International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 03 | Mar 2020 www.irjet.net p-ISSN: 2395-0072

Cross-site Scripting on Banking Application and mitigating attack using Honey Encryption Standard

Gopinath M.P.[1*], Shenbagavadivu S.[2], Hemanth Raj N[3], Karthick V.C.[4], Hari Prasad G[5]

[1,3,4,5]UG Students, Department of Information Technology, SRM Valliammai Engineering College, Tamil Nadu, India
[2]Assistant Professor, Department of Information Technology, SRM Valliammai Engineering College, Tamil Nadu, India

---

Abstract - *The rapid increase in the amount of data used in real-time applications and the advent of technologies such as Big Data has revolutionized today's data-driven world and has enabled the process of integrating such huge quantity of data with real-time applications. But, with such data comes the added issue of privacy which involves protecting such sensitive data from malicious fraudsters. This so-called secure data storage was achieved with the help of a process known as encryption. The problem associated with this technique is that it is highly vulnerable to password-based attacks. The attacker, on inputting a wrong key, receives an invalid plaintext message as a result of decryption. This enables the attacker to further proceed with the attack as the plaintext for the guessed key looks invalid. This paper proposes a technique known as Honey Encryption (HE) where the attacker gets redirected to a page comprising fake details of user accounts in case of a bogus attempt to log into the system. An online banking system has been developed in order to design, implement and evaluate the honey encryption mechanism.*

Key Words: Honey encryption, password-based attacks, brute force, cross-site scripting, cryptography

1. INTRODUCTION

In today's world, with the advent of digital technology, there has been an enormous rise in the amount of data generated by a vast number of applications. This tremendous increase has affected the industrial and corporate sectors in two primary aspects – one being the inevitable need of a huge number of resources for storage and the other being how securely the data can be stored. With the emergence of technologies such as big data and cloud computing, the former problem has been somewhat resolved with terms such as "data tsunami" being more commonly used in our day-to-day lives. But, having such large amount of data has made its security even more difficult. One tiny bug in a code may lead to the complete destroyable of data within a fraction of seconds. Therefore, it becomes equally important to secure the data, be it on premise, off-premise or in a specific configuration of a cloud, so that attackers can be prevented access to sensitive data and that data breaches can be less common.

With attacks such as brute force and denial of service becoming more and more common, the development of an effective and efficient detector system is of primary concern. Brute force attacks, in general, generate a large number of traffic as the attacker tries different combinations of usernames and passwords in order to successfully gain access into the application. This type of attack is known as a dictionary attack where the attacker uses a dictionary of already used most common usernames and passwords until a login attempt turns successful. This continuous application of a large number of combinations generate considerably large network traffic making the target server slow to respond to legitimate requests and in the worst case leading to the complete failure of the server due to a crash. Though many approaches have been already proposed, they provide security at the cost of other attributes. For example, current banking systems tend to block user accounts if more than three wrong guesses are made on the user's part. Though this blocks the user's account and provides a temporary solution, legitimate users may sometimes fall prey to this approach in case of a hacker brute forcing their credentials. This involves physical approach needed to be taken by the bank client in order to gain back access to the account.

In this paper, an efficient approach called Honey Encryption (HE) has been proposed which will be discussed in the following sections.

1.1 OBJECTIVE

The objectives of the proposed system include,

- Implementing a banking system with real-time features.
- Mitigation of Cross-site scripting techniques by fraudsters.
- Using HE to prevent access into credible bank accounts by malicious hackers.
- Encryption of bank details to provide wrong information to those with illegal access to legitimate bank accounts.

1.2 CHALLENGES

The challenges associated with the proposed system involve,

- In certain cases, such as ones involving very small message spaces, HE might fail thereby making the underlying data vulnerable to exploiters.
- In cases involving very large message spaces, it would take a large amount of time in order to finish the encryption and decryption process.
- As there is no central authority such as an admin involved, cases involving the leakage of original information rather than fake information might lead to privacy issues as the data would have already been in the hands of an attacker before the admin and the user getting notified.
- There might arise situations involving the blocking of legitimate user accounts in case the hacker bypassed the defence system thereby falling back to traditional account blocking approach.
- If the aforementioned situation takes place, then HE falls back to traditional technique where a higher authority such as a bank admin takes control of permanently deactivating or reactivating the account again.

2. LITERATURE SURVEY

[1] Fawaz Mahiuob Mohammed Mokbal, Wang Dan, Azhar Imran, Lin Jiuchuan, Faheem Akhtar, and Wang Xiaoxi propose a method that detects Cross Site Scripting using Multilayer Perceptron Technique. - The authors propose a method that detects Cross Site Scripting using Multilayer Perceptron Technique. This paper utilizes quality of data, appropriate feature vectors and ANN technique for the detection process.

[2] Junjie Wang, Bihuan Chen, Lei Wei, and Yang Liu Nanyang Technological University, Singapore 'Skyfire: Data-Driven Seed Generation for Fuzzing' – 2017 IEEE Symposium on Security and Privacy This paper propose a novel data-driven seed generation approach, named Skyfire, which leverages the knowledge in the vast amount of existing samples to generate well-distributed seed inputs for fuzzing programs that process highly-structured inputs.

[3] Michele Carminati, Mario Polino, Andrea Continella, Stefano Zanero – Politecnico di Milano "Security Evaluation of a Banking Fraud Analysis System" examine Banksealer, a decision support system for banking fraud analysis, This paper is based on evaluating the influence on the detection performance of the granularity at which the spending habits are modelled and its security against evasive attacks.

[4] Ari Juels_ and Thomas Ristenpart University of Wisconsin–Madison"Honey Encryption: Security Beyond the Brute-Force Bound" introduces honey encryption (HE), a simple, general approach to encrypting messages using low min-entropy keys such as passwords.

[5] Peter Kairouz, Pramod Viswanath ECE Department, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA. Sewoong Oh IESE Department, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA "The Composition Theorem for Differential Privacy" - In this paper, we answer the fundamental question of characterizing the level of overall privacy degradation as a function of the number of queries and the privacy levels maintained by each privatization mechanism.

[6] Tan Soo Fun -University Malaysia Sabah (UMS) Zarul Fitri Zaaba, Azman Samsudin Unibersiti Sains Malaysia "Enhanced Security for Public Cloud Storage with Honey Encryption" , this paper reviews the attack models and security defenses mechanisms of existing public cloud service providers. Subsequently, this paper extends the Honey Encryption scheme to enhance the file storage security on the public cloud computing.

[7] Abiodun Esther Omolara, Aman Jantan, Oludare Isaac Abiodun and Howard Eldon Poston "A Novel Approach for the Adaptation of Honey Encryption to Support Natural Language Message" Though these techniques provide confidentiality, they do not provide resilience against brute-force attacks. Honey encryption (HE) was proposed as a countermeasure to this problem of cryptography.

[8] Wei Yin, Jadwiga Indulska, and Hongjian Zhou North China Institute of Computing Technology, Beijing, China School of ITEE, The University of Queensland, Brisbane, QLD, Australia "Protecting Private Data by Honey Encryption" proposes the design of the distribution-transforming encoder (DTE). According to the probabilities of a message in the message space, it maps the message to a seed range in a seed space, then it randomly selects a seed from the range and XORs it with the key to get the cipher text.

3. MODULE DESCRIPTION

3.1 CLIENT MODULE

In this module, the existing user can login to the banking application. The user need to enter the username and password that has been sent by the bank via email. They can perform transactions and manage their account.

3.2 ADMIN MODULE

In this module, we are going to monitor the customers' accounts and providing security to their accounts using honey encryption. The admin account has special privileged rights in order to deactivate and reactivate any accounts, if found suspicious.

3.3 TRANSACTION MODULE

This module is used to create transactions for entry/release, billing, and account transferring. Customer can initiate a single transaction (with single debit and single credit) or add a group of such transactions and submit the group. Each transaction in such a group will be treated as independent transaction for limit purpose.

3.4 BLOCKED ACCOUNTS MODULE

This module helps to unblock the blocked accounts that are handled by the admin. It maintains a log of accounts that have been blocked.

3.5 INTRUDER MODULE

This module handles the unauthorized users to fool them by giving fake details whenever the suspicious user attempts to steal data. This also gives the identity of the intruder which can be captured.

4. ARCHITECTURAL DESIGN

The architecture of HE is shown in Fig -1. The actors involved in the system include Bank Database, Bank/Server Admin, Client and Hacker/Attacker. The various actions that can be performed between these actors include (1) Creating Client account and storing in Database (2) Mailing account credentials to respective clients (3) Clients logging into accounts in order to perform bank related actions (4) Hacker or attacker logging with illegal activity (5) Blocking the IP address/account of suspicious users. (6) Reporting illegal activity to the associated client and bank admin (7) Redirecting attackers to fake page using HE.

4.1 EXISTING SYSTEM

Traditional system of blocking accounts on wrong passwords tried for n attempts for hourly basis. System vulnerable to XSS based attacks. Legitimate users get blocked from accessing without their knowledge. Unauthorized users tries to access the legitimate account and gets the account blocked. Stealing of information in case of successful login by attacker. Existing system has Two Factor Authentication system for authentication.

4.2 PROPOSED METHOD

The proposed system of the use of Honey Encryption (HE) for preventing brute force and other password-based attacks works as follows. To implement this technique in order to mitigate such attacks, an online banking system architecture was built as a testbed in order to test the strategy in real-time. A banking system with options of creating and modifying accounts, transferring funds between registered accounts with additional options of knowing the account balance and so on was developed with the help of web tools. The system included the creation of also an admin account which enabled the bank admin to block suspicious accounts and reactivate wrongly blocked accounts. The admin had the rights of sending a registered user his/her credentials, the username and password respectively, on successful registration. Once some kind of suspicious activity such as a brute force attack is noted by the admin, the IP address associated with the activity can be traced and further login attempts from that particular address can be prevented by black-listing the IP address. Also, cross-site scripting attacks such as cookie stealing and so on are some of the vulnerabilities associated with web applications, more specifically banking applications.

Hackers use some of the flaws associated with popular websites by first analyzing the website. Then, they go on with the attack phase by carrying out various attacks such as SQL injection, XSS and password cracking attacks in order to break the system and thereby gain access to the data stored in the backend database.

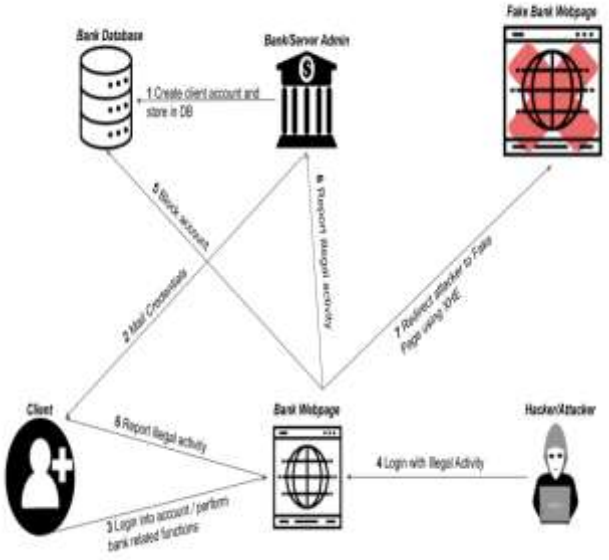

Fig -1: Architecture Diagram

SQL injection is popularly used by attacking injectable webpages by slightly modifying the query request passed to a page. This in turn renders unexpected results, in some cases displaying all the user credentials associated with the database of the website.

Therefore, it is important that applications built for protecting information must prevent injection attacks by deploying techniques such as input validation, escaping queries, preventing dynamic queries and using prepared statements. Another method most prevalently used for stealing important data pertains to cross-site scripting also known by XSS. In this case, the attacker loads malicious scripts in some other page which tries to send a request to the target application with the aim of gaining access to the elements such as scripts and the objects associated with the

target web application for the purpose of stealing cookies and modifying information.

All the browsers have same origin policy. Once a web page is logged in, we need not provide authentication details for every page in it, since they have same origin. Interaction between webpages from different domains can be through links, as the whole internet concept works on this method. Web documents can be accessed using DOM object. A DOM object from one domain cannot access contents on another domain. Same origin can be defined by Protocol, Hostname, Port but however URL's path is not considered part of the origin. For example, consider the following tag <script src = "app/some_url"></script> present in a site deployed by an attacker. This script is considered to have originated from attacker site due to same origin policy and not from the application. This allows attackers to initiate HTTP requests directly to the application's website thereby rendering malicious activity with the identity of a real user.

Cookie authentication is used in most of the web applications. There is a lifetime for cached cookies and HTTP authentication credentials. Let us consider a situation where a user has opened two tabs in a browser. Consider one tab has an online banking web page and the other has some news page. Cookie and session token for the banking application is stored in the browser. If some attackers manipulate the user to access the banking application from the advertisements on the news page, the attacker page may capture the cookie information and have a possibility to access the banking application using the identity of a registered user. Existing systems seem to be vulnerable to these type of attacks.

On the whole, there has to be some defense mechanism present in the application so that such dangerous threats can be averted in the future.

4.2. Online Banking System

This section consists of implementing an online system for banking with secure storage using popularly used development tools and techniques and explains about the process involved in case of an attack.

4.2.1 Implementation of MVC pattern for development of a web application

Model-View-Controller (MVC) resembles an architectural pattern most commonly used in the development of web applications. MVC gives the developer three primary layers to work with: (1) Model (2) View and (3) Controller. This model is popularly used as a standard design pattern in the web development process as it represents a complete framework. MVC model provides three different types of classes namely,

1) *Model* - The logic of data domains are mainly developed using the help of these model classes. The insertion, updation and retrieval of data into the database associated with the web application are carried out using the classes available in the model.
2) *View* – The primary aim of providing an easy-to-use and visually stunning user interface can be guaranteed with the use of classes of views. This interface allows bi-directional communication between the user and the application on the whole.
3) *Controller* - Controller classes are used to respond to the user's requests. Controller classes perform the users requested actions. These classes work with model classes and select the appropriate view that should be displayed to the user according to user requests.

4.2.2 Data Encryption using MD5 Algorithm

Once the banking application is created and deployed for real-time use by public customers, the bank administrator can help with the creation of user accounts in case users opt for one. This can be initiated by directly visiting the desired bank or by contacting the respective admin through any digital system. During the account creation process by the admin, the data which has been input into the system are encrypted for enabling a safe and secure storage mechanism. The proposed mechanism uses MD5 technique for storing passwords and data alike the backend database securely. The MD5 algorithm in cryptography is a one-way function accepting a message with its length as input and returning a message digest of fixed-length in order to authenticate the original message.

4.2.3 Hosting of the system on third-party Cloud Environment

In order to achieve the state attained by real-time systems, the application was hosted in a cloud environment ref Fig -2. The cloud environment can be of three types – private, public and protected. This banking application was deployed in a public cloud so as it provides continuous access to public users and users of other organizations. A cloud helps in balancing the load, providing fault tolerance and scaling resources on demand by using the already available components such as load balancers and techniques such as migration. On the developer's part, only the cloud provider and the type of service to be utilized has to be chosen. The cloud service provider will take care of the environment, rightful deployment and automatic update in the cloud environment.

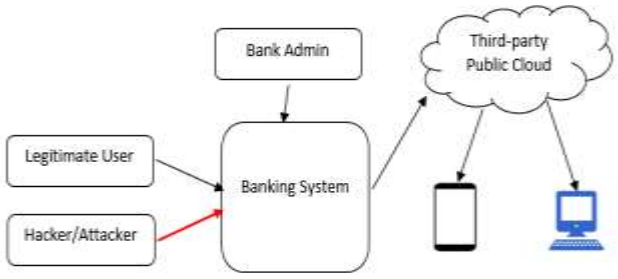

Fig -2: Hosting the banking application on a third-party cloud

4.2.4 Capturing IP addresses and image if possible, of suspicious client

In case of a brute force attack being detected, the system captures the IP address of the device trying its hand in the process. This can be performed by capturing the object from which the address of the suspicious client can be extracted. Once this is complete, the application blocks the device involved in this attempt from future access to the web application. Also, if possible, using the web camera access on the client-side systems, the image of the suspicious client can be taken, processed and transmitted to necessary actors for further action in the process.

4.2.5 Reporting suspicious access to bank administrator and client

After the execution of the aforementioned steps, the information regarding the suspicious login attempt is then forwarded to the bank administrator, the rightful owner of the account and other necessary actors. The rightful client can then approach the associated bank directly in order to request the service pertaining to reactivation of the blocked account to the administrator which can be done on successful verification.

4.3 Honey Encryption (HE)

This section puts forward the use of Honey Encryption in diverting attackers to a fake page which consists of user account information resembling data similar to real accounts.

4.3.1 Creation of a fake page

In order to deceive the hacker/attacker in making them believe that their login attempt was indeed correct and the goal of gaining access into the system was successfully achieved, the attackers were redirected to a fake page consisting of some random banking details such as account information ref Fig -3. This was done so that the brute force attempts can be effectively thwarted rather than blocking the accounts and IP address which is a really long process. Also, blocking accounts in cases where even legitimate users, without their knowledge, would have typed the wrong password multiple times is a daunting task and reactivating them is even more time-consuming. In the fake page attacker cannot do any usual operations like money transaction, balance enquiry, adding beneficiary and changing password. The attacker would rather try again or take the fake data as he would think it as original data. Only the legitimate user knows whether the displayed data is original or not. This method improves the security for authentication process and further enhances the security.

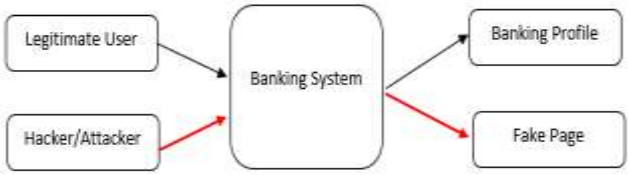

Fig -3: Redirecting illegitimate login attempts to a fake page

4.3.2 Encryption of client information

The data associated with the clients should be encrypted in some form such that data is secure in case of a data breach ref Fig -4. But, encrypting the data with some commonly used techniques will lead to the attacker being aware of the fact the key tried during the brute force process produced some gibberish text. This makes the attacker to know that the current key combination failed as the attacker reads some wrong text instead of getting the intended plain text message. This makes the attacker continue with the brute force process encouraging to not stop until the intended message has been retrieved. Therefore, the use of honey encryption encrypts the message in such a way that the plain text looks like a correct text but is actually some random generated string.

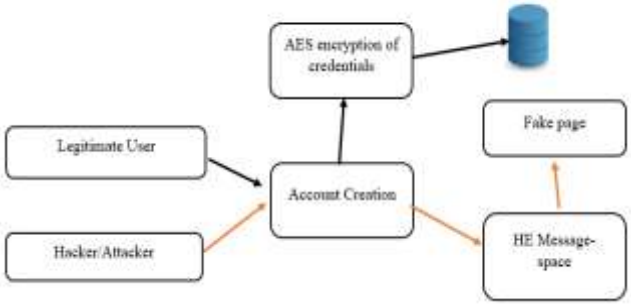

Fig -4: Using HE message space to display fake information

5. CONCLUSIONS

The existing system for preventing password-based attacks such as Brute force attack, Cross site scripting attack, Man in the middle attack deploy password-based mechanisms in which attackers are alerted by invalid-looking plaintext. In Honey Encryption, the messages displayed in case of wrongly guessed key values resemble those from a similar message space thereby fooling the attacker into believing that the desired information has been achieved. This technique is further extended by redirecting such attack attempts to a fake page which consists of user account information encrypted using HE. Even though HE has its own advantages, it comes with certain limitations: (1) Storage of a large number of messages to form the message space is a complex process; (2) HE takes time to encrypt messages where a large message space is involved.

REFERENCES

[1] Mokbal, Fawaz Mahiuob Mohammed, Wang Dan, Azhar Imran, Lin Jiuchuan, Faheem Akhtar, and Wang Xiaoxi. "MLPXSS: An Integrated XSS-Based Attack Detection Scheme in Web Applications Using Multilayer Perceptron Technique." IEEE Access 7 (2019): 100567-100580.

[2] Wang, Junjie, Bihuan Chen, Lei Wei, and Yang Liu. "Skyfire: Data-driven seed generation for fuzzing." In 2017 IEEE Symposium on Security and Privacy (SP), pp. 579-594. IEEE, 2017.

[3] [3] Carminati, Michele, Mario Polino, Andrea Continella, Andrea Lanzi, Federico Maggi, and Stefano Zanero. "Security evaluation of a banking fraud analysis system." ACM Transactions on Privacy and Security (TOPS) 21, no. 3 (2018): 1-31.

[4] Juels, Ari, and Thomas Ristenpart. "Honey encryption: Security beyond the brute-force bound." In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 293-310. Springer, Berlin, Heidelberg, 2014.

[5] Kairouz, Peter, Sewoong Oh, and Pramod Viswanath. "The composition theorem for differential privacy." IEEE Transactions on Information Theory 63, no. 6 (2017): 4037-4049.

[6] Fun, Tan Soo, Azman Samsudin, and Zarul Fitri Zaaba. "Enhanced security for public cloud storage with honey encryption." Advanced Science Letters 23, no. 5 (2017): 4232-4235.

[7] Omolara, Abiodun Esther, Aman Jantan, Oludare Isaac Abiodun, and Howard Eldon Poston. "A novel approach for the adaptation of honey encryption to support natural language message." In Proceedings of the International MultiConference of Engineers and Computer Scientists, vol. 1. 2018.

[8] Yin, Wei, Jadwiga Indulska, and Hongjian Zhou. "Protecting Private Data by Honey Encryption." Security and Communication Networks 2017 (2017).