF BCX MODEL FOR CROSS-SITE REQUEST FORGERY

A component diagram displays the structural relationship of components of a software system. These are mostly used when working with complex systems that have many components such as sensor nodes, cluster head and base station. It does not describe the functionality of the system but it describes the components used to make those functionalities.

Components communicate with each other using interfaces. The interfaces are linked using connectors. The Figure 3.6 shows a component diagram.

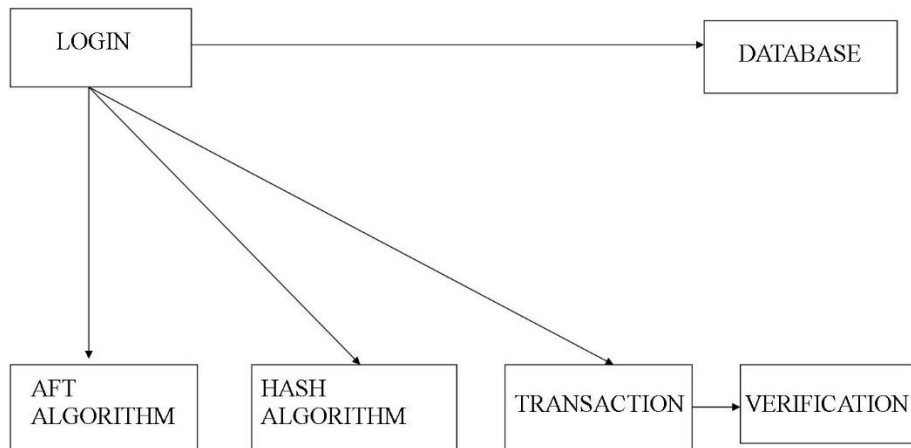


Figure 3.6 Component diagram of BCX Model for Cross-Site Request Forgery

3.1.7 DEPLOYMENT DIAGRAM OF BCX MODEL FOR CROSS-SITE REQUEST FORGERY

A deployment diagram shows the hardware of your system and the software in those hardware. Deployment diagrams are useful when your software solution is deployed across multiple machines such as sensor nodes, cluster head and base station with each having a unique configuration. The Figure 3.7 represents deployment diagram for the developed application.

Deployment Diagram in the figure 3.7 shows how the modules such as AFT Algorithm, Hash Algorithm, Transaction, Database gets deployed in the system.

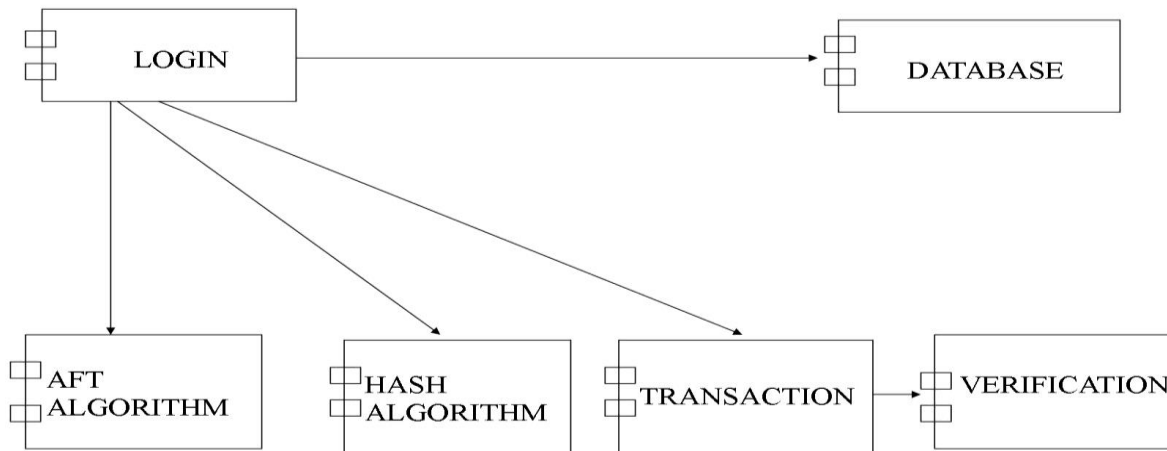


Figure 3.7 Deployment diagram of BCX Model for Cross-Site Request Forgery

3.1.8 PACKAGE DIAGRAM OF BCX MODEL FOR CROSS-SITE REQUEST FORGERY

Package diagrams are used to reflect the organization of packages and their elements. When used to represent class elements, package diagrams provide a visualization of the namespaces.

Package diagrams are used to structure high level system elements. Package diagrams can be used to simplify complex class diagrams, it can group classes into packages. A package is a collection of logically related UML elements.

Packages are depicted as file folders and can be used on any of the UML diagrams. The Figure 3.8 represents package diagram for the developed application which represents how the elements are logically related.

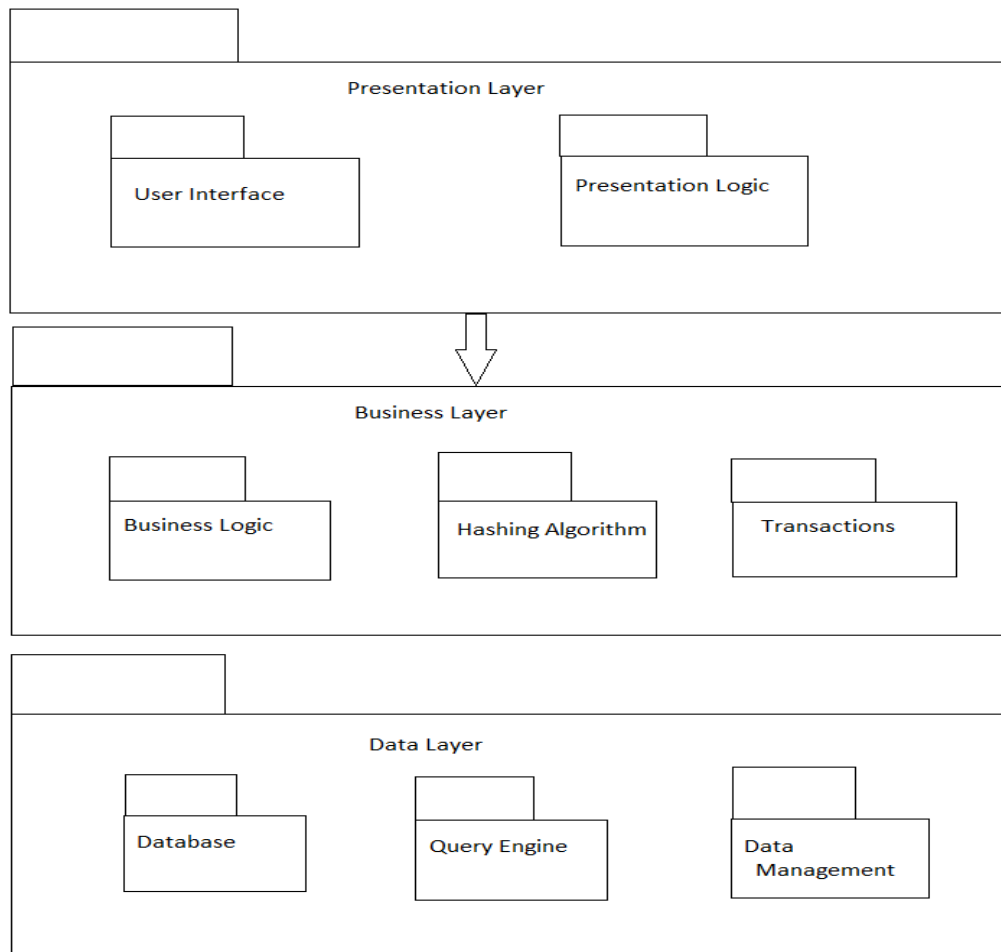


Figure 3.8 Package diagram of BCX Model for Cross-Site Request Forgery

Package is a namespace used to group together elements that are semantically related and might change together. It is a general purpose mechanism to organize elements into groups to provide better structure for system model.

3.1.9 OBJECT DIAGRAM OF BCX MODEL FOR CROSS-SITE REQUEST FORGERY

Object is an instance of a particular moment in runtime, including objects and data values. A static UML object diagram is an instance of a class diagram; it shows a snapshot of the detailed state of a system at a point in time, thus an object diagram encompasses objects and their relationships at a point in time.

An Object Diagram can be referred to as a screenshot of the instances in a system and the relationship that exists between them. Since object diagrams depict behavior when objects have been instantiated, we are able to study the behavior of the system at a particular instant. Object diagrams are vital to portray and understand functional requirements of a system.

Object Diagrams use real world examples to depict the nature and structure of the system at a particular point in time. Since we are able to use data available within objects, Object diagrams provide a clearer view of the relationships that exist between objects.

Object diagrams use a subset of the elements of a class diagram in order to emphasize the relationship between instances of classes at some point in time. They are useful in understanding class diagrams. They don't show anything architecturally different to class diagrams, but reflect multiplicity and roles.

Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams. Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system

but this static view is a snapshot of the system at a particular moment. Object diagrams are used to render a set of objects and their relationships as an instance.

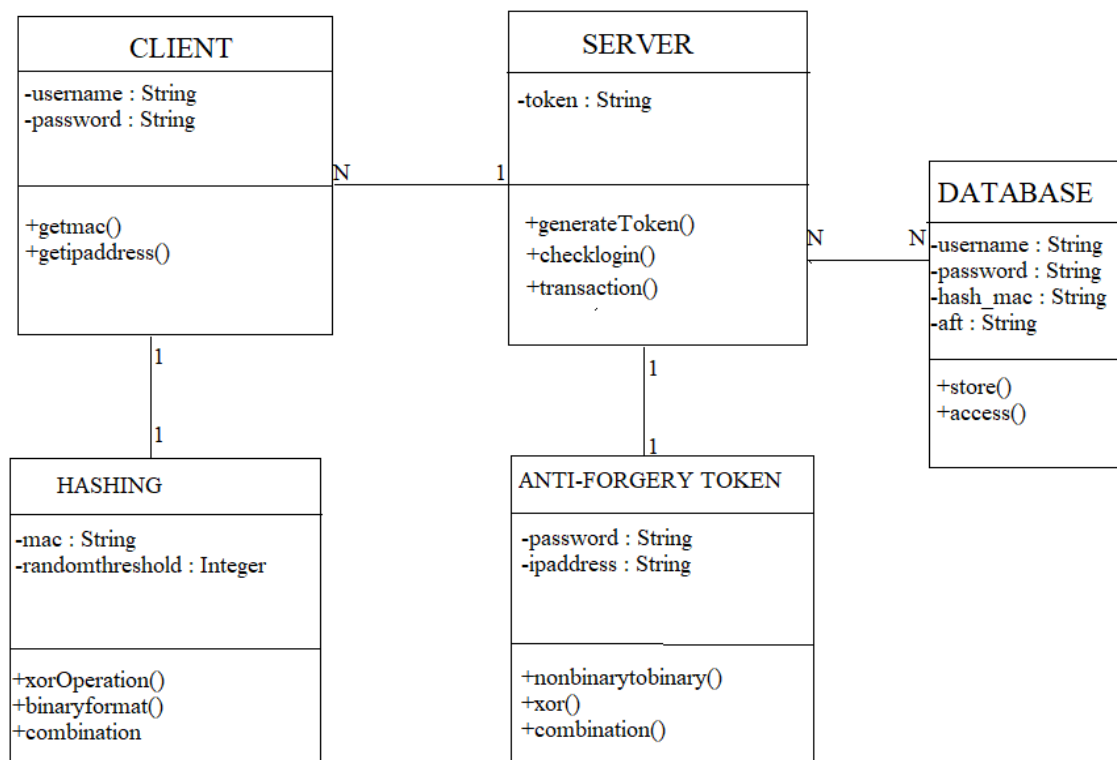


Figure 3.9 Object diagram of BCX Model for Cross-Site Request Forgery

The Figure 3.9 represents Object diagram for the developed application and also represents the relationship between the objects such as N:1 relationship between the client object and server object.

CHAPTER 4

SYSTEM ARCHITECTURE

In this chapter, the System Architecture for the BCX model for Cross-Site Request Forgery is represented and the modules are explained.

4.1 ARCHITECTURE DESCRIPTION

In system architecture the detailed description about the system modules and the working of each module is discussed as shown in figure 4.1.

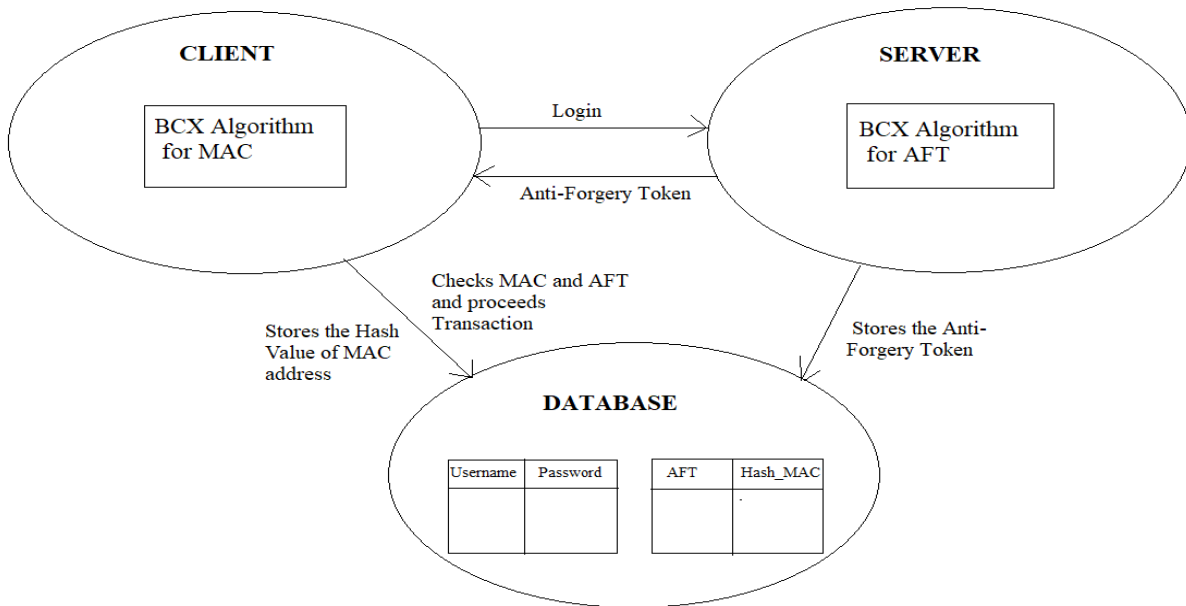


Figure 4.1 System Architecture of BCX model for Cross-Site Request Forgery

The Major modules of this proposal are Client Module, Server Module, Database, BCX Hash Algorithm for MAC Address and the BCX Hash Algorithm for Anti-Forgery Token generation. The user access the

web application from the server and the server verifies the identity of the client with the help of the database.

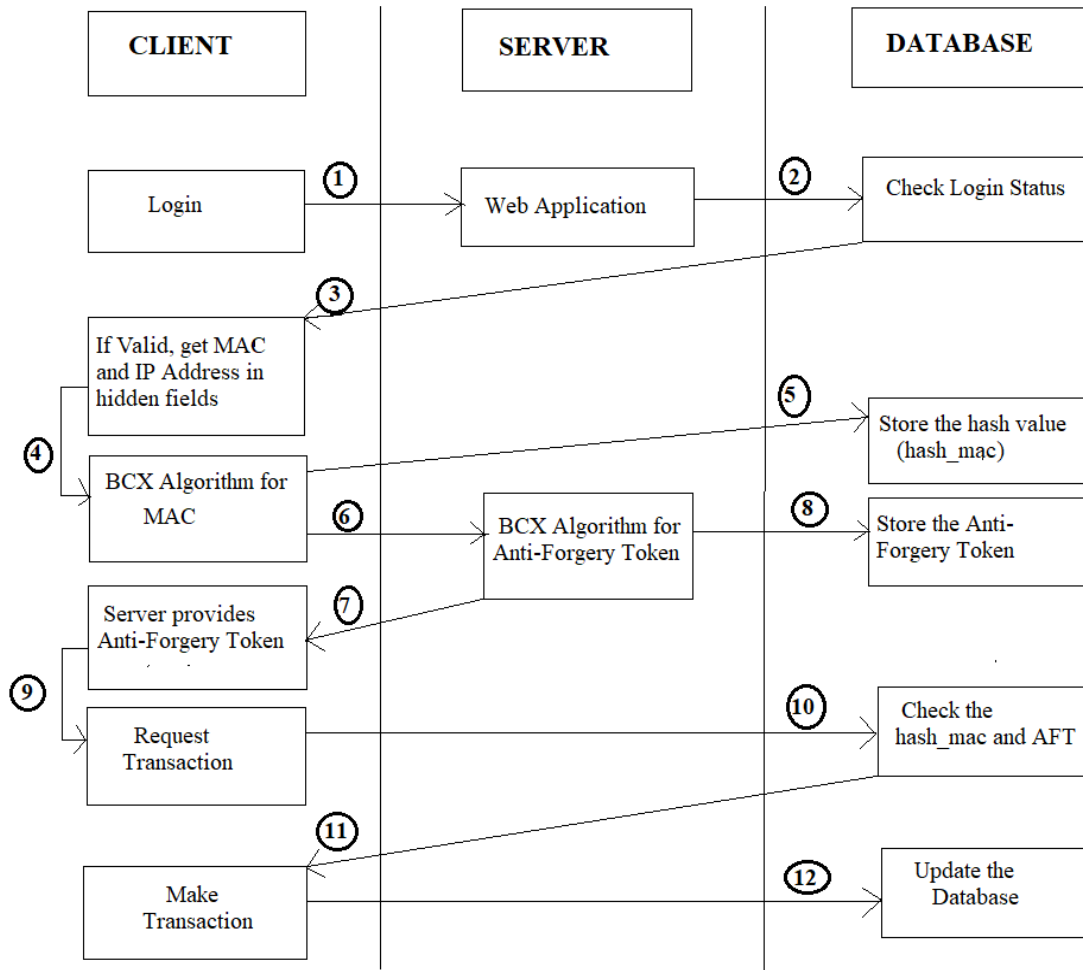


Figure 4.2 Interactions between Client Module, Server Module and Database

The Interaction between the client module, the server module and the database is represented in the figure 4.2. The Client module is in built with the BCX hash Algorithm for MAC Address, since sometimes when the

attackers get the unhashed MAC value of the user, it leads to a security threat to the user and hence only the hashed value of the MAC Address is passed to the server, so that the attackers may not know what it is all about.

The Attackers often change their IP Address and hence the Anti-Forgery token is based on the user's password and the IP Address. The Combination of both is a string and it is subjected to the BCX hash Algorithm for Anti-Forgery Token Generation. In both the cases of Login and the Transaction, the verification of the Anti-Forgery Token is made in order to confirm that the user who logs in make the transaction within the same session.

The database consists of two tables, one is the existing database of the web application which validates the login of the user in the Client module and the another table is newly created, which saves the entries for login action of every attempt that is made. The new table consists of the username, password, the hash value of the MAC Address from the BCX Algorithm and AFT at once it is generated at the time of login.

4.2 CLIENT MODULE

Whenever the user logs in, the username and the password of the user is verified by the existing Login Information, that is available in the corresponding Server. After the validation of the login information the BCX Algorithm uses a new table in the database to record the login activity of the user, for every new login that has been done. The new table makes an entry of the username, password, the hash value produced by the BCX hash Algorithm of the MAC Address. The server later generates an Anti-Forgery

Token by the same BCX Hash Algorithm by combining the current IP address taken from the hidden form field and the password of the user. The generated Anti-Forgery token is saved in the newly created table and the Anti-Forgery Token is being carried out to the Transaction page when the user makes the transaction. After all these procedures, the user enters into the Home Page.

The User enters into the Home Page after login validation, where the user has a lot of options about the Customization, Personalization of the Account and Transactions that can be made. Whenever the user wants to make the transaction, the user clicks on the Transaction Icon.

The Transaction Page has the name of the beneficiary and the amount to be transferred. In the transaction page, the Beneficiary and the Amount to be transferred is fixed and when the user clicks 'OK' to proceed with the transaction.

4.2.1 BCX HASH ALGORITHM IN CLIENT MODULE FOR GENERATING HASH VALUE OF MAC ADDRESS

To generate the hash value for the MAC Address, the algorithm gets the input as the MAC Address and add each number in the Mac address and have it as Sum_Of_Mac. The Random Threshold value is set between 0 and the Sum_Of_Mac. Two values A and B are set such that A equals the Sum_of_Mac divided by the Random Threshold and B value equals to the Sum_Of_Mac modulo Random Threshold. If the value of A is greater, then compute AC_B else compute the value of BC_A . Initialize an Integer array of size 6, and put on the ASCII values of the MAC Address. The ASCII values are X-ORed with the Combination values and store the resultant values in the new array. The new array's values are subjected to Binary

conversion and after each string the next binary value is appended. The number of 1's and 0's are counted and concatenated as a Integer and named as Pre-hash value. The hash value of the MAC Address is obtained by computing the value of $Pre\text{-}Hash_value \oplus C_{Sum_Of_Mac}$. This value is stored as the hash value of the MAC Address in the database.

4.3 SERVER MODULE

The Server plays an important role in collecting the information and saving it to the database. Moreover, it has the important role of generating the Anti-Forgery Token by the BCX Hash Algorithm, which is built inside the server. In the transaction page, the Beneficiary and the Amount to be transferred is fixed and when the user clicks 'OK', the server takes the MAC address again from that page and given to the BCX algorithm in the client module. The hash value of the MAC Address is prepared. Then, the IP address of the current tab is taken and along with the password, the values are passed in the hidden form values to the server. The Anti-Forgery token is again generated with newly considered IP Address and password and checked with the Anti-Forgery token generated when the user login.

If the AFT is matched, then the hash value of the MAC Address is compared with the hash value of the MAC Address by the server, when the user logs in, in order to double confirm that the same user who login, is now making the transaction in the same session. If both the AFT and the hash value of the MAC Address is matched, the transaction is proceeded, otherwise the session is made time-out and the server immediately makes the session to be logged out.

4.3.1 BCX HASH ALGORITHM IN SERVER MODULE FOR GENERATING THE ANTI-FORGERY TOKEN

The Anti-Forgery Token is produced by combining the password of the user and the current IP Address of the Client's System. The password is made up of Letters and Numbers. The Non-Numeric terms in the password are identified and are converted into Numeric terms by their corresponding ASCII value and now all are Integers. Now, the non-binary integers are converted into Binary values to obtain a big Binary value and stored into the string called 'pwd_ip_hash'.

The IP Address is subjected to the String Tokenizer to remove the '.' and the non-binary terms are converted to Binary terms. This binary term is appended to the pwd_ip_hash. The string in the pwd_ip_hash is subjected to a split of 3^s and then they are converted into Decimal values and passed into the BCX hash algorithm to get the Anti-Forgery Token.

CHAPTER 5

SYSTEM IMPLEMENTATION

In this chapter, the System Implementation for the BCX model for Cross-site request forgery is explained in detail.

5.1 IMPLEMENTATION OF BCX MODEL FOR CROSS SITE REQUEST FORGERY

The Algorithm and the page designs are implemented as a Web Application in Netbeans IDE. Here, the various functionalities required for the application are implemented by coding them in Java. The various technologies used are Html, CSS, Servlets, Java Database Connectivity and Java Server Pages.

For every End user who uses the application, instantly gets the Login Screen. The Login Screen shown in the figure 5.1 is built using Java Server Pages. It contains Username and Password text boxes for the user to enter. It also contains two hidden form fields, one for the MAC address and the other for the IP address. The login credentials of the user is confirmed by Server1 only after the user clicks on the Login Button on the Login Screen. For every verified credentials, the hash value of the MAC address is computed and the IP address is taken in the Hidden Form Fields. For the Invalid credentials, the user is to get a Pop-up message indicating a wrong username or password.

In the Server1, the BCX Algorithm is coded and the MAC value id passed to it. It returns the hash value of the MAC address to Server2. The Server2 accepts the Hash value of the MAC address and it takes the IP address to generate the Anti-Forgery Token. The Server2 stores the hash

values in the database. There is a table named HASH in the database that stores the Hash value of the MAC address, the Anti-Forgery token, the Customer-Id and the Time Stamp. The Server2 returns the Anti-Forgery token to the Client. In the figure 5.1 Login Page is shown.



LOGIN

USERNAME :

PASSWORD :

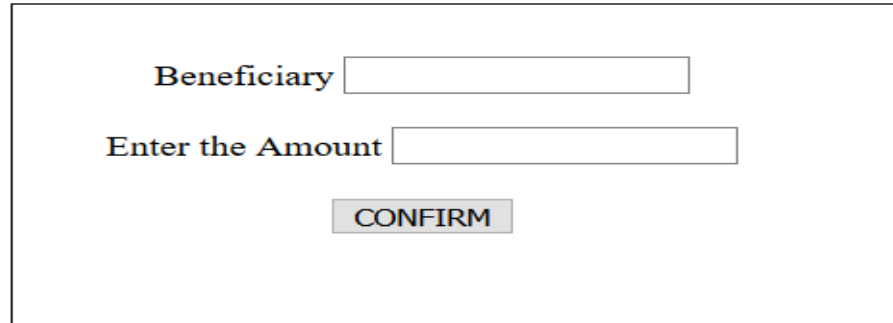
Figure 5.1 Login Page

After getting a successful login, the client enters into the Transaction Page shown in the figure 5.2 where the user can make the Money Transaction. The user gets the text box to enter the beneficiary and the amount to be transferred. The user can enter the person's account number to whom the money is to be transferred and the amount.

The amount is checked with the balance amount of the user for Money sufficiency. If the money is not sufficient to make the Transaction, the user is indicated by a decent message as Insufficient balance. When the money is sufficient to make a transaction the user gets the Confirm button to make the amount Transaction and the balance is displayed on another screen.

Whenever the confirm button is clicked, the current MAC address is taken in the Hidden form fields. The MAC address is forwarded to Server1 to call the BCX Algorithm to compute the Hash Value. The hash value is then passed to Server2 to check the Hash value of the MAC address and

the generated Anti-Forgery Token in the database. On a perfect match, the transaction is done.

A screenshot of a web form titled "Transaction Page". It contains two input fields: "Beneficiary" and "Enter the Amount", each followed by a text box. Below these fields is a button labeled "CONFIRM".

Beneficiary

Enter the Amount

Figure 5.2 Transaction Page

The database contains two tables. The First Table is the login table in the database as shown in the table 5.1 that contains details of the user that are collected in prior and saved for Authentication. This table is used to validate the Login of the user from the client module.

Table 5.1 Structure of Login Table

CUSTOMER_ID	USERNAME	PASSWORD	BALANCE	RANDOM_VALUE

The Second table is the hash table, which creates a row for each login attempt made on the server with the values of username, password, hash value of the MAC Address and the Anti-Forgery Token to re-authenticate the user on making a Money Transfer, such that there is no third-party intervention in between the session. The table 5.2 shows the hash table.

The MAC_HASH_VALUE and ANTI-FORGERY TOKEN are generated using BCX Algorithm and are stored as Big Integers. The Timestamp is generated at the time of valid login.

Table 5.2 Structure of Hash Table

CUSTOMER_ID	MAC_HASH_VALUE	ANTI-FORGERY TOKEN	TIMESTAMP

5.2 PSEUDO CODE FOR GENERATING HASH VALUE OF MAC ADDRESS

INPUT : MAC ADDRESS

OUTPUT : MAC HASH VALUE

START

GET THE MAC ADDRESS

COMPUTE SUM_OF_MAC VALUE

GENERATE RANDOM VALUE BETWEEN 0 AND SUM_OF_MAC

COMPUTE a VALUE AND b VALUE :

$$a = \text{SUM_OF_MAC} / \text{RANDOM VALUE}$$

$$b = \text{SUM_OF_MAC} \% \text{RANDOM VALUE}$$

IF $a > b$,

$$\text{COMPUTE VALUE_OF_COMBINATION} = {}^aC_b$$

ELSE

COMPUTE $\text{VALUE_OF_COMBINATION} = {}^b\text{C}_a$

GENERATE ASCII TOKENS FOR MAC ADDRESS

STORE THE TOKENS IN AN ARRAY

PERFORM X-OR OPERATION :

$\text{ASCII VALUES} \oplus \text{VALUE_OF-COMBINATION}$

CONVERT X-OR TERMS INTO BINARY FORM

STORE THE BINARY TERMS IN A STRING

COUNT THE NUMBER OF 0's AND 1's IN THE BINARY STRING

APPEND THE VALUES AND STORE IT AS NEW_VAL

$\text{NEW_VAL} = 1's + 0's \text{ (CONCATENATION)}$

PERFORM FINAL COMBINATION :

$\text{FINAL_COMBINATION} = {}^{\text{NEW_VAL}}\text{C}_{\text{SUM_OF_MAC}}$

HASH VALUE OF MAC ADDRESS = FINAL_COMBIANTION

STOP

5.3 PSEUDO CODE FOR GENERATING ANTI-FORGERY TOKEN

INPUT : IP ADDRESS AND PASSWORD

OUTPUT : ANTI-FORGERY TOKEN

START

CONVERT THE NON-NUMERIC TERMS INTO NUMERIC TERMS IN
THE PASSWORD

TOKENIZE THE IP ADDRESS INTO A NUMBER

APPEND MODIFIED PASSWORD AND THE TOKENIZED IP ADDRESS

CONVERT THE NON-BINARY TERMS INTO BINARY TERMS

SPLIT THE BINARY VALUES WITH THE SIZE OF 3

CONVERT THE SPLIT VALUES INTO THE EQUIVALENT DECIMALS

GIVE THIS AS INPUT STRING TO THE BCX ALGORITHM FOR
GENERATING THE HASH VALUE

FIND THE ANTI-FORGERY TOKEN

STOP

CHAPTER 6

CODING AND SCREENSHOTS

The following coding corresponds to the BCX Algorithm for MAC Address. The Input to the code is the MAC Address. The Output corresponds to the MAC Hash Value.

6.1 BCX MODEL FOR CROSS-SITE REQUEST FORGERY

```
package Clients;

import Server.Server;

import java.math.BigInteger;

import java.net.*;

import java.sql.*;

import java.util.*;

import java.util.logging.*;

public class Bcx_alg {

    protected    BigInteger    bcx(int    customer_id)    throws    SocketException,
    UnknownHostException{

        Connection conn = null;

        Statement st=null;

        ResultSet rs=null;

        InetAddress inetaddress=InetAddress.getLocalHost();
```

```

NetworkInterface network = NetworkInterface.getByInetAddress(inetAddress);

byte[] macArray = network.getHardwareAddress();

StringBuilder str = new StringBuilder();

for (int i = 0; i < macArray.length; i++)

{

str.append(String.format("%02X%s", macArray[i], (i < macArray.length - 1) ? "-" :
""));

}

String mac = null;

mac=str.toString();

System.out.println("MAC Address "+mac);

int cid=customer_id;

int sumofmactoken=0;

String mac_combined="";

int sum_of_mac=0,random_t1,value_a,value_b;

int half_fact,full_fact,value_of_combination;

int temp,single_digit,String temp_token;

int ascii[]=new int[6],i=0,j;

int new_array_xor[]=new int[6];

```

```

int bin_val,bin_str_int,one=0,zero=0,newval,count;

String binary_str="",updated="";

StringTokenizer token=new StringTokenizer(mac,"-");

while(token.hasMoreTokens())

{

mac_combined+=token.nextToken();

}

System.out.println(mac_combined);

char[] mac_array = mac_combined.toCharArray();

for (char output : mac_array)

{

int val=(int)output;

if(output=='A' || output=='B' || output=='C' || output=='D' || output=='E' ||
output=='F')

val=val-55;

else

val=val-48;

sum_of_mac+=val;

}

```

```

System.out.println("Sum of MAC Address "+sum_of_mac);

try

{

Class.forName("com.mysql.jdbc.Driver");

conn  =  DriverManager.getConnection("jdbc:derby://localhost:1527/csrf_bcx",
"csrf", "csrf");

st=conn.createStatement();

String query1="select * from login";

rs=st.executeQuery(query1);

int flag=0,random_value = 0;

while(rs.next())

{

if(rs.getInt("customer_id")==cid)

{

if(rs.getInt("random_value")==0)

flag=1;

else

random_value=rs.getInt("random_value");

break;

```

```

    }

    }

}

catch (SQLException ex)

{

    Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);

}

if(flag==1)

{

    Random rand = new Random();

        random_t1 = rand.nextInt( sum_of_mac);

    }

else

        random_t1=random_value;

try

{

    Class.forName("com.mysql.jdbc.Driver");

        conn = DriverManager.getConnection("jdbc:derby://localhost:1527/csrf_bcx",

"csrf", "csrf");

```

```

st=conn.createStatement();

String query="select * from login";

rs=st.executeQuery(query);

String final_update_query="update login set random_value=? where
customer_id=?";

while(rs.next())

{

if(rs.getInt("customer_id")==cid)

{

PreparedStatement stmt = conn.prepareStatement(final_update_query);

stmt.setInt(1,random_t1);

stmt.setInt(2,cid);

stmt.executeUpdate();

break;

}

}

}

catch (SQLException ex)

{

```



```

Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);

}

value_a=sum_of_mac/random_t1;

value_b=sum_of_mac%random_t1;

System.out.println("Random value between 0 and sum_of_mac = "+random_t1);

System.out.println( "value_a = "+value_a+" value_b = "+value_b);

if(value_a>=value_b)

{

half_fact=half_factorial(value_a,value_b);

    full_fact=full_factorial(value_b);

    value_of_combination=half_fact / full_fact;

}

else

{

half_fact=half_factorial(value_b,value_a);

    full_fact=full_factorial(value_a);

    value_of_combination=half_fact / full_fact;

}

System.out.println("value of combination = "+value_of_combination);

```

```

StringTokenizer token1=new StringTokenizer(mac,"-");

while(token1.hasMoreTokens())

{

temp_token=token1.nextToken();

    try

    {

temp=Integer.parseInt( temp_token)+96;

    }

    catch(Exception e)

    {

temp=0;

        char[] temp_array=(temp_token).toCharArray();

        for(char output : temp_array)

        {

int val=(int)output;

if(output=='A' || output=='B' || output=='C' || output=='D' || output=='E' ||
output=='F')

val=val-55;

else

```

```

val=val-48;

        temp+=val;

    }

temp+=96;

    }

while(temp>123)

temp-=26;

    ascii[i++]=temp;

}

System.out.println(" ASCII values of MAC tokens \n ASCII code ");

for(i=0;i<6;i++)

System.out.println(ascii[i]+"");

for(i=0;i<6;i++)

new_array_xor[i]=      (ascii[i]|value_of_combination)      &      (~ascii[i]      |
~value_of_combination);

System.out.println(" Exclusive OR (value_of_combination and ASCII values of
MAC tokens)\n XOR value ");

for(i=0;i<6;i++)

System.out.println(new_array_xor[i]+"");

System.out.println(" Converting XOR terms into binary terms");

```

```

for(i=0;i<6;i++)

{

bin_val=new_array_xor[i];

    while(bin_val > 0)

    {

temp=bin_val % 2;

    if(temp==1)

one++;

else

zero++;

binary_str= binary_str + "" + temp;

bin_val=bin_val / 2;

    }

}

System.out.println("binary string = "+binary_str);

System.out.println("Count the number of 1's and 0's");

System.out.println("Number of one's = "+one);

System.out.println("Number of zero's = "+zero);

updated=updated+one;

```

```

updated=updated+zero;

System.out.println("Number of 1's and 0's combined "+updated);

newval=Integer.parseInt(updated);

value_a=newval;

value_b=sum_of_mac;

BigInteger fact= BigInteger.ONE;

count=value_b;

for ( i = value_a; count>0; i--,count--)

fact = fact.multiply(new BigInteger(String.valueOf(i)));

BigInteger fact1= BigInteger.ONE;

for ( i = 2; i <= value_b; i++)

fact1 = fact1.multiply(new BigInteger(String.valueOf(i)));

BigInteger fact_div= BigInteger.ONE;

fact_div=fact.divide(fact1);

System.out.println("MAC Hash Value = "+fact_div);

return fact_div;

}

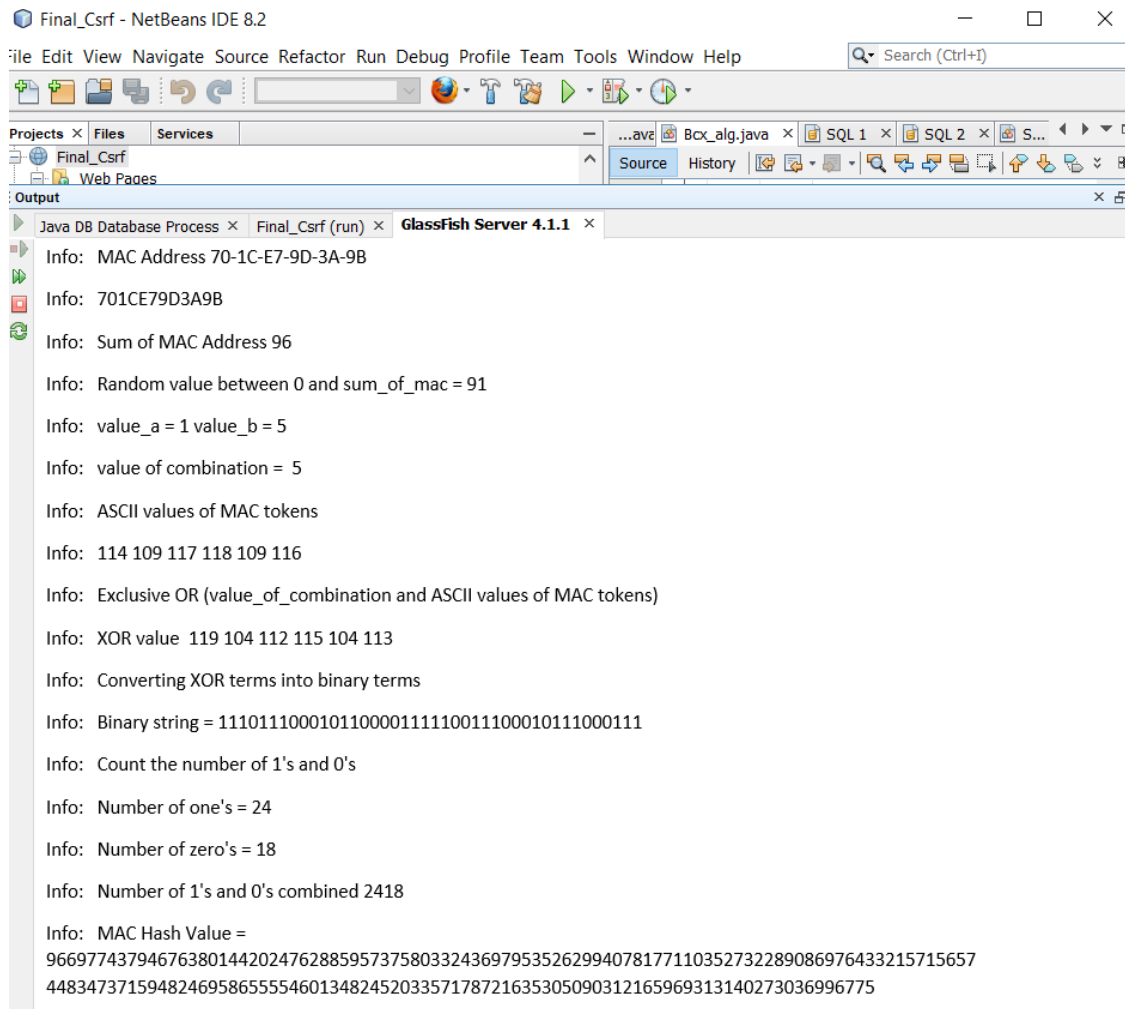
}

```

6.2 OUTPUT AND COMPARISON

6.2.1 HASH VALUE OF THE MAC ADDRESS

The implementation of the algorithm is done to generate the hash_mac value and the following figure 6.1 represents the derivation of the hash value of the MAC address.



```
Final_Csrf - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
...ave Bcx_alg.java x SQL 1 x SQL 2 x S...
Source History
Output
Java DB Database Process x Final_Csrf (run) x GlassFish Server 4.1.1 x
Info: MAC Address 70-1C-E7-9D-3A-9B
Info: 701CE79D3A9B
Info: Sum of MAC Address 96
Info: Random value between 0 and sum_of_mac = 91
Info: value_a = 1 value_b = 5
Info: value of combination = 5
Info: ASCII values of MAC tokens
Info: 114 109 117 118 109 116
Info: Exclusive OR (value_of_combination and ASCII values of MAC tokens)
Info: XOR value 119 104 112 115 104 113
Info: Converting XOR terms into binary terms
Info: Binary string = 111011100010110000111110011100010111000111
Info: Count the number of 1's and 0's
Info: Number of one's = 24
Info: Number of zero's = 18
Info: Number of 1's and 0's combined 2418
Info: MAC Hash Value =
96697743794676380144202476288595737580332436979535262994078177110352732289086976433215715657
448347371594824695865554601348245203357178721635305090312165969313140273036996775
```

Figure 6.1 Hash value of the MAC address

The MAC Address of the client is taken and sum of MAC Address is computed. The Random value is chosen. The value_a and value_b is computed. The Value of Combination is calculated. The ASCII value of

MAC tokens is generated. The X-OR operation is carried out between ASCII tokens and the Value of Combination. The count of 1's and 0's are taken from the binary representation of X-OR terms. The Final Combination value is calculated by using the count values and the sum of MAC.

6.2.2 GENERATION OF THE ANTI-FORGERY TOKEN

The following figure 6.2 represents the derivation of the Anti-Forgery Token using client's IP Address and Password.

```

Final_Csrf - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)

Projects Files Services
Final_Csrf
Web Pages

Output
Java DB Database Process x Final_Csrf (run) x Glassfish Server 4.1.1 x
Info: Password = csrf
Info: Converting Password into Number format
Info: 99115114102
Info: Converting Number Format Into Binary Format
Info: 1-0-0-1-1-0-0-1-1-1-1-0-1-1-1-1-0-0-1-0-1-0
Info: IP Address = 192.16.80.4
Info: Tokenized IP Address = 19216804
Info: Combined Binary Format of IP Address and Password
Info: 100110011110111100101011001101110100001
Info: Splitting the binary values
Info: AFT initial String4636745315641
Info: random = 3 value_a = 18 value_b = 1
Info: ASCII code 0 0 99 0 0
Info: Exclusive OR (value_of_combination and ASCII values of MAC tokens)
Info: XOR Values 18 18 113 18 18
Info: Converting XOR terms into binary terms 01001010011000111010010100101001
Info: Number of one's = 14 Number of zero's = 18
Info: Number of 1's and 0's combined 1418
Info: Combination of Count_val and sum of input token
Info: fact_big(n-r)
76109083351694464168417531151522318145791701736800506025952963423189938021975065543011825544
645214541262693000750138553911687521979311013182074138331454426316800000000000000
fact_small(value_b) 126964033536582759259651008475665169595803210514494367622758400000000000000
Info: Anti-Forgery Token :
59945388651948259780682805971047380065954876979319088724919544658946111889983242448472347074
04307520
  
```

Figure 6.2 Generation of the Anti-Forgery Token

At the Server Side, the Password and the IP Address of client are taken and are represented in number format. Each decimal number is then converted into appropriate binary number. The binary numbers are then separated with a split size of 3 and then their appropriate decimal number is obtained and appended to form the AFT initial string. The Random value is chosen. The value of combination is computed from the value_a and value_b. The X-OR operation is carried out and the terms are converted into binary values. The count of 1's and 0's in the binary string are taken. The final combination that corresponds to the Anti-Forgery Token is calculated using the count values and the sum of the initial AFT string terms.

6.2.3 RESULT ANALYSIS

The BCX Algorithm is compared with other hash algorithms such as MD5, SHA-512 and WHIRLPOOL. The Analysis is represented in the table 6.1.

Table 6.1 Comparison of Hash Algorithm's

Analyzing Subject	MD5	SHA-512	WHIRLPOOL	BCX	Better Algorithm
Output Length	128 bit	512 bit	512 bit	Variable Length	BCX
Memory Usage (bytes)	300060	402676	703556	363216	MD5
Processing Time (ms)	17233	45633	50267	23541	MD5
Time for Collision Attack	$O(2^{64})$	$O(2^{256})$	$O(2^{256})$	Infinite	BCX

By analyzing various properties of hash algorithm it is concluded that MD5 and BCX Algorithm has better performance. Though MD5 Algorithm is better in terms of Memory Usage and Processing Time, it does not provide as much as security provided by other algorithms. MD5 hash technique is easily broken by the collision attack and also it provides same hash value for the same set of data. Comparatively, in the BCX Algorithm the Brute-Force Attack and Collision Attack is not possible as the hash value is of variable length. Thus BCX Algorithm is more secure for computing hash values.

The figure 6.3 shows the performance analysis of MD5, SHA-512, WHIRLPOOL and BCX based on memory utilization, processing time and the possibility of collision attack.

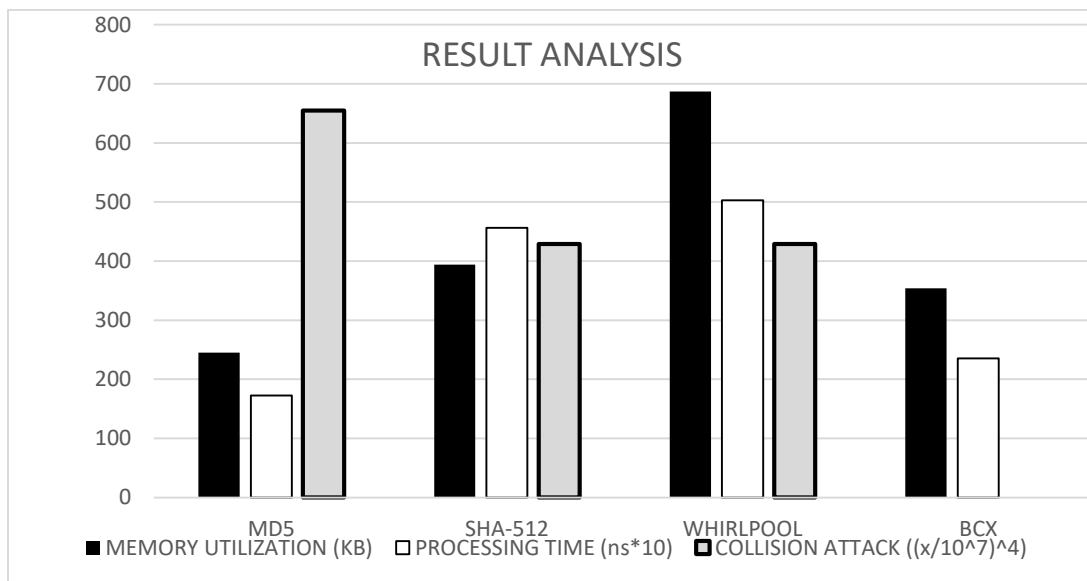


Figure 6.3 Performance Analysis

In the figure 6.3, Memory Utilization is mapped in terms of Kilo Bytes, Processing time in terms of nanoseconds, the possibility of Collision Attack based on the Order of time the attack takes place.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

Thus, the proposed system for Cross Site Request Forgery using the BCX Mathematical model, will reduce the risk of the money being stolen in the Online Websites. The usual remedies taken to reduce the Cross Site Request Forgery Attacks were solely based on the Network hacks and the Authentication mechanisms. The main concept of this proposal is to confirm the user actions in the same session. To reconfirm the actions of the user at the Client side and to verify that there is no third party interruption within the session, the MAC address is taken twice in the hidden form field and the hash value is stored in the database and checked on switching to a transaction. Apart from the MAC address, the Anti-Forgery token is produced from the IP address instead of any Random number that pays a way for the hackers to make an attack.

7.2 FUTURE WORK

The proposal made is completely based on the desktop transactions and hence for the mobile transactions, the IMEI number of the mobile on login session and it can be subjected to an efficient hash algorithm. The hash value of the IMEI number can be verified on making a transaction. By doing this, we can successfully avoid CSRF Attacks both on mobile phones and Desktop transactions.

REFERENCES

1. Adam Barth, Collin Jackson, John C. Mitchell (2008) 'Robust Defenses for Cross-Site Request Forgery'.
2. Purnima Khurana, Purnima Bindal (2014) 'Vulnerabilities and Defensive Mechanism of CSRF'.
3. Prathiba Yadav, Mr. Chandresh, D. Parekh (2017) 'CSRF Security Challenges and Prevention Techniques'.
4. Xiaoli Lin, Pavol Zavarisky, Ron Ruhl, Dale Lindskog (2009) 'Threat Modeling for CSRF Attacks'.
5. Tanjila Farah, Moniruzzaman Shojol, Md. Maruf Hassan, Delwar Alam (2016) 'Assessment of vulnerabilities of web applications of Bangladesh'.
6. Dudhatra Nilesh and Prof. Malti Nagle (2014) 'New Cryptography Algorithm with high Throughput'.
7. Madhumitha Panda (2016) 'Performance Analysis of Encryption Algorithms for Security'.
8. Fatma Kahri, Hassen Mestiri, Belgacem Bouallegue and Mohsen Machhout (2017) 'An Efficient Fault Detection Scheme for the Secured Hashing Algorithm SHA-512'.
9. Anak Agung Putri Ratna, Ahmad Shaugi, Prima Dewi Purnamasari, Muhammad Salman (2013) 'Analysis and Comparison of MD5 and

SHA-1 Algorithm Implementation in Simple-O Authentication based Security System’.

- 10.H.Mirvaziri, Kasmiran Jumari, Mahamod Ismail, Z. Mohd Hanapi, (2007) ‘New Algorithm of Hash Function based on the Combination of the Existing Digest Algorithms’.
- 11.Abhishek Bhardwaj and Subhranil Som (2016) ‘Study of Different Cryptographic Techniques and Challenges in Future’.
- 12.Chris Karlof, J.D. Tygar, David Wagner, Umesh Shankar (2007) ‘Dynamic Pharming Attacks and Locked Same-Origin Policies for Web Browsers’.
- 13.Yameng Haung, Zhouchen Lin (2018) ‘Binary Multidimensional Scaling for Hashing’.
- 14.Bhawna Mewara, Sheetal Bairwa and Jyoti Gajrani (2014) ‘Browsers Defense against Reflected Cross-Site Scripting Attacks’.
- 15.Hossain Shahriar and Mohammad Zulkernine (2010) ‘Client-Side Detection of Cross-Site Request Forgery Attacks’.
- 16.Omar ISMAIL, Masashi ETOH, Youki KADOBAYASHI, Suguru YAMAGUCHI (2004) ‘Implementation of Automatic Detection / Collection System for Cross-Site Scripting Vulnerability’.
- 17.Rupali D. Kombade, Dr. B.B. Meshram, (2012) ‘CSRF Vulnerabilities and Defensive Techniques’.
- 18.Marco Rocchetto, Mart in Ochoa and Mohammad Torabi Dashti (2014) ‘Model-Based Detection of CSRF’.
- 19.D. Kavitha, M.R. Akshaya, M. Karthick, K. Baghya, K. Gomathi Raja Eswari (2016) ‘Prevention of CSRF and XSS Security Attacks in Web Based Applications’.

- 20.Parimala (2018) 'Efficient Web Vulnerability Detection Tool for Sleeping Giant - Cross Site Request Forgery'.
- 21.Vikas K. Malviya, Saket Saurav, Atul Gupta (2013) 'On Security Issues in Web Applications through Cross Site Scripting (XSS)'.
- 22.Akash Agarwall, Maheshwari Shubh, Jagmohan, Projit Bandyopadhyay, Venkatesh Choppella (2017) 'Modeling and Mitigation of Cross-Origin Request Attacks on Federated Identity Management Using Cross Origin Request Policy'.
- 23.Wasim Akram Shaik, Rajesh Pasupuleti (2015) 'Avoiding Cross Site Request Forgery (CSRF) Attack Using Two Fish Security Approach'.
- 24.Yin-Chang Sung, Michael Cheng Yi Cho, Chi-Wei Wang, Chia-Wei Hsu, Shiuhyng Winston Shieh (2013) 'Light-Weight CSRF Protection by Labeling User-Created Contents'.
- 25.Nenad Jovanovic, Engin Kirda, and Christopher Kruegel (2006) 'Preventing Cross Site Request Forgery Attacks'.
- 26.Zhenqi Wang and Lisha Cao (2013) 'Implementation and Comparison of Two Hash Algorithms'.
- 27.Dave Ferguson (2009) '<http://appsecnotes.blogspot.com/2009/01/netflix-csrf-revisited.html>'
- 28.World Economic Forum (2018) '<https://www.weforum.org/reports/the-global-competitiveness-report-2018>'