# Title: Preventing the Bypassing of Client-Side Controls

Information Security Management (CSE3502)
J Component Project

Winter Semester 2022

1. Prithish Samanta - 19BCE2261
2. Ritvik Kohli - 19BCE2223

B.Tech. Computer Science and Engineering


School of Computer Science and Engineering

Vellore Institute of Technology

Vellore

April, 2022

## Abstract

1. Motivation

   According to recent studies, web apps are becoming one of the primary platforms for representing data and services distributed through the World Wide Web as the use of electronic devices grows. A number of flaws have been discovered in current online applications. A large number of online apps that run on the Internet are becoming increasingly insecure. As aspiring Web Developers, we're really interested in learning about the current issues that Web Designers are dealing with. We're excited to investigate and learn about the different ways our websites can be attacked, as well as how to protect them.

2. Aim

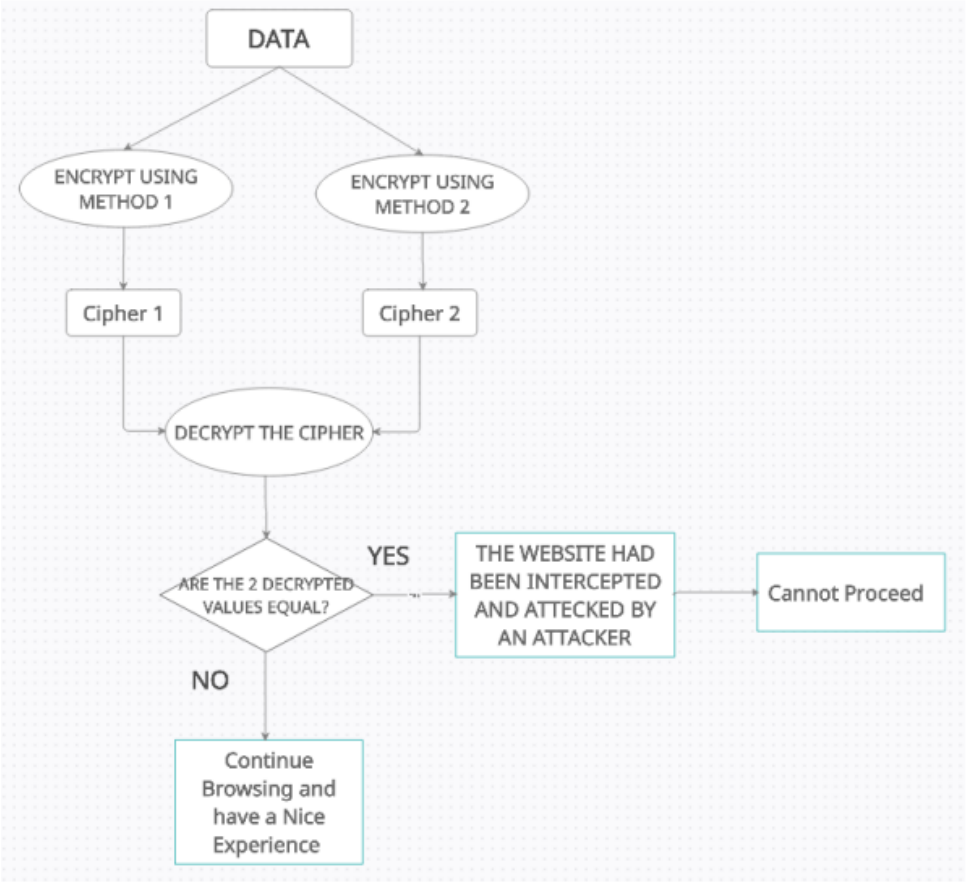
   The major goal of this project is to demonstrate how we can prevent the attackers from getting beyond some of the client-side limits that developers utilise, which can lead to serious web application vulnerabilities. This allows us to assess how safe the website is and how web attacks occur. We will also try to secure cookies using the same method in this project. Because HTTP is a stateless protocol, servers can identify users who make requests and provide them access when we utilise cookies.

3. Objective

   Encrypt Data and other information using 2 encryption algorithms.
   Encrypt the information stored in cookies and the info to be transferred so that the attacker does not have access to sensitive information.
   Decrypt the encrypted info properly and find out whether the data was tampered with by the attacker.
   Prevent the attacker from using the website if caught.

4. Methodology

   Encrypt the information 2 times separately using a different encryption algorithm and a different key each time.
   Send the Data to the new page after clicking the submit button.
   The attacker might intercept the data and change the information being sent.
   Because the data is encrypted by 2 different algorithms, when decrypted using the key their values might not turn out to be the same.
   As a result, we can assume that the data had been tampered with by the attacker and therefore stop the attacker from using the website further.




5. Expected Outcome

   a. When an attacker tampers with the data by intercepting the website, the act is caught and the attacker is logged out.

   b. If the user uses the website normally without tampering with the data, he or she should be able to use the website freely.

*Keywords - Bypass, Encryption, AES, Triple-DES, Intercept*

## Introduction

1. Overall Idea

   The central idea of our project is to detect and prevent data tampering by an attacker. To avoid this, we will encrypt the data twice, each time with a different cryptographic algorithm. If an attacker alters the information by intercepting data packets, the decrypted data values will change and no longer match. As a result, we can conclude that the data packet was intercepted by an attacker. Because the two variables were encrypted using two different algorithms, each with a random key, the decrypted values of the two keys have a very small chance of matching, and thus tampering will be detected most of the time.

2. Background

Through this project, we aim to detect and prevent the common mistakes done by developers with innovative prevention techniques. Here we are dealing with some uncommon security-related errors made by developers while creating websites and not common authentication techniques. We are showcasing how some silly mistakes/ignorance can completely compromise the access control of a web application. This project sets up minimum authentication criteria for websites, which web developers can use to make their websites secure.

3. Statistics

For encrypting data there are already many algorithms that are used. These algorithms include AES, Triple DES, DES, IDEA, RSA, etc. Each of these algorithms has its own advantages and disadvantages and need to be chosen based on the scenario.

- To carry out operations, AES employs bytes rather than bits. Because the block size is 128 bits, the cipher can handle 128 bits (or 16 bytes) of incoming data at a time.
- The number of rounds is determined by the length of the key as follows:
  - 10 rounds with a 128-bit key
  - 12 cycles using a 192-bit key
  - 14 rounds using a 256-bit key
- A Key Schedule algorithm is used to calculate all the round keys from the key. As a result, the initial key is used to generate a number of other round keys, each of which is used in the subsequent encryption round.
- Triple DES is a method of encryption that uses three DES instances on the same plaintext. It employs a number of key selection techniques, including The first has all keys that are different, the second has two keys that are the same and one that is different, and the third has all keys that are the same.
- Because Triple-DES is susceptible to a man-in-the-middle attack, it employs a total security level of 21112 rather than a 168-bit key. A block collision attack is also possible due to the small block size and the use of the same key to encrypt large amounts of text.

4. Advantages/Disadvantages of Proposed Method

- This technique is a good way to prevent attacks on small websites.
- It prevents the owner of the website from facing unwanted issues due to the tampering of data by the attackers.
- The technique which we are using to detect the tampering of data by an attacker, cannot be used for big websites of huge established businesses.

5. Advantages/Disadvantages of Related Methods

- There are a few other ways to prevent similar attacks like these. These attacks can be XSS attacks, Cookie Manipulation attacks, etc.
- According to the author of the paper “Bypassing XSS Auditor: Taking Advantage of Badly Written PHP Code”, the most crucial rule against XSS assaults is "Filter Input - Escape Output". When we escape output data, we make sure that the parser or interpreter cannot misinterpret the data. The clear examples are characters "<" and ">" which i HTML denotes element tags.
- If the characters could be added to an input provided by the user, an attacker would be able to enter additional tags that would be rendered by the browser. Therefore, developers should escape these specific characters by using the PHP function htmlspecialchars(). This function turns other special characters like "&," "" and “,,” into HTML entities. Escape can be done with the PHP htmlspecialchars() function.

## Literature Survey / Related Works

| Name, Author, Date | Summary/ Idea | Disadvantage |
|---|---|---|
| **1. Preventing Session Hijacking using Encrypted One-Time-Cookies By Renascence Tarafder Prapty, Shuhana Azmin, Md. Shohrab Hossain, Husnu S. Narman (2020)** | The authors propose a novel solution in this paper to ensure cookie confidentiality, authenticity, and integrity while also preventing session hijacking via replay and cookie poisoning attacks. They used a reverse proxy server to issue and verify one-time cookies as session cookies, as well as a custom cryptography operations module to manage encrypted one-time cookies. They conducted a security analysis in order to validate our proposed system. By using OTC instead of expensive HTTPS connections, they can prevent session hijacking via replay attacks and cookie poisoning attacks. | The only minor issue with this proposed work is that it takes slightly longer than the original method. The two operations (Cookie Data Encryption and Digital Signature Creation) take longer, but this has no effect on the proposed system's execution. The time required for Cookie Data Encryption and Digital Signature Creation initially increases with the number of requests, but after a certain value is obtained, it decreases despite the increased number of requests. |
| **2. An Analytical Study on Cross-Site Scripting By Mehul Singh, Prabhishek Singh, Dr. Pramod Kumar (2020)** | In this paper, the author discusses a problem that many websites face, namely XSS or Cross-Site Scripting. According to one study, over half of all websites are vulnerable to XSS attacks. When malicious scripts are injected into trusted websites, XSS occurs. According to the author, the flaws that allow this attack to succeed are remarkably widespread and occur anywhere a web application handles user input without validating or encoding it. The author has discussed two methods for | Input Encoding is the least efficient of the two methods discussed by the author in this paper. It has many limitations and is best suited for small-scale websites that are created by a single person. |

| | | |
|---|---|---|
| | preventing this. Input Encoding and Output Encoding are two of these methods. Input Encoding is appropriate for small-scale websites created by a single person. When compared to Input Encoding, Output Encoding is superior. | |
| **3. Reducing attack surface corresponding to Type 1 cross-site scripting attacks using secure development life cycle practices By Syed Nisar Bukhari, Muneer Ahmad Dar, Ummer Iqbal (2018)** | According to the author, XSS is one of the most common web-based application vulnerabilities and occurs when a web-based application uses unvalidated or un-encoded user input at intervals the output it generates. In such cases, the victim is unaware that their data is being transferred from a trusted website to a different site controlled by the malicious user. The author discusses how to reduce attack surfaces related to type 1 or "nonpersistent cross-site scripting" attacks using secure development life cycle practices and techniques in this paper. Among the measures discussed by the author are: encoding any Web response data that may contain user input or other untrusted input, Microsoft Anti-Cross Site Scripting Library (AntiXSS), etc. | The authors discuss a few methods that can be used to mitigate the discussed problem in this paper. The authors have discussed a few existing libraries that can be used, but they have not discussed any proper/efficient method of permanently preventing the attacks from occurring. |
| **4. Attacks on Web Application Caused by Cross-Site Scripting By K.Pranathi, S.Kranthi, Dr.A.Srisaila, P.Madhavilatha (2018)** | The authors of this report reviewed cross-site scripting vulnerabilities and demonstrated how they can be exploited. Local (Type 0), reflected (Type 1), and persistent vulnerabilities are the three types of vulnerabilities (Type 2). An attack scenario was displayed for each of the three types of vulnerabilities, and an additional inside and out attack scenario was displayed for a thought-up Type 2 vulnerability in a message. The thoughts examined in this exhibition are spread out and linked to the Internet as a whole. Cross-site scripting vulnerabilities are constantly being discovered on the internet, and some of the lessons learned from this activity can be used to protect our sensitive data from attackers. | In this paper, the authors have discussed various types of XSS attacks and how they can be misused. The author hasn't made a website or tried the practical implementation yet. |
| **5. Model Checking for the Defense against Cross-site Scripting Attacks By Yu Sun, Dake He (2012)** | This paper proposes an automatic modelling method for defending against cross-site scripting attacks. The website's bugs are discovered, and counterexamples are displayed using the model checker SMV, which is based on the website | In this paper, the author hasn't compared the execution time of the proposed system. This is a very essential part of any experimentation and needs to be done in the future. |

| | | |
|---|---|---|
| | behaviour model expressed in CTL. An operation's behaviour is judged if it complies with the website's requirements for legal behaviour to prevent XSS attacks from the point of operation. The automatic modelling algorithm for HTML code is proposed, and a case study of the algorithm's performance is also presented. | |
| **6. Cross-Site Scripting (XSS) Attack Detection Using Intrusion Detection System By Kunal Gupta, Rajni Ranjan Singh, Manish Dixit (2017)** | Because everyone now relies on the Internet for a variety of tasks, the opportunity for attackers to corrupt data and expose vulnerable systems has increased. The proposed work proposes a framework for a system that uses an intrusion detection system to detect Cross-Site Scripting attacks. This paper focuses on using an intrusion detection system to detect XSS attacks. An attack signature is used in this case to detect an XSS attack. To test the usefulness and effectiveness of the proposed work, a proof of concept prototype was created using SNORT IDS. In this work, snort rules are introduced to detect XSS attacks. Experiments were conducted in a real-world network environment. | A few false positives were generated during the experiments, which may need to be addressed. Additionally, the detection process's speed should be increased. |
| **7. A Google Chromium Browser Extension for Detecting XSS attack in HTML5 based Websites By Arun Prasath Sivanesan, Akshay Mathur, Ahmad Y Javaid (2018)** | This document demonstrates how to create a browser extension to protect against XSS attacks. A browser extension adds new features to the browser. Extensions can be found in the browser store, where they can be downloaded and installed. The first and most difficult element of this work is to construct a repository that contains attack vector tags and properties.<br>Second, by linking the repository generated in the first step, an extension is built. This extension's role is to work in the background, looking for unusual activity and alerting users via their web browser. HTML, CSS, JavaScript, and JSON are among the technologies used to construct this extension. | HTML 5 features and properties have introduced a slew of new security risks to the web. No support for other web browsers. |
| **8. Cookies in Cross-site scripting: Type, Utilization, Detection, Protection, and Remediation By Pushpanjali** | The authors of the preceding paper discuss XSS attacks and cookies. Cookies are described as evasive tools used to maintain a connection between a user and a website. It's a simple text format for storing web data in the form of lightweight databases, allowing for improved user experiences. | The authors of the preceding paper had a wonderful idea for preventing XSS attacks using cryptography, but they did not put it to the test by creating a prototype. As a result, the above concept has yet to be proven to be a success. |

| | | |
|---|---|---|
| **Mishra, Charu Gupta (2020)** | The authors have also described the various cookie categories/types (Secure, Third Party, First Party, Session, Zombie, etc). They go over how cookies save information about users' browsing preferences, credentials, and websites visited, as well as the type of browser and devices they use. They also discussed various methods for protecting cookies from XSS attacks using the concept of cryptography. They talked about public-key cryptography and symmetric-key cryptography. They've also developed their own way, on a flowchart, to prevent attackers from stealing cookies using Cross-Site Scripting attacks. | |
| **9. Preventing parameterized vulnerability in Web-based applications By Moses Garuba, Jiang Li (2007)** | An attacker can use these strategies to pass application parameters in a web application, according to this paper. We also point out their flaws and provide techniques for counteracting them.<br>Manipulation of data transferred between the browser and the online application has long been a simple but effective approach for attackers to make apps do things the user shouldn't be allowed to do. Malicious users can change pricing in web carts, session tokens or cookie settings, and even HTTP headers in an improperly planned and developed web application. | Unless data transmitted to the browser is cryptographically protected at the application layer, no data provided to the browser can be trusted to stay the same. The transport layer's cryptographic protection (SSL) provides no protection against attacks like parameter manipulation, in which data is corrupted before it reaches the wire.<br>Any malicious user has the ability to alter cookie content to his benefit. It's a common misperception that non-persistent cookies can't be changed, but this isn't the case; programmes like Winhex are free to use. SSL only protects the cookie while it is in transit. |
| **10. An detection algorithm for ARP man-in-the-middle attack based on data packet forwarding behavior characteristics By Ming Ren, Yanhui Tian, Siqi Kong, Dali zhou, Danping Li (2020)** | At the data link layer, the ARP protocol is used.<br>Due to the lack of an authentication mechanism in the ARP protocol, an attacker can simply impersonate a gateway or a target server to capture user data packets for nefarious reasons.<br>This study presents a technique for swiftly detecting man-in-the-middle attacks based on ARP spoofing and locking the attacker's physical position in the corporate network's LAN environment.<br>The features of packets when a man-in-the-middle attack based on ARP spoofing happens in this environment will be examined in this research, and an effective technique for detecting the attack will be proposed. | At the data link layer, the ARP protocol is used.<br>Due to the lack of an authentication mechanism in the ARP protocol, an attacker can simply impersonate a gateway or a target server to capture user data packets for nefarious reasons.<br>The time complexity and space complexity of the algorithm are both O (n). |

| | | |
|---|---|---|
| **11. Detection of Cross-Site Scripting Attacks using Dynamic Analysis and Fuzzy Inference System**<br>**By Olorunjube James Falana, Ife Olalekan Ebo, Carolyn Oreoluwa Tinubu, Andeson Ntuk, Olusesi Alaba Adejimi (2020)** | Cross-site scripting (XSS) attacks have been a major threat in web applications. This exploit exploits flaws in online applications.<br>XSS attacks, among other things, can result in session hijacking, account hijacking, DDoS attacks, worm evasion, the disclosure of sensitive information, and the loss of confidentiality. The author of the preceding paper presented a system for detecting XSS attacks. The researchers employed a hybrid system that combined dynamic analysis with fuzzy approaches. In the Fuzzy model, the Levenshtein distance is used to compare and switch strings in the system. The detection system's performance demonstrates its high accuracy in detecting vulnerabilities in web applications, providing users with a reliable and effective method of mitigating XSS attacks. | The authors of the above paper have not written anything in the future work section of their Research Paper. Even though they have done a lot of research there must be some setbacks faced by the authors, but they have not mentioned them. |
| **12. Cross-Site Scripting for Graphic Data: Vulnerabilities and Prevention**<br>**By Dmytro Zubarev, Inna Skarga - Bandurova (2019)** | According to the authors of the aforementioned study, the best techniques for dealing with XSS attacks should include using a combination of white and black lists, sorting out files with vulnerabilities, and removing harmful content using regular expressions. The authors advise using the embedded validation and screening functions of the input and output data, which have an advantage over self-written data.<br>They also advocated for the use of white lists and the specification of encoding on each web page to prevent dangerous code from being embedded. They also warned against disabling cookies such as HttpOnly, which prevents scripting languages from stealing cookies. | The strip tags() function discussed in this paper is ineffective in preventing vulnerabilities because it only removes tags that are not on the whitelist, not their contents. This strategy accomplishes its goal without overwriting the information. |
| **13. Automated Repair of Cross-Site Scripting Vulnerabilities through Unit Testing**<br>**By Mahmoud Mohammadi, Bill Chu, Heather Richter Lipford (2019)** | The authors propose a unit-testing-based approach for automatically repairing cross-site scripting vulnerabilities caused by incorrect encoder usage. This approach is simple to incorporate into existing software development practices, and it can help developers fix XSS vulnerabilities early in the development cycle when unit testing their code, without the need for security experts to be involved. The authors' findings also imply that the auto-fixing technique used in this study can solve a variety of XSS vulnerabilities in web | The authors state that one of the limitations of their approach is that they cannot do auto-repair for code restructuring, etc. This needs to be improved by them in the future. |

| | | |
|---|---|---|
| | applications by replacing faulty encoders without requiring fundamental software changes. | |
| **14. Automatic XSS Detection and Automatic Anti-anti-virus Payload Generation By Lin Li , Linfeng Wei (2019)** | The preceding article focuses on the SVM, MLP, and DQN algorithms. The SVM algorithm is used to check user parameters for malicious XSS assault parameters. The DQN algorithm is used to deform the non-deformed XSS payload in order to circumvent the rules-based WAF system and achieve an anti-virus effect. A specific WAF is used in the creation of anti-virus payloads. It is also possible to determine whether the relevant WAF is safe by identifying whether an anti-virus payload is generated. Automatic XSS detection function models and automatic payload creation anti-virus are used to create a tool that can detect XSS automatically and generate payload anti-virus automatically. | According to the paper, the RNN LSTM algorithm can be used to automatically generate XSS loading tools. However, due to the small amount of data and the variety of XSS attacks, generating a single attack payload is impossible. |
| **15. Bypassing XSS Auditor: Taking Advantage of Badly Written PHP Code By Anastasios Stasinopoulos, Christoforos Ntantogian, Christos Xenakis (2014)** | The authors of the aforementioned research focused on the faults that PHP web application developers make, particularly when it comes to managing variables, and how they unknowingly assist attackers in bypassing the XSS Auditor. They attempted to develop and provide a few samples of poorly written PHP code, as well as how an attacker would construct an attack vector for an XSS attack. They have suggested some secure coding guidelines for PHP developers to guard against the discovered threats. | In this paper, the authors have discussed 3 real-world prevention methods, but they haven't discussed which method to use when or the disadvantages of each method. |
| **16. A second-order SQL injection detection method By Chen Ping (2017)** | Second-order SQL injection is a significant security risk. It is harder to identify than first-order web applications. Injection of SQL data. The payload of a second-order SQL injection attack is derived from erroneous user input and saved in a database or file. The SQL statement submitted by the web application is evaluated by the system. Typically, a trusted constant string in the dynamically built software and untrustworthy user input, and the database management system (DBMS) is unable to cope. The trusted and untrusted parts of a SQL statement should be distinguished.<br>The second-order SQL injection defence | Because the XPATH string obtained by the updatexml() function includes a syntax problem, a database error is issued when the DBMS performs the above SQL command.<br>The attacker can deduce that the password for user admin is admin123 from the error prompt message. The implementation of a second-order SQL injection attack has been ineffective. |

| | | |
|---|---|---|
| | mechanism proposed in this paper randomises SQL keywords in trusted constant strings in Web applications, resulting in dynamically new sets of SQL keywords that can be distinguished from standard SQL keywords in attack payloads injected and stored in databases or file systems. We can detect a SQL injection attack by looking for standard keywords in the SQL statement. | |
| **17. Detection of SQL injection attacks by removing the parameter values of SQL query By Rajashree A. Katole, Swati S. Sherekar, Vilas M. Thakare (2018)** | Assailants can steal confidential information by utilising SQL injection. The detection of SQL injection attacks by eliminating parameter values from SQL queries is explored in this work, and the results are shown. The method of query processing and its applications are discussed in this work.<br>The removal of parameters is utilised as a whole to create a method, which determines the difference between the original SQL query and the modified SQL query and the updated, injected SQL query through parameters.<br>Approaches have evolved from the creation of web applications to the creation of mobile apps. Web application security makes web applications more secure and dependable, taking the parameters out of the equation and comparing the results.<br>If you combine them with the original query structure, you'll get better results. | The research does not address effective ways for detecting and preventing SQL injection attacks. To reduce the amount of time it takes to detect SQL injections, more novel and robust approaches must be created. To limit the danger of SQL injection attacks, the impact on enterprises must be understood. For accurate detection of SQL injection attacks, more research is required. |
| **18. Identification and Mitigation Tool for SQL Injection Attacks (SQLIA) By W. H. Rankothge, Mohan Randeniya, Viraj Samaranayaka (2020)** | The paper proposes a method for detecting and preventing SQL injection attacks using a single tool, allowing software testers to spot SQL injection vulnerabilities in their web applications during the testing process. The proposed method is based on parameterized queries and the validation of user input. Our findings reveal that the tool is completely accurate and efficient in identifying and mitigating SQL issues. | Extensive evaluation and testing on multiple different web applications have not been done. |
| **19. From AES-128 to AES-192 and AES-256, How to Adapt Differential Fault Analysis Attacks on Key Expansion By** | The authors of this paper apply DFA techniques initially developed for AES-128 to obtain the entire keys of AES-192 and AES256. Faults during encryption and KeyExpansion computations have been investigated as the two main types of injection localisation. The analysis of this | The DFA on KeyExpansion adaption with this methodology is more complicated than the DFA on state, and each attack must be treated as a separate case.<br>For both 192-bit and 256-bit keys, the adaption requires 16 pairs, which is double the number required for AES-128, which |

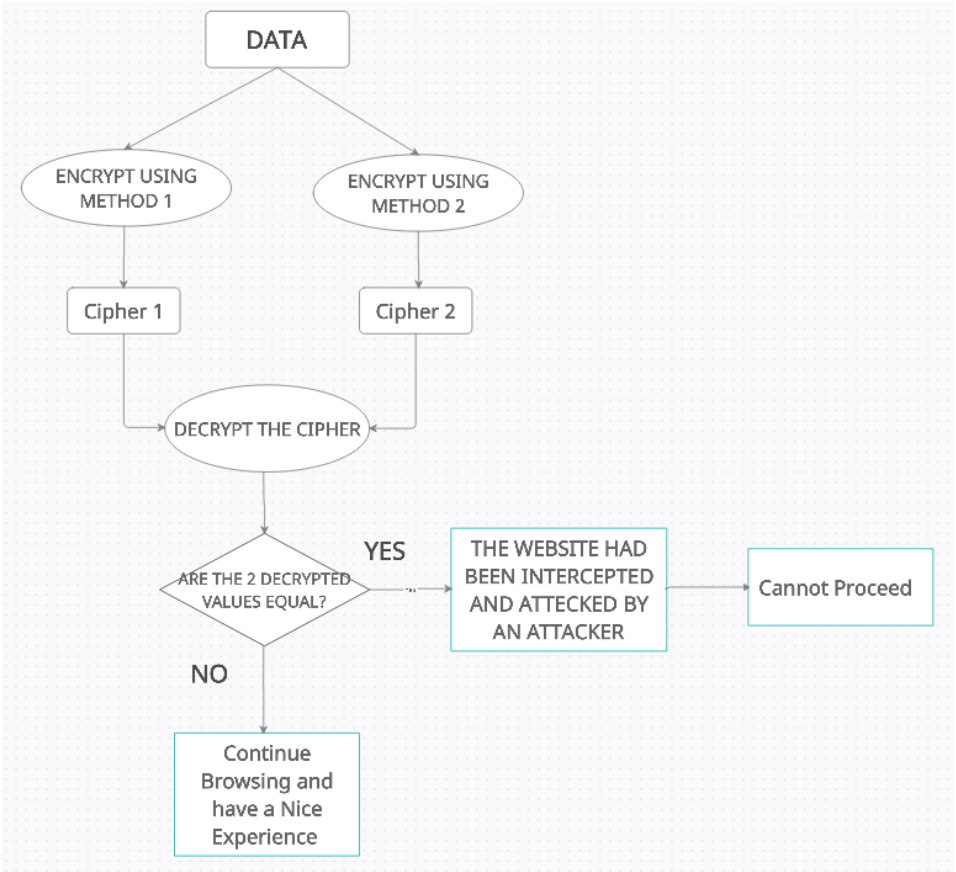
| | | |
|---|---|---|
| **Noemie Floissac, Yann L'Hyver (2011)** | last scenario reveals many fault diffusion problems that must be overcome in order to exploit differential faults.<br>Finally, on KeyExpansion, we suggest the first attack against AES192 and AES-256. In all situations, this approach leads to the discovery of the entire initial key with 16 fault injections. | is more time-consuming. |
| **20. A single, triple chaotic cryptography using chaos in the digital filter and its own comparison to DES and triple-DES By Reatrey Pich, Sorawat Chivapreecha, Jaruwit Prabnasak (2018)** | This research compares DES and triple DES to the effective performance and results of a single and triple chaotic cryptography using chaos in the digital filter. This comparison was done pair-by-pair between the single structure and the triple form. Finally, the implementation aspects of single chaotic cryptography using chaos in digital filters can stand as better performance speed with a small complexity algorithm, pointing out the similarities to DES and triple DES with similar security confirmation results without reaching the triple form of the structure. The simulation was carried out with the help of Matlab using a grayscale image as input. | In terms of complexity, both DES and 3-DES are commonly made as O(m), with 8 S-Boxes and several rounds of XOR Operations (16, 48), which is the key issue affecting the system's performance.<br>After conducting the experiment, the security validation of each technique gave a similar exclamation while producing varying complexity running performance. |
| **21. Cache-Out: Leaking Cache Memory Using Hardware Trojan By Mohammad Nasim Imtaiz Khan, Asmit De, and Swaroop Ghosh** | This paper presents Cache-out, a class of system attacks involving hardware compromised with a Trojan embedded in the CPU. A memory Trojan trigger is contained in L1 d-cache and is activated if a certain address in L1 d-cache is repeatedly pounded with a specific data pattern. Nanometer cache memories are mainly exploited e.g. WLUD, NBL and static RAM for preventing reading and writing. This paper also tells how NBL with column multiplexing can be used for stealing data. Cache-out is validated using GEM5 architectural simulator. | Detection of this attack is easy for people who have vast knowledge as after triggering the trojan, there is some delay and difference in power consumption. This type of attack can be misused easily as only the circuit and some knowledge of hardware is required. Person misusing this attack is also not easily traceable. This attack is most active if connected to the L1 level of cache. But it cannot be connected to L1 cache as data is not long kept in L1 level of cache. This attack is best suited for a victim program causing controversy. |
| **22. Hardware Trojan Detection using FBHT in FPGAs By Sundus Qayyum, Kashif Naseer Qureshi, Faisal Bashir, Najam Ul Islam, Nazir Malik** | This paper proposes the idea of Frequency-based Hardware Trojan (FBHT) in FGPAs. Addition of a circuit inside chip may perform DOS, data leakage, and change the functionality of a device. Here HT is intentionally inserted into the chip and some detection methods are applied to observe its behaviour. The 2 approaches used for detection are side channel analysis and any kind of testing (logic, real time & functional). The author has used a side channel analysis approach using path delay measurements for | The method proposed is covering only 2 types of HT, making it limited. This paper does not contain the reserve pins and the Side Channel Analysis (SCA) of the board. The source of the correct output is not given properly or proven. |

| | | |
|---|---|---|
| | the detection of hardware Trojan. | |
| **23. Hardware Trojan Detection Using Changepoint-Based Anomaly Detection Techniques By Rana Elnaggar, Krishnendu Chakrabarty, Mehdi B. Tahoori** | This paper talks about Diverse parts of semiconductor design and manufacturing flow to various parties all around the world. Outsourcing raises the possibility of attackers introducing harmful logic, known as hardware Trojans, into the original design. This method employs a changepoint-based anomaly detection algorithm to detect Trojans that introduce aberrant patterns in data streams acquired from performance counters. Detection of this method is done by activation of Trojans that cause DOS, degradation of system performance and change in functionality of a microprocessor core. | This whole model is dependent on the assumption that the DCM should be trusted. Otherwise this model fails. Cannot detect Trojan inside the L2 layer of multi level cache of CPU. This method does not detect those Trojans which cause read, write disturbances. |
| **24. Hardware Trojan Detection by Stimulating Transitions in Rare Nets By Tapobrata Dhar, Surajit Kumar Roy and Chandan Giri** | Using tri-state buffers and scan registers, this study provides an effective method for increasing the likelihood of transitions. In benchmark circuits, simulation findings demonstrate an increase in the number of net transitions. Hardware Trojans are placed into IC networks with a minimal number of signal transitions, allowing them to remain undetected. Increase the transition probability (TP) of the signals within the IC using Design for Security (DFS) ways of adding extra hardware components, which can disclose the presence of any malicious circuit by parametric analysis or direct activation of the payload. Dummy scan flip flops are used to identify Hardware Trojans by raising the TP in the circuits to a certain level. To get around this neutralisation strategy, this article proposes a technique in which tri-state buffers supply weighted signals to various sections of the IC, increasing the TP in the nets to the greatest possible value and thereby detecting the existence of Hardware Trojans with minimal DFT insertions. | The addition of tri-state buffers to the circuit raises the mean value and lowers the variation in transition probability across the board. These nets should be regarded as a vital component of the resulting circuit and should be closely monitored since they are subject to compromise by an adversary. Even after the addition of a tri-state buffer, the benchmark circuits reveal a number of nets with low transition probability. More research and development are needed for these critical networks. |
| **25. A Hardware-Trojans Detection Approach Based on eXtreme Gradient Boosting By Jinghui Chen, Chen Dong, Fan Zhang, Guorong He** | This paper proposes a machine learning based HT detection based method at gate-level. First, by the analysis of the Trojan circuits new Trojan-net features are put forward. After that, the scoring mechanism of the e eXtreme Gradient Boosting (XGBoost) is used to set up a new effective feature set of 49 out of 56 features. Finally, The effective feature set was used to train the hardware-Trojan classifier. | This method focuses on IC security. Hence, it cannot be used for L1/L2 cache levels of hardware trojans. The dataset extracted is not very vast and not dynamic. This model uses eXtreme Gradient Boosting which has the following disadvantages. Gradient Boosting Models will continue to improve in order to reduce all mistakes. This can lead to overfitting by exaggerating outliers. Computationally |

| | | |
|---|---|---|
| | | costly - typically necessitates a large number of trees (>1000), which can be time and memory consuming. Because of the approach's tremendous flexibility, there are numerous parameters that interact and strongly impact its behaviour (number of iterations, tree depth, regularisation parameters, etc.). During tuning, this necessitates a massive grid search. Less interpretive in nature, however this may be easily remedied with a variety of techniques. |

## Overall Architecture

1. Diagrammatic representation of the architecture




2. Detailed explanation of the flow of the architecture.

Encrypt the information 2 times, separately, using a different encryption algorithm and a different key each time. Now we have 2 cipher texts. Send the cipher texts to the new page after clicking the submit button. The attacker might intercept the data and manipulate the information being sent. When the next page loads the cipher text is converted back to plain text using the 2 different encryption methods. Because the data is encrypted by 2 different

algorithms, when decrypted using the key, their values might not turn out to be the same if the data was tampered. In such a case we can assume that the data had been tampered with by the attacker and therefore stop the attacker from accessing the website any further, since the 2 plain texts retrieved were different.

3. Detailed explanation of each and every block in the architecture.

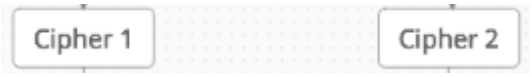
**i. Plain Text**


This is the data or the plaintext that needs to be encrypted to prevent the attackers from accessing it. This data can be anything that is important or essential to the user or the website owner.

**ii. Encryption Algorithms**


These are the 2 encryption algorithms that are used to encrypt the data and store it in 2 different variables. In this step we convert the plain text to cipher text.

**iii. Cipher text**

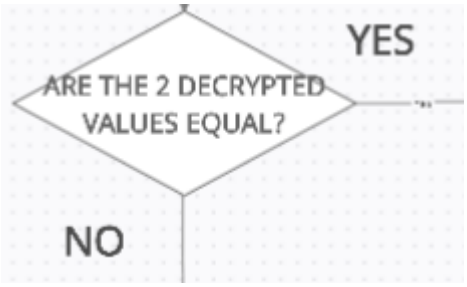



This is the cipher text, or the encrypted plain text.

**iv. Decryption**


This function in the next page of the website, decrypts the encrypted cookies or data back to the plain text for further processing of the data.

**v. Check Function**




This function is used to check whether the 2 decrypted plain text values are equal or not. If

the values are equal we can assume that the website had not been attacked by any adversary.

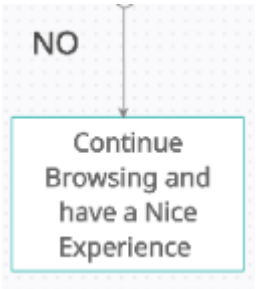
If the 2 values are not equal we can assume that the website had been intercepted in between by an adversary and they had tried to change the encrypted values.

Because we are using 2 different encryption algorithms to encrypt the plain text with different and random hashes, we are assuming that the adversary will not have the sufficient amount of time to change the encrypted values in such a way that the new values are same when they are decrypted

**vi. If the website was intercepted and the data was changed**


If the adversary was caught, he would not be able to proceed further, and would have to logout.

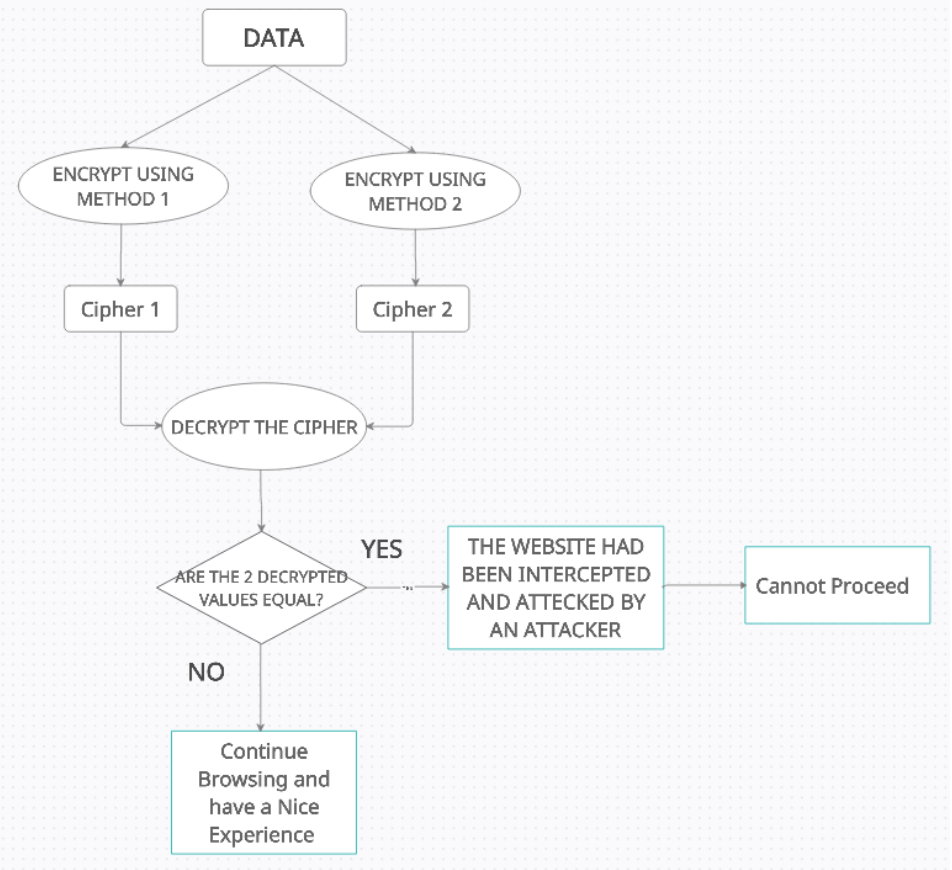
**vii. If nothing was changed**




If no tampering is detected the user is allowed to continue browsing.

## Proposed Methodology

1. Explanation of the method used with diagrammatic representation




\Encrypt the data two times, each time using a different encryption algorithm and a new key. We now have two cypher texts. After hitting the submit button, the cypher texts will be sent to a new page. The data could be intercepted and the information sent manipulated by the attacker. The cypher text is transformed back to plain text using the two separate encryption methods when the next page loads. Because the data is encrypted using two separate algorithms, its values may not be the same when decoded using the key if the data has been tampered with. We can assume that the data has been tampered with by the attacker in this situation, and we can prevent the attacker from accessing the website further, as the two plain texts received are different.

2. Advantages and disadvantages of the methods used

Advantages of AES:

It is the most secure security protocol because it is implemented in both hardware and software. It encrypts data with longer key sizes, such as 128, 192, and 256 bits. As a result, the AES algorithm becomes more resistant to hacking. It is the most widely used security protocol for a wide range of applications, including wireless communication, financial transactions, e-business, encrypted data storage, and so on. It takes roughly 2128 attempts to crack 128 bit.

As a result, it is extremely difficult to hack, making it a very safe protocol.

Disadvantages of AES:
It employs an overly simplistic algebraic structure. Every block is encrypted in exactly the same way. Software implementation is difficult. When it comes to performance and security, AES in counter mode is difficult to implement in software.

Advantages of TDES:
In both hardware and software, 3DES is simple to implement (and accelerate). It is widely used: it is supported by most systems, libraries, and protocols. It is thought to be secure up to "2112" security.

Disadvantages of TDES:
Because it was built for hardware implementations, it is slow in software.
It is applied three times on the same data, resulting in significant time waste.

3. Justification on why you have chosen this particular method

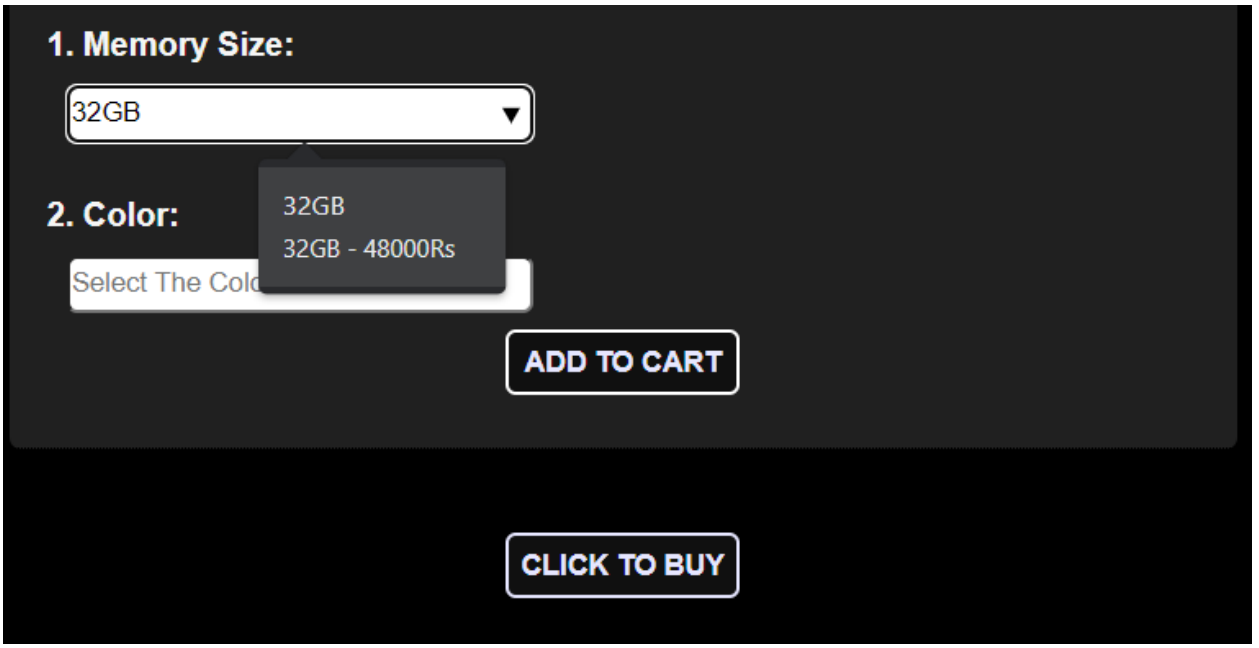
We have chosen this particular method for tackling this problem because the attackers would have no would not be able to crack both the TDES cipher text and the AES cipher text and match the changed values correctly. They would face some problem in changing the cipher values in such a way that when the values are decrypted they turn out to be the same. On top of that we have kept the encryptionIV and encryptionKey random for each session, so every time the attacker logs in, he/she would have to perform different sets of operations in order to bypass the controls.

# Results

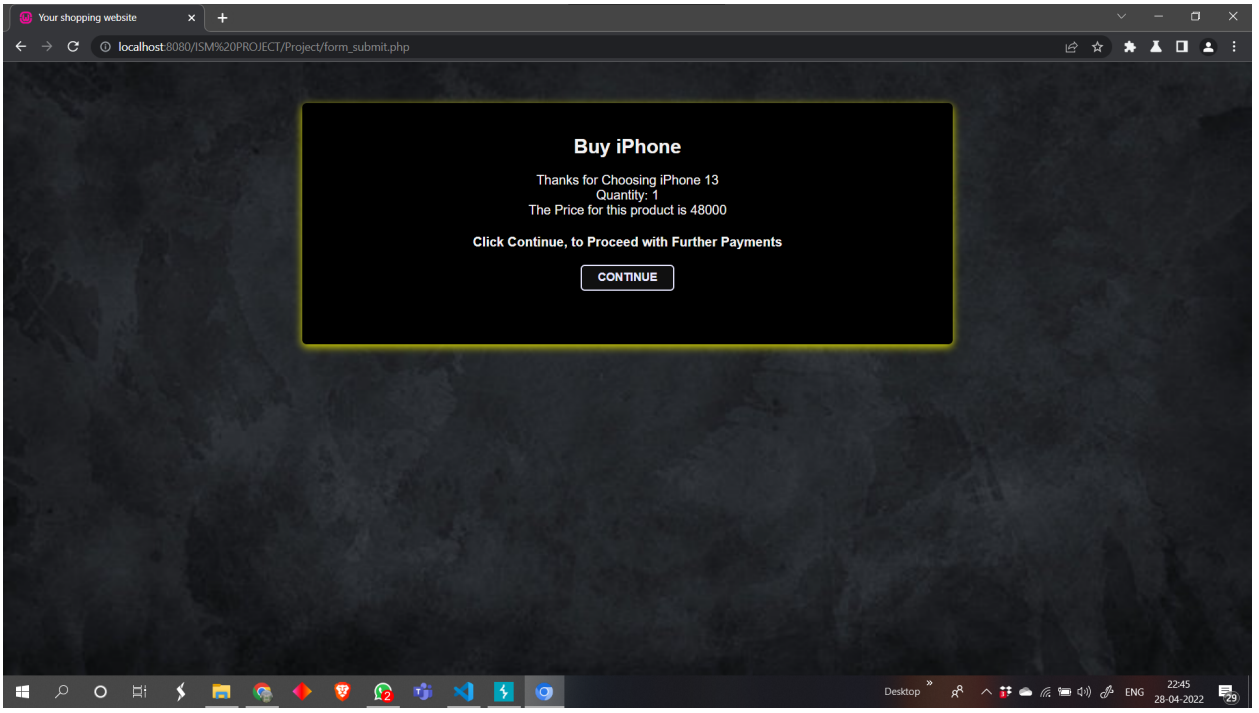
## 1. Results obtained

### a. Test Case Without Tampering the Cookies

Selected Price:




On clicking “CLICK TO BUY” (Not Tempering the cookies):




### b. Test Case 1

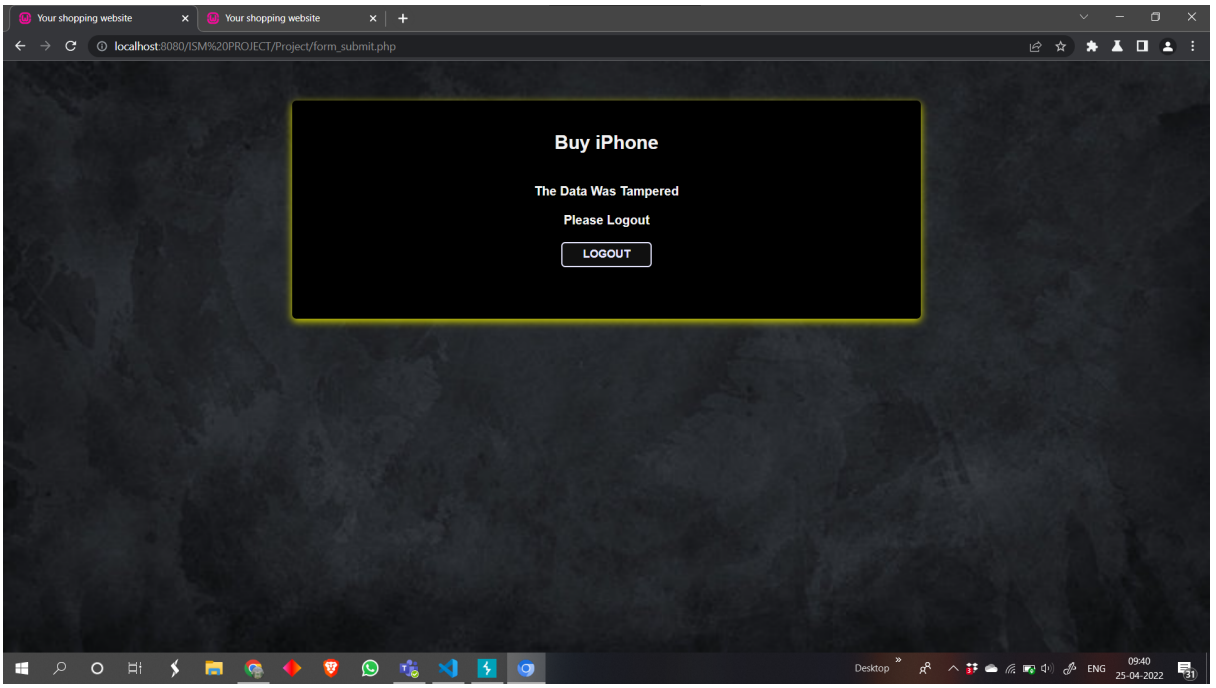
Old Encrypted Values:

```
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: temp_price=NTYwMDA%3D; product_price=RapP644%3D; PHPSESSID=2899dl4g8v6em5fbd9oea7iqmp
Connection: close

submit=CLICK+TO+BUY
```

Changed Encrypted Values:

```
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: temp_price=NTYwCDA%3D; product_price=FapP644%3D; PHPSESSID=2899dl4g8v6em5fbd9oea7iqmp
Connection: close

submit=CLICK+TO+BUY
```

Result:




**c. Test Case 2**

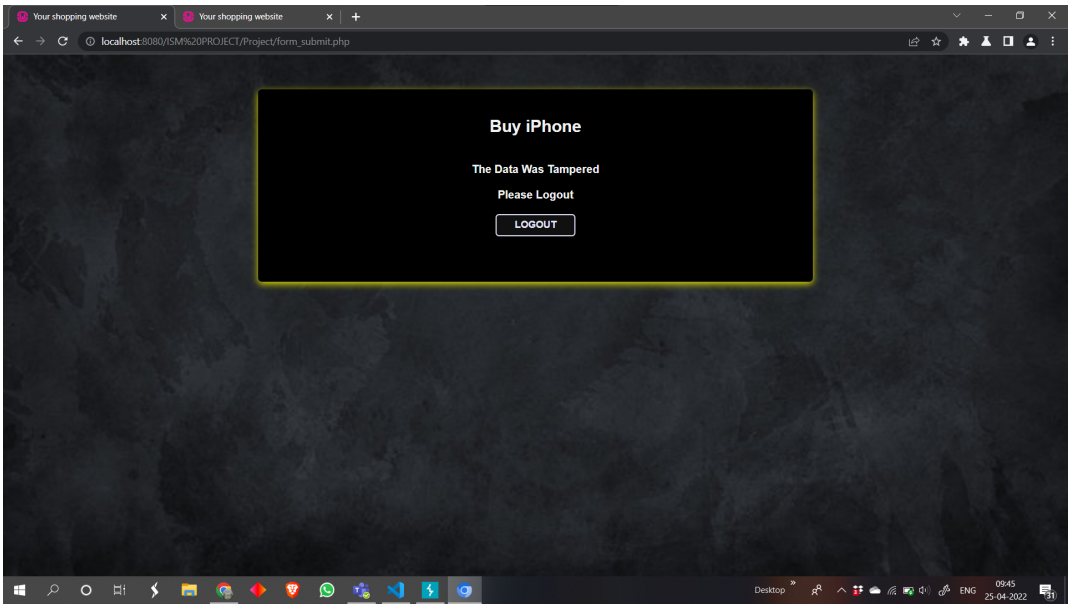
Old Encrypted Values:

```
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: temp_price=NTYwMDA%3D; product_price=XBEh9nw%3D; PHPSESSID=2899dl4g8v6em5fbd9oea7iqmp
Connection: close

submit=CLICK+TO+BUY
```

Changed Encrypted Values:

```
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: temp_price=NTYwbjkdb8%3D; product_price=hGBa9nw%3D; PHPSESSID=2899dl4g8v6em5fbd9oea7iqmp
Connection: close

submit=CLICK+TO+BUY
```

Result:




## d. Test Case 3

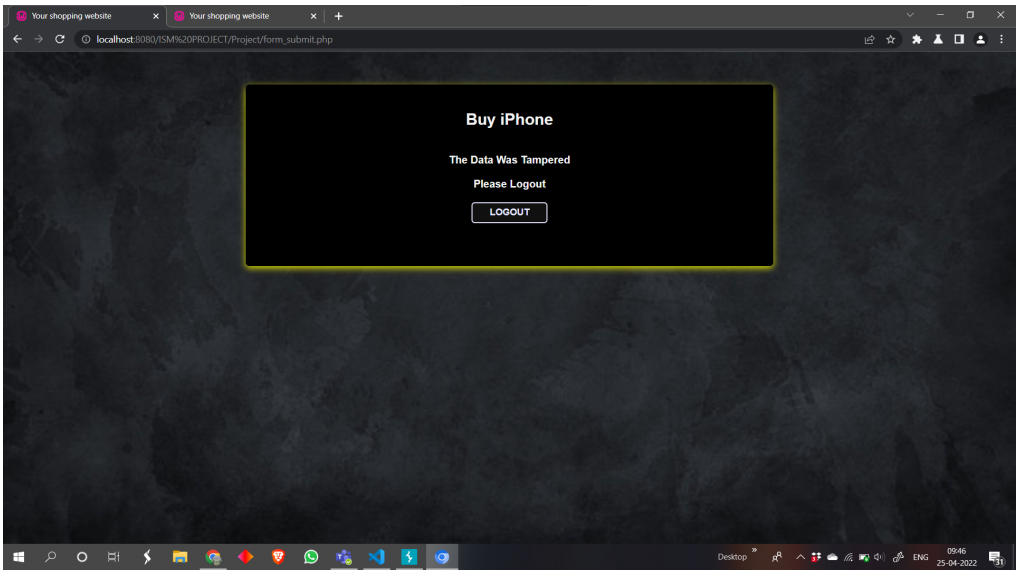
Old Encrypted Values:

```
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: temp_price=0DQwMDA%3D; product_price=KylS2bw%3D; PHPSESSID=2899dl4g8v6em5fbd9oea7iqmp
Connection: close

submit=CLICK+TO+BUY
```

Changed Encrypted Values:

```
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: temp_price=QFe2MDA%3D; product_price=Kygehjwbw%3D; PHPSESSID=2899dl4g8v6em5fbd9oea7iqmp
Connection: close

submit=CLICK+TO+BUY
```

Result:



### e. Test Case 4

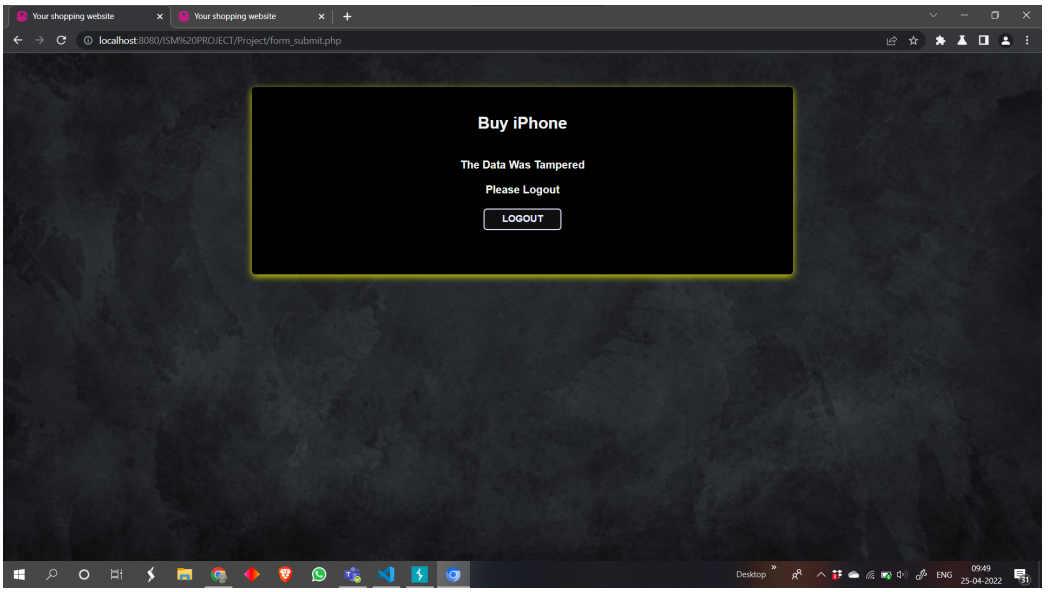
Old Encrypted Values:

```
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: temp_price=NDgwMDA%3D; product_price=D3KpCdU%3D; PHPSESSID=2899dl4g8v6em5fbd9oea7iqmp
21 Connection: close
22
23 submit=CLICK+TO+BUY
```

Changed Encrypted Values:

```
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: temp_price=NDDA%3D; product_price=D3KpCdU%3D; PHPSESSID=2899dl4g8v6em5fbd9oea7iqmp
21 Connection: close
22
23 submit=CLICK+TO+BUY
```

Result:




### f. Test Case 5

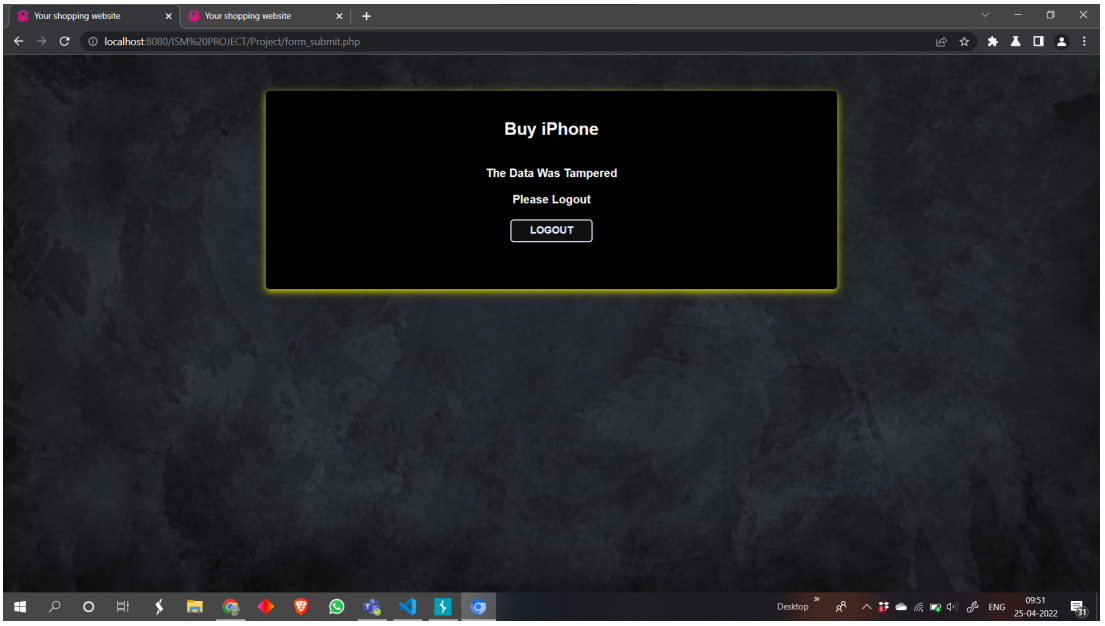
Old Encrypted Values:

```
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: temp_price=NzAwMDA%3D; product_price=ikruD74%3D; PHPSESSID=2899dl4g8v6em5fbd9oea7iqmp
21 Connection: close
22
23 submit=CLICK+TO+BUY
```

Changed Encrypted Values:

```
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: temp_price=NzAwMDA%3D; product_price=ikruD74p; PHPSESSID=2899dl4g8v6em5fbd9oea7iqmp
21 Connection: close
22
23 submit=CLICK+TO+BUY
```

Result:




## g. Test Case 6

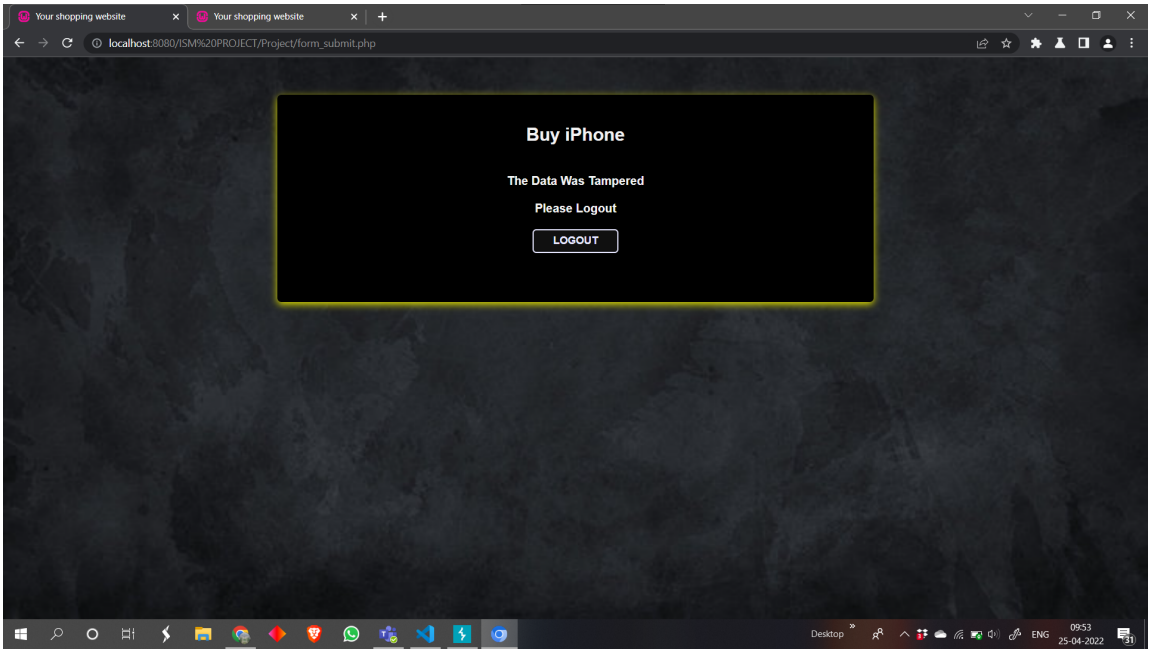
Old Encrypted Values:

```
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: temp_price=NTYwMDA%3D; product_price=AE7w43Y%3D; PHPSESSID=2899dl4g8v6em5fbd9oea7iqmp
21 Connection: close
22
23 submit=CLICK+TO+BUY
```

Changed Encrypted Values:

```
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: temp_price=NTYA%3D; product_price=AE7w43Y%3D; PHPSESSID=2899dl4g8v6em5fbd9oea7iqmp
21 Connection: close
22
23 submit=CLICK+TO+BUY
```

Result:



## h. Test Case 7

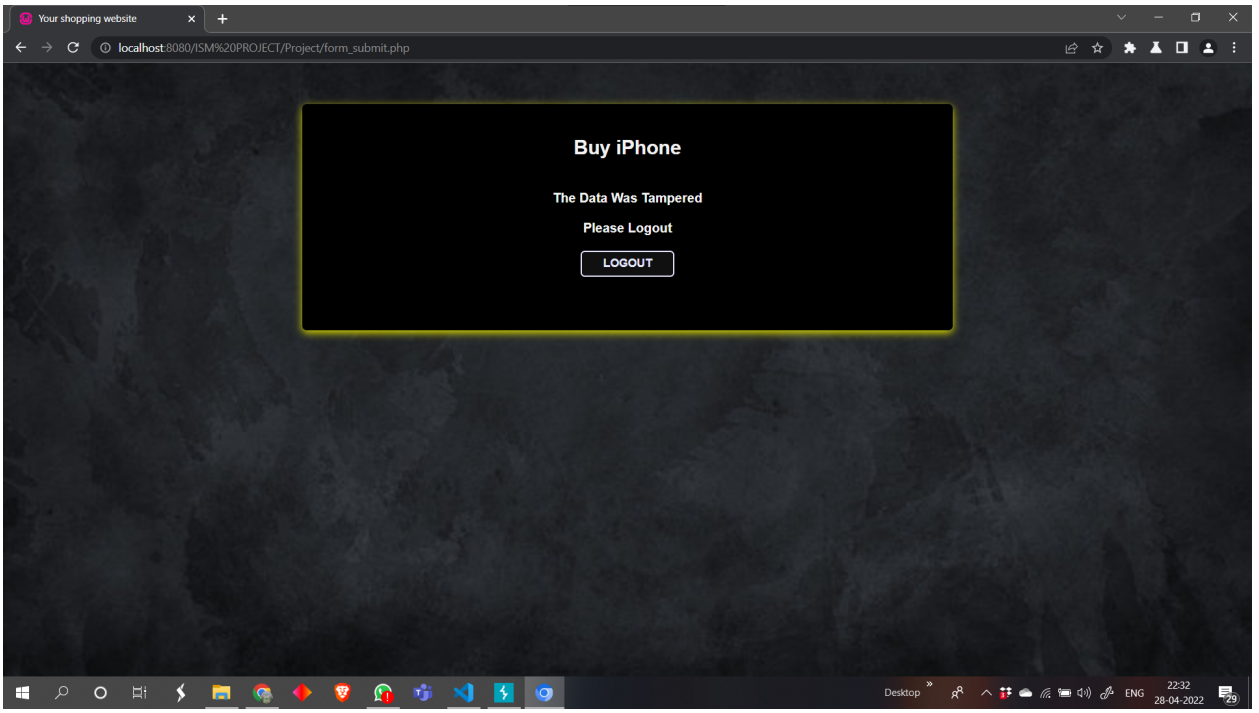
Old Encrypted Values:

```
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: temp_price=NzAwMDA%3D; product_price=j8ZEu5o%3D; PHPSESSID=umikeurcro76f2mbuohdjkeb21
Connection: close

submit=CLICK+TO+BUY
```

Changed Encrypted Values:

```
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: temp_price=#bkdkjbkD; product_price=$bhcb%w%3D; PHPSESSID=umikeurcro76f2mbuohdjkeb21
Connection: close

submit=CLICK+TO+BUY
```

Result:




## i. Test Case 8

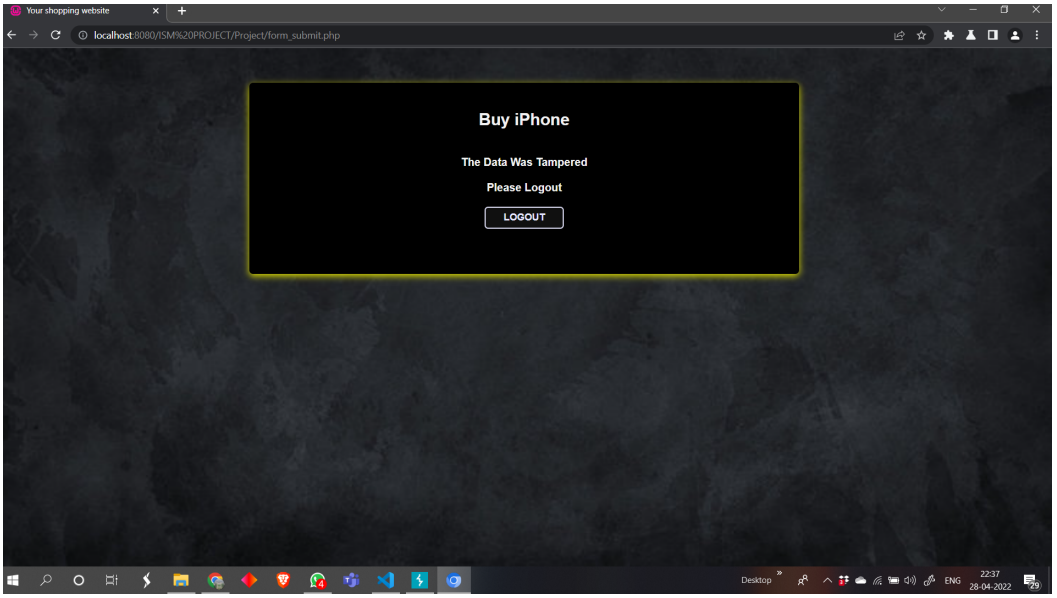
Old Encrypted Values:

```
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: temp_price=0DQwMDA%3D; product_price=kG%2FjaXA%3D; PHPSESSID=umikeurcro76f2mbuohdjkeb21
21 Connection: close
22
23 submit=CLICK+TO+BUY
```

Changed Encrypted Values:

```
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: temp_price=0DhfhkhhhD; product_price=kG&hjs3D; PHPSESSID=umikeurcro76f2mbuohdjkeb21
21 Connection: close
22
23 submit=CLICK+TO+BUY
```

Result:




## j. Test Case 9

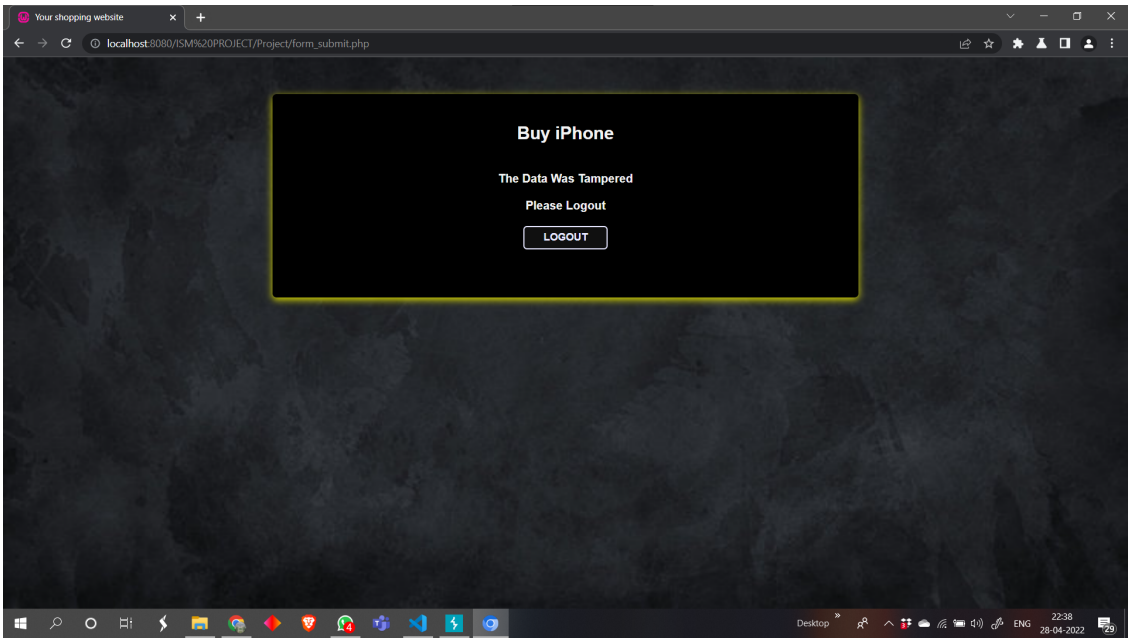
Old Encrypted Values:

```
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: temp_price=0DQwMDA%3D; product_price=kvtBCgc%3D; PHPSESSID=umikeurcro76f2mbuohdjkeb21
Connection: close

submit=CLICK+TO+BUY
```

Changed Encrypted Values:

```
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: temp_price=0DQwMDA%3D; product_price=bcvt$c%3D; PHPSESSID=umikeurcro76f2mbuohdjkeb21
Connection: close

submit=CLICK+TO+BUY
```

Result:



### k. Test Case 10

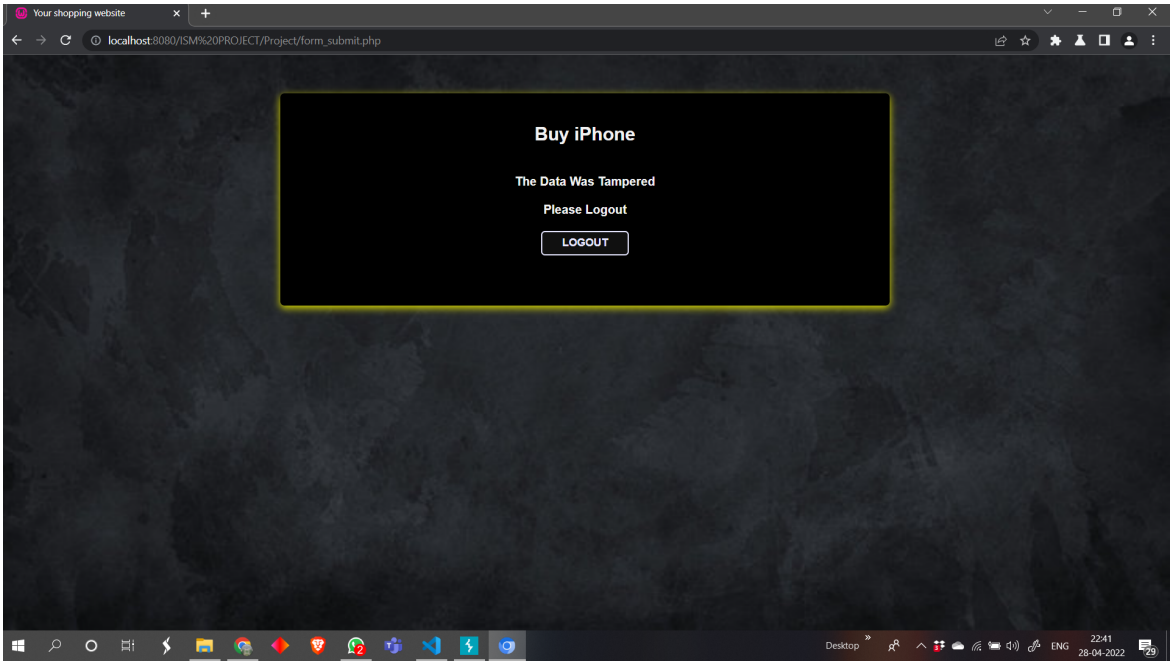
Old Encrypted Values:

```
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: temp_price=NzAwMDA%3D; product_price=4wjdhkhdfdD; PHPSESSID=umikeurcro76f2mbuohdjkeb21
21 Connection: close
22
23 submit=CLICK+TO+BUY
```

Changed Encrypted Values:

```
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: temp_price=Nzhgfhjdgh%3D; product_price=4wjdhkhdfdD; PHPSESSID=umikeurcro76f2mbuohdjkeb21
21 Connection: close
22
23 submit=CLICK+TO+BUY
```

Result:




**2. Interpretation of the obtained results.**

From the above results obtained we can conclude by saying that our proposed system has worked properly in each and every test performed till now. In the future we would have to perform more rigorous testing inorder to call this method fully efficient.

## Analysis

1. Analysis of the results obtained.

We tested the aforesaid website 50 to 100 times and got the same results each time, i.e., when we tampered with the data, we were prompted to log out, and when we continued without changing any values, the website worked regularly.

From these many trials we can conclude that this proposed system is efficient enough and for now it is not so easy for the attackers to bypass this.

2. Comparison of the obtained results with the already existing results.

We have not discovered any previous work of a similar nature, so we are unable to compare it to any existing works. Although we can presume that if a website has an SSL certificate, it is inherently secure and does not require this type of attack prevention approach. This proposed methodology is primarily aimed at safeguarding small business websites that cannot afford a full-fledged protected website.

3. Efficiency obtained should be included along with the metrics used.

Currently we have performed around 100 tests on this website and we haven't found any false displays or warnings. Till now the proposed methodology has worked properly all this while. We still have to perform some vigorous testing on this website inorder to get better results. The time taken to execute the above prevention algorithms is equal to the time complexity of the TDES encryption algorithm. This is because we have used 2 encryption algos out of which TDES takes the most of the amount to run.

## Conclusion and Future Work

1. Significance of the project

Cookie manipulation prevention can be used in transaction-based websites.
Ex: E-commerce websites of small businesses. Hashing can be used to secure login information details of the user from attackers, and the information is also not visible to the database admin ensuring total security. It provides a secure and cheap alternative to securing a website's important information.

2. Results and Efficiency obtained

We have performed tests on the created websites over a hundred times and till now we were not able to bypass the controls of the proposed methodology. Hence we can conclude by saying that our proposed idea is working efficiently.

3. Difficulties faced in performing this projects

One of the difficulties faced was the estimation of the level of security. There is a need for security experts to evaluate the level of security provided by our project and estimate its accuracy.

4. What could be done in the future?

In the Future, we would need the help of a few security experts to check the functionality of the project properly and to try and bypass the proposed methodology. Since we lacked the knowledge on cracking encrypted cypher texts we found it difficult to test the proposed methodology properly. With the help of a security expert we would be able to overcome this problem and also understand how efficient our system really is.

## References


[1] Prapty, R. T., Md, S. A., Hossain, S., & Narman, H. S. (2020, April). Preventing session hijacking using encrypted one-time-cookies. In *2020 Wireless Telecommunications Symposium (WTS)* (pp. 1-6). IEEE

[2] Singh, M., Singh, P., & Kumar, P. (2020, March). An Analytical Study on Cross-Site Scripting. In *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)* (pp. 1-6). IEEE.

[3] Bukhari, S. N., Dar, M. A., & Iqbal, U. (2018, February). Reducing attack surface corresponding to Type 1 cross-site scripting attacks using secure development life cycle practices. In *2018 fourth international conference on advances in electrical, electronics, information, communication and bio-informatics (AEEICB)* (pp. 1-4). IEEE.

[4] Pranathi, K., Kranthi, S., Srisaila, A., & Madhavilatha, P. (2018, March). Attacks on web application caused by cross site scripting. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)* (pp. 1754-1759). IEEE.

[5] Sun, Y., & He, D. (2012, August). Model checking for the defense against cross-site scripting attacks. In *2012 International Conference on Computer Science and Service System* (pp. 2161-2164). IEEE.

[6] Gupta, K., Singh, R. R., & Dixit, M. (2017, June). Cross site scripting (XSS) attack detection using intrustion detection system. In *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)* (pp. 199-203). IEEE.

[7] Sivanesan, A. P., Mathur, A., & Javaid, A. Y. (2018, May). A google chromium browser extension for detecting XSS attack in HTML5 based websites. In *2018 IEEE International Conference on Electro/Information Technology (EIT)* (pp. 0302-0304). IEEE.

[8] Mishra, P., & Gupta, C. (2020, June). Cookies in a Cross-site scripting: Type, Utilization, Detection, Protection and Remediation. In *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)* (pp. 1056-1059). IEEE.

[9] Ren, M., Tian, Y., Kong, S., Zhou, D., & Li, D. (2020, June). An detection algorithm for ARP man-in-the-middle attack based on data packet forwarding behavior characteristics. In *2020 IEEE*

*5th Information Technology and Mechatronics Engineering Conference (ITOEC)* (pp. 1599-1604). IEEE.

[10] Garuba, M., & Li, J. (2007, April). Preventing parameterized vulnerability in Web based applications. In *Fourth International Conference on Information Technology (ITNG'07)* (pp. 790-793). IEEE.

[11] Falana, O. J., Ebo, I. O., Tinubu, C. O., Adejimi, O. A., & Ntuk, A. (2020, March). Detection of cross-site scripting attacks using dynamic analysis and fuzzy inference system. In *2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS)* (pp. 1-6). IEEE.

[12] Zubarev, D., & Skarga-Bandurova, I. (2019, June). Cross-Site Scripting for Graphic Data: Vulnerabilities and Prevention. In *2019 10th International Conference on Dependable Systems, Services and Technologies (DESSERT)* (pp. 154-160). IEEE.

[13] Mohammadi, M., Chu, B., & Lipford, H. R. (2019, October). Automated repair of cross-site scripting vulnerabilities through unit testing. In *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)* (pp. 370-377). IEEE.

[14] Li, L., & Wei, L. (2019, October). Automatic xss detection and automatic anti-anti-virus payload generation. In *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)* (pp. 71-76). IEEE.

[15] Stasinopoulos, A., Ntantogian, C., & Xenakis, C. (2014, December). Bypassing XSS Auditor: Taking advantage of badly written PHP code. In *2014 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)* (pp. 000290-000295). IEEE.

[16] Ping, C. (2017, December). A second-order SQL injection detection method. In *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)* (pp. 1792-1796). IEEE.

[17] Katole, R. A., Sherekar, S. S., & Thakare, V. M. (2018, January). Detection of SQL injection attacks by removing the parameter values of SQL query. In *2018 2nd International Conference on Inventive Systems and Control (ICISC)* (pp. 736-741). IEEE.

[18] Rankothge, W. H., Randeniya, M., & Samaranayaka, V. (2020, November). Identification and Mitigation Tool for Sql Injection Attacks (SQLIA). In *2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS)* (pp. 591-595). IEEE.

[19] Floissac, N., & L'Hyver, Y. (2011, September). From AES-128 to AES-192 and AES-256, how to adapt differential fault analysis attacks on key expansion. In *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography* (pp. 43-53). IEEE.

[20] Pich, R., Chivapreecha, S., & Prabnasak, J. (2018, January). A single, triple chaotic cryptography using chaos in digital filter and its own comparison to DES and triple DES. In *2018 International Workshop on Advanced Image Technology (IWAIT)* (pp. 1-4). IEEE.

[21] Khan, M. N. I., De, A., & Ghosh, S. (2020). Cache-out: Leaking cache memory using hardware trojan. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *28*(6), 1461-1470.

[22] Qayyum, S., Qureshi, K. N., Bashir, F., Islam, N. U., & Malik, N. (2020, January). Hardware Trojan Detection using FBHT in FPGAs. In *2020 17th International Bhurban Conference on Applied Sciences and Technology (IBCAST)* (pp. 422-427). IEEE.

[23] `Elnaggar, R., Chakrabarty, K., & Tahoori, M. B. (2019). Hardware trojan detection using changepoint-based anomaly detection techniques. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *27*(12), 2706-2719.

[24] Dhar, T., Roy, S. K., & Giri, C. (2019, January). Hardware trojan detection by stimulating transitions in rare nets. In *2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID)* (pp. 537-538). IEEE.

[25] Chen, J., Dong, C., Zhang, F., & He, G. (2019, August). A Hardware-Trojans detection approach based on eXtreme Gradient Boosting. In *2019 IEEE 2nd International Conference on Computer and Communication Engineering Technology (CCET)* (pp. 69-73). IEEE.

## Appendix

Work done by each and every individual student.

1. Student Name 1: Prithish Samanta (19BCE2261) - Worked on creating the prototype of the proposed system, and also helped with creating report of the project.
2. Student Name 2: Ritvik Kohli (19BCE2223) - Worked on creating the project report, also helped in the creation of the proposed system.