International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 2 Feb. 2023, pp: 822-833 www.ijaem.net ISSN: 2395-5252

# Survey on SQL Injection and Cross-Site Scripting Malware Injection Attacks

Suren Krishnan[1], Mohamad Fadli Zolkipli[2]
[1]*Awang Had Salleh Graduate School, School of Computing, Universiti Utara Malaysia, Kedah, Malaysia*
[2]*School of Computing,Universiti Utara Malaysia,Kedah, Malaysia.*

Date of Submission: 15-02-2023 Date of Acceptance: 25-02-2023

**ABSTRACT**: A malware injection attack is a type of cyberattack where an attacker exploits vulnerabilities in software applications to inject malicious code into a target system. This code can then execute unauthorized commands, steal sensitive data, or provide unauthorized access to the attacker. In this overview, we will overview the different types of malware injection attacks, SQL injection attack and Cross-Site scripting injection attack. We will also discuss the techniques used by attackers to initiate these attacks and exploiting unpatched vulnerabilities. Additionally, we will explore the ways in which organizations can defend against these attacks, including automated reasoning with static analysis, a prototype approach called AMNESIA, white-box and black-box vulnerability scanners, firewalls, Hybrid Approaches of Mitigation, XPath expressions and regularly patching software. Overall, this overview will provide a comprehensive understanding of malware injection attacks and the steps organizations can take to protect themselves against them.

**KEYWORDS:**Malware injection attack,SQL injection, Cross-site scripting.

## I. LITERATURE REVIEW

Malware injection attacks are a common and persistent threat in today's cyber landscape. They can cause significant damage to organizations, resulting in data theft, financial loss, and reputational damage. These attacks involve the injection of malicious code into a targeted system, which can then perform unauthorized actions, such as stealing sensitive data or giving an attacker remote control over the compromised system. As software applications become more complex, the number of potential vulnerabilities increases, providing attackers with a greater opportunity to exploit weaknesses in a system's defense. Therefore, it is critical for organizations to understand the different types of malware injection attacks, the methods used by attackers to initiate them, and the steps that can be taken to prevent them. All vibrant web applications have one thing in common where they all require the usage of a database to keep information, which can then be retrieved by the user or created, modified, or removed. Rational databases are the most common form of database. SQL is the language used by most relational database management systems, comprising Oracle, MS SQL, MySQL server, Postgres and MS Access. SQL's flexibility makes it an excellent language. It enables the individual to request information without having any understanding of how the information will be obtained. An attacker can delete information from the database using compromised availability data. An attacker can directly affect the host operating system through remote command execution [1].

Nevertheless, the widespread usage of SQL-based systems has drawn the attention of attackers. A common security risk to database-driven web applications is the SQL injection attack. When a SQL injection attack becomes successful, the hacker is given access to crucial private data. The server and client tiers are two of the conventional execution levels seen in contemporary online applications. According to Maurel, Vidal, and Rezk [2] a range of languages are needed for the writers to complete the steps, including JavaScript for the web client and Node.js and PHP for the web server. A TCP SYN flood of segment, for example, could be an IP address brush that causes a running host to respond or it could be a SYN flood attack that seeks to overwhelm the network and render it unreliable. Intelligence gathering activities may be seen as a harbinger of an imminent attack because attackers often inspect targets before attacks. In other words, it is the beginning of the onslaught. As a result, it might be difficult to distinguish between offensive and reconnaissance actions when using the phrase "exploit." Both our daily lives and our enterprises now place a high value on computer networks and the Internet. As we rely increasingly on computers

International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 2 Feb. 2023, pp: 822-833 www.ijaem.net ISSN: 2395-5252

and communication networks, malicious activity is increasing in frequency. A significant issue in the current communications environment is network attacks. You must monitor and analyse network traffic to find harmful activities and attacks if you want to make sure that your network operates dependably and that user data is secure [3].

## II. INTRODUCTION

These days, web applications such as courses, shopping, social networks, banking, online services, and email play a significant part in online business. Because of their accessibility and convenience of using it, web applications are more widely used to provide online services than in-person services. To access a web application and enjoy the online services it offers, all a basic user requires a computer and a connection to the internet.Because they provide attackers unrestricted access to the databases that power the program and the potentially sensitive data they contain, web applications are significantly at risk from SQL injection attacks[5]. The SQL injection problem has been addressed in a variety of ways by practitioners and researchers, however, current solutions either don't fully address the scope of the problem or have disadvantages that hinder their usage and acknowledgement. The huge diversity of attack techniques available to individuals aiming to exploit SQL injection problems is so broad that only a small part of them is known by several scholars and practitioners [5].

It is very necessary to create a networking environment that is both useful and secure. If a vulnerability exists on widely used websites, then many people will be targeted, which would have unimaginable consequences. Cross-site scripting, often known as XSS, is one of the vulnerabilities that is most frequently found in web applications.This overview will provide a comprehensive analysis of malware injection attacks, including the techniques used by attackers, the different types of attacks, and the various measures that can be implemented to protect against them. By examining these factors, organizations can take proactive steps to mitigate the risks of malware injection attacks and safeguard their critical assets.The goal of this paper is to detectand prevent malware injection attacks from being injected into a system or application by an attacker, and to detect and remove any such malware that may have already been injected. Malware injection attacks can be very harmful to a system, as they can allow an attacker to gain unauthorized access, steal data, or cause other types of damage.The rest of the paper is organized as follows: Section 2 describes literature review of malware injections attacks. Section 3 will discover various types of malware injection attacks.Section 4 will explore the techniques and threat models of malware injection, including how the injections occur, the impact of the attacks, and malware injection detection and prevention. Section 5 explores the challenges of malware injections and discussions. Section 6 concludes this article in the conclusion part and follows with acknowledgements and references.

## III. SQL INJECTION ATTACK

i. Introduction

SQL injection vulnerabilities have been identified as one of the main threats to Web-based systems. Applications on the web that are vulnerable to SQL injection might provide an intrusive party full access to the databases at their base. These databases typically include sensitive user or customer information, and security breaches can result in fraud, unlawful access, and the deletion of personal data. In rare circumstances, attackers can even control and through a SQL injection vulnerability, the web application's hosting system might be damaged. The prevalence of web applications that are liable to SQL Injection Attacks (SQLIAs) has been demonstrated by a Gartner Group analysis of over 300 Internet websites, showing that the most of them could be at risk. High-profile victims including FTD.com, Guess Inc. and Travelocity have been successfully targeted by SQLIAs. SQL injection is a type of code-injection attack in which user information is inserted into a SQL query in such a manner that a part of the user's input is interpreted as SQL code [6].

ii. Techniques

Many other methods have been suggested by researchers to deal with the SQL injection issue. These methods cover everything from fully autonomous frameworks for identifying and avoiding SQLIAs to development industry standards. The benefits and drawbacks of each strategy are listed below after we analyze the techniques that have been offered. Inadequate validation is the main factor contributing to SQL injection flaws. Accordingly, employing appropriate defensive coding standards is the simple fix for closing these loopholes. In this article, we offer a summary of some of the top recommendations made for avoiding SQL injection vulnerabilities. Depending on the strategy, attackers may use one of three fundamental techniques to carry out SQL injections.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 2 Feb. 2023, pp: 822-833 www.ijaem.net ISSN: 2395-5252

There are three distinct varieties of SQLi, which are referred to as In-band SQLi, Inferential SQLi, and Out-of-band SQLi. SQL injections are used to gain access to databases and evaluate the severity of any possible harm. The simplest kind of SQL injection attack happens when a hacker utilizes the same interaction channel to issue and receive the results of a logical SQL command in-band. This is the most common form of SQL injection attack. SQL injection based on faults and SQL injection based on unions are the two most common methods for performing SQL injection in-band. Both techniques are also known as fault injection and union injection. SQLI Using the Blind Inferential Operator Because neither the data nor the outcome of the send are disclosed by the web service during an inferential SQLI attack, the perpetrator of the assault cannot determine how it will turn out. Injecting payloads into a Deductive Reasoning SQL injection and watching the responses from both the web application and the database server allows an attacker to rapidly alter the database schema [7].

Verifying input type Injecting instructions into a text or numerical argument can be used to conduct SQLIAs. One possibility is to outlaw the usage of these meta-characters doing so would make it more difficult for non-malicious users to provide legitimate inputs including these characters. Utilizing routines that encrypt strings in a way that the database interprets all meta-characters as regular characters is a preferable method. A positive pattern matching is where the data verification procedures that distinguish between good and incorrect input should be established by developers. Positive validating is the generic term for this method as compared to negative validating, which scans insight for illegal sequences or SQL tokens. Positive validation is a safer technique to check inputs because developers might not be able to foresee every sort of attack that could be launched against their application but should be able to describe all the permissible input formats. In terms of identification of all input sources, all entry to the program must be verified by the developers[8].

There are several potential entry streams for applications. These input resources may be used to build a query, which might allow an attacker to include a SQLIA. Briefly stated, every input source has to be looked out. Though defensive coding techniques remain the finest approach to avoid SQL injection problems, their practical implementation is challenging. In contrast to automatic methods, defensive coding is less often utilized and more vulnerable to human error. Even if most developers do make an attempt to write safe code, it is very challenging to follow defensive coding best practices consistently across all input types. Many SQL injection vulnerabilities found in real systems are caused by human error: developers neglected to include checks or did not do sufficient input validation [9].To put it another way, developers attempted to identify and prevent SQLIAs in these programs, but they did not succeed in doing so sufficiently or at all necessary sites. These instances offer more proof of the drawbacks of relying on developers' protective code[10].

Furthermore, the widespread marketing and acceptance of so-called "pseudo cures" undermine defensive coding-based strategies. We go over two of the most popular hoaxes. The first of these fixes is scanning user input for SQL operators like the solitary quote or comment controller as well as SQL keywords like “WHERE”, "FROM," and "SELECT"[8]. This idea's justification is that the existence of such terms and processors can point to a failed SQLIA attempt. Given that SQL operators may be used to express formulae or even names like O'Brian and that many programs allow for the inclusion of SQL keywords as part of regular text entries, this method inevitably leads to a significant percentage of false positives. The second often advocated "fix" is to employ prepared statements or stored procedures to stop SQLIAs. Sadly, unless developers strictly follow defensive coding rules, functions and set by organizations can likewise be susceptible to SQLIAs.

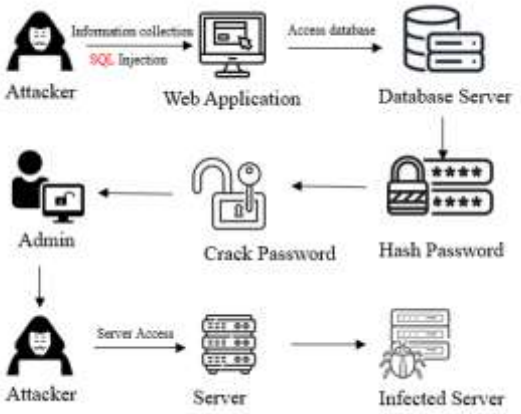


**Figure 1: The SQL Injection Attacks**

iii. Detection

Researchers have proposed a range of ways to assist developers and overcome the shortcomings of defensive coding. First, testing of black box is where in order to check Web

International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 2 Feb. 2023, pp: 822-833 www.ijaem.net ISSN: 2395-5252

applications for SQL injection susceptibilities, Huang and colleagues suggest WAVES, a black-box method. To locate all prospective SQLIA injection points in an online application, the approach makes use of a web crawler. It then creates attacks that particularly aim for those places utilizing a pre-set set of attacking tactics and methods[10]. The application's response to the attacks is then monitored by WAVES, which utilizes machine learning to improve its attacking plan. By directing its testing with machine learning techniques, the specific technology surpasses conventional penetration-testing methods. Unfortunately, it cannot ensure completeness like all other black-box and exploitation test methods[11]. In terms of Static code checkers, a method for statically evaluating the kind of accuracy of SQL queries created dynamically is JDBC-Checker[12]. This method was not designed to identify and stop conventional SQLIAs, but it may still be used to stop attacks that rely on type mismatches in a constantly generated request message. One of the main sources of SQLIA vulnerabilities in programming is poor type checking of input, which JDBC Checker is able to identify. Yet, because the majority of these exploits comprise of models and type-correct queries, this method would miss more widespread SQLIAs.The existing SQL detection algorithm is built on grammatical and structural patterns for legitimate and erroneous query requests. SQL detection is precise and false-positive whenever a legitimate query request includes a phrase that the semantic tree conceptual model considers suspicious[13].

iv. Prevention

To ensure that the SQL queries created in the application level contain no tautology, Wassermann and Su present a method that combines automated reasoning with static analysis [14]. This method's main flaw is that it can only identify and avoid tautologies; it cannot identify other forms of attack. In terms of combined static and dynamic analysis, A prototype approach called AMNESIA combines real time monitoring and static analysis [15]. AMNESIA's static phase employs analysis method to create models of the various query types that applications are permitted to produce at every stage of database access. AMNESIA's dynamic phase involves monitoring all inquiries before they are delivered to the system and comparing them to the models that were created statically. The database identifies queries that break the model as SQLIAs and forbids them from running[6]. The developer is responsible for cleaning up all input, not only that which comes through online forms such as login forms. preventing database issues from displaying on the live version of your website [13].

## IV. CROSS-SITE SCRIPTING (XSS)

i. Introduction

XSS vulnerabilities are quite similar to SQL injection problems in a number of ways. This kind of attack takes use of an application's output function, which makes use of user input that has not been properly cleaned up.Cross-site scripting XSS vulnerabilities target the HTML output function that transmits data to the browser, in contrast to SQL injection vulnerabilities, which aim to compromise the query function that communicates with the database. Cross-site scripting is a kind of hacking that, in general, refers to a method that allows an attacker to transfer damaging information from a user and gather data from the victim by exploiting holes in the code of a web application. Cross-site scripting [16].The most fundamental component of XSS injection is the use of special characters for the purpose of transitioning web browser interpreters from a data context to a code context. 1 For example, when an HTML page references a user input as data, an attacker might include the tag <script>, which can invoke the Java- Script interpreter. If the application does not filter these special characters, a successful XSS injection gives the attacker the ability to perform attacks such as account takeover, cookie poisoning, denial of service (DoS), and manipulation of web content. A variety of input sources, including external files, cookies, URLs, and HTML forms, are often altered by attackers. JavaScript is the most common choice for attackers, although XSS may also occur with VBScript, Flash, and other client-side languages that browsers might be able to comprehend[17].

ii. Techniques

One way to manually test for stored and reflected XSS vulnerabilities is to manually insert a JavaScript snippet into each and every HTML input field. This may be done in order to manually test for vulnerabilities. The next step is to determine whether HTTP responses include the input that was supplied. Utilizing the browser developer tools may make manual testing for DOM-based XSS vulnerabilities caused by URL manipulation simpler. These vulnerabilities may arise when a URL is manipulated. On the other hand, analyzing the JavaScript code for non-URL DOM-based attacks such as the non-HTML sinks document.cookie and setTimeout may be highly challenging and time consuming [18]. XSS

International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 2 Feb. 2023, pp: 822-833 www.ijaem.net ISSN: 2395-5252

vulnerabilities may be typically divided into three categories: reflected, stored, and DOM-based. These categories are determined by how HTML websites connect to user inputs. A Web application server programme that makes use of received user input in the leaving website has reflected or nonpersistent XSS problems. These flaws might cause the input to be misused. These XSS attacks are frequently found in the results of search queries as well as error messages. The XSSed project (http://xsed.com) has discovered a large number of reflected XSS weaknesses in McAfee, which are vulnerabilities that cybercriminals might use to deceive users into downloading malware. Stored cross-site scripting vulnerabilities, also known as persistent XSS vulnerabilities, occur when a server programme saves user input, including injected code, in a permanent data storage, such as a database, and then accesses it on a website at a later time. XXS vulnerabilities are often exploited during cyberattacks directed at social networking websites. One example of this kind of situation is the Samy worm, which, on October 4, 2005, less than 24 hours after it was released, exponentially grew the friend lists of one million Myspace users, ultimately leading to a denial-of-service attack. Both reflected and stored cross-site scripting attacks are possible if server-side scripts do not correctly manage user input. However, DOM-based XSS vulnerabilities in web applications arise when client-side scripts make unchecked references to user inputs that are then dynamically pulled from the structure of the Document Object Model. A DOM-based XSS vulnerability is shown in the Bugzilla bug 272620 (https://bugzilla.mozilla.org/showbug.cgi?id=272620), which may be seen here[19]. For instance, an adversary may use phishing in conjunction with the XSS virus in order to alter the DOM structure and data on a web page. The login box for the website is imitated by the JavaScript code. After being entered by the user, the username and password are sent to the server, where they are then checked for XSS vulnerabilities [20].

Attacks using XSS may be grouped into three categories: DOM-based, persistent, or non-persistent[21].

1) XSS that is based on domes. Dome-based cross-site scripting is often referred to as type 0 XSS. It is an attack that takes place on the client side and occurs when the client-side script of a web application inserts data from the user into the document object model (DOM). After that, the web application simultaneously read data from the Document Object Model. An adversary might even include a payload within the DOM itself, which would be carried out whenever the data was read back from the DOM. Web application firewalls and security professionals who check the server log may have problems spotting this attack since the payload of the attacker is never sent to the server [22].

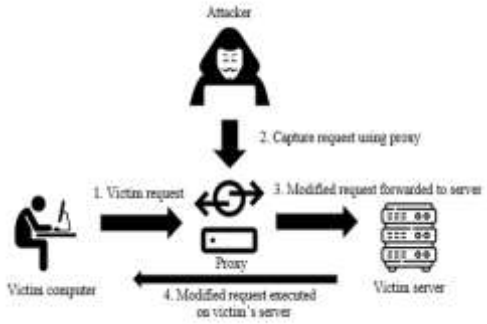


**Figure 2: DOM XSS Attack**

2) Stored XSS over an extended period of time or type 1 There are a few other names for cached XSS, including XSS. The attacker made use of a malicious script that was included into the attack and was saved on the server in a persistent location. The comments section of a blog post or forum article is one of the most common places where this sort of assault is carried out, making it one of the most common examples [22].

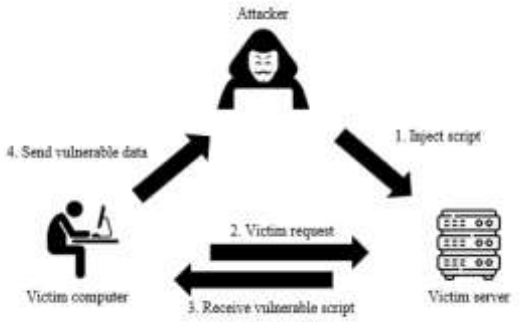


**Figure 3: Stored XSS Attack**

3) Reflected XSS non-persistent type 2 vulnerability The term "XSS" may also be used to refer to "reflected XSS." The perpetrator of this attack starts by creating a link that leads to a malicious website. After the malicious link has been crafted, the system will email the user with the URL and encourage them to follow the link. After that, the user sends a request to the server asking for access to the necessary page. A reliable server will reply to the user's request and then

International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 2 Feb. 2023, pp: 822-833 www.ijaem.net ISSN: 2395-5252

provide them with a return page that includes harmful code. When a person logs in, a potentially dangerous cross-site scripting link is launched inside their browser, giving the attacker with access to sensitive data[22].

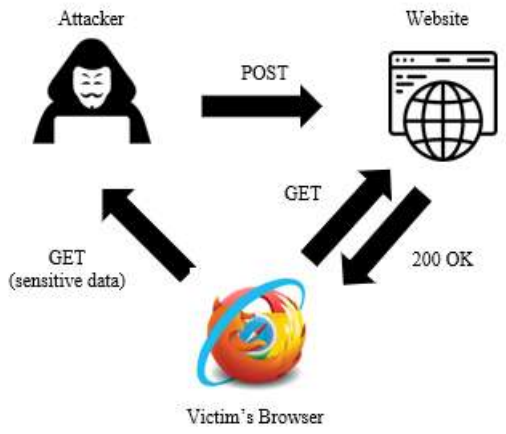


**Figure 4: Reflected or Indirect XSS attack**

iii. Detection

Filtering and a variety of other detection approaches are being used in the investigation of cross-site scripting (XSS) attacks. This section provides an explanation of the relevant work.Analysis of strings via the use of static analysis Learning static string analysis is something that is recommended for imperative programming languages by A.S. Christensen, A. Mller, and M.I. Schwartzbach. They demonstrated the usefulness of string analysis for troubleshooting the reflective code of Java programmes and checking dynamically produced SQL queries for faults [23]. In order to do an analysis of Java, they used finite state automata as a target language representation. They prefer FSAs due to the fact that regular language procedures may be used to shut them. In addition to this, computational linguistics techniques, such as the usage of finite state automata, were used in order to provide accurate CFG approximations [24]. This method is not as effective as prior string analyses in identifying XSS vulnerabilities since the origin of the input was not determined, and Finite State Automata had to be created between each operation. Consequently, this method is less efficient. When creating a string analysis for PHP, Y. Minamide used a methodology that was quite similar to this one, although he did not attempt to approximate CFGs to finite state machines. This technique checks the presence of "<script>" tag in the whole document [25]. This technique is not at all effective for identifying XSS concerns since online programmes more regularly utilise their own scripts and because there are many different ways to start a JavaScript interpreter. Because of these two factors, the methodology is not at all useful[26].

The authors of Bounded Model Checking[27]Tsai, use counterexample traces to reduce the number of sanitization processes inserted and to pinpoint the major cause of mistakes. This improves the accuracy of code instrumentation as well as error reporting. It was necessary to provide states to variables that reflected the current degree of trust in order to conduct a check on the legal flow of information inside a web application. Using the Bounded Model Checking strategy, it was now time to validate the accuracy of each safety state by confirming that the Abstract Interpretation of the programme had been correctly interpreted. Significant inaccuracies were evident in their methodology, such as the failure to include alias analysis or the insistence on addressing file resolution concerns[28].

Techniques for judging the quality of software In order to locate security flaws in a web application, Y. Huang, S. Huang, Lin, and Tsai [11] use a variety of software testing strategies, such as black-box testing, fault injection, and behaviour monitoring. This strategy incorporates black-box testing with user behaviour modelling and user experience modelling. A great number of supplementary programmes, such as WebInspect, APPScan, and ScanDo, have used methods that are similar to these. Because the purpose of these approaches is to locate mistakes at an early stage in the development process, it is possible that they will not be able to instantly protect web applications, nor will they be able to ensure that all flaws will be found[29].

The tactic of tracing the transit of information from its origin to its destination, known as the taint propagation approach, is based on the examination of data flow. Both static and dynamic techniques make use of this technique[30]. This strategy is based on the following presumptions: If sanitization is carried out throughout each channel, beginning at the source and ending at the sink, the programme may be considered safe [31]. Because it is possible for certain XSS vectors to readily bypass several filters that are designed to be robust, it is not advisable to rely just on the user's filter and ignore the sanitization function. As a direct consequence of this, it does not provide a dependable strategy for ensuring security in this scenario [32].

International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 2 Feb. 2023, pp: 822-833 www.ijaem.net ISSN: 2395-5252

Using Data Provided by Users in Conjunction with Untrusted Scripts Dangerous scripts are detected in user-provided data by using a list of untrusted scripts. This technique raises a doubt on the approach that Wassermann and Su are using at the moment. In this particular case, the procedures that were done included the creation of policies and untrusted tag regular expressions, as well as the determination of whether or not the CFG provided by String taint static analysis and the created regular expression overlap. In the event that the outcome was favourable, it was necessary to carry out further steps. Utilizing script that is not reliable is a poor concept, despite the fact that it may seem to be simple. In the similar vein, the OWASP paper makes the same point[4]. The article makes it very clear that checking for cross-site scripting vulnerabilities (XSS) in input or encrypting output should not be done via "blacklist" validation. the search and replacement of a few undesired characters (such as "", ">") that was badly planned but was utilised efficiently. The blacklist validation process may be readily sidestepped by a wide variety of XSS attacks.

Dynamic analysis techniques as well as embedded policies that the browser is responsible for enforcing the web application provides the browser with a whitelist of all safe scripts, which safeguards the browser from being compromised by malicious code. It was a good idea to restrict script execution to those on the provided list, but because the parsing algorithms used by browsers differ, a filtering technique that works well for one browser may not be effective for another browser. It was a good idea to restrict script execution to those on the provided list. Even if the method described in this article is highly effective against the circumstances described in the previous sentence, the policy cannot be implemented without first changing each browser. Because of this, the online application suffers from scalability issues, as seen from the perspective of the user [33]. Every single one of our customers' computers must have this most recent version of the browser installed.

Technique of Syntactical Structure Su and Wassermann proposed a hypothesis that asserts that the exploited entity's syntactical structure changes whenever an injection attack is successful[34]. This theory is known as the syntactical structure technique. They provide a method for locating harmful payloads by analysing the syntactic structure of the output string, which may be found in their work. It is necessary to provide metadata in the user input in order to trace this sub-string from its origin to its destination. This information assists the modified parser in analysing the syntactical structure of the dynamically produced string by marking the end and start locations of the user-provided data. This information may be found in the modified parser's output. In addition, the process was stopped if there was even the slightest indication of an anomaly. It has been shown that this method is highly effective in finding injection vulnerabilities other than XSS. These types of workflow vulnerabilities are brought on by the interaction of many modules, and it is not possible to eliminate them just by examining the syntactic structure [35].

Noxes is a web proxy that uses a proxy-based method to prohibit the transfer of sensitive information from the site of the victim to the site of a third party. This keeps the victim's information safe [36]. The elimination of malware as well as its detection is the purpose of this application-level firewall. Users have full control over every connection that comes into or goes out of their local workstations, down to the most granular level. When the firewall detects that a connection does not comply with its rules, it will inquire as to whether the user wishes to accept or deny the connection. If a URL is blacklisted, there is no guarantee that it will prevent cross-site scripting assaults. The proxy-based technique careful setup and lacks the capacity to identify errors automatically. This method may result in a greater number of false positives due to the fact that it protects the unexpected connection without addressing the problem. Pietraszek and Berghe have developed a method that is based on the utilisation of an interpreter that tracks untrusted input at the character level and employs instrumentation to discover vulnerabilities via context-sensitive string evaluation at each vulnerable sink. Pietraszek and Berghe's method can be found in their paper “A Method Based on the Utilization of an Interpreter That Tracks Untrusted In”[37]. This approach is sound, and after the level of security has been raised, it could be possible to enhance it by replacing the interpreter. Changing the interpreter is a strategy that may be used for a variety of different online programming languages, such as Java, JSP, and Servlets. However, it is more difficult to adapt to this approach for these languages.

Analyses of both the static and dynamic aspects utilising a lattice-based method In order to identify potential vulnerabilities, a piece of software known as WebSSARI combines static and runtime features, then does static taint propagation analysis. WebSSARI makes use of an intra-procedural flow sensitive technique that is based on

International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 2 Feb. 2023, pp: 822-833 www.ijaem.net ISSN: 2395-5252

a type state and lattice model in order to locate potential security flaws. This programme will automatically incorporate runtime safeguards, often known as sanitization processes, if it determines that corrupted data has made its way to a sensitive function after it has detected that the function has been reached. The intra-procedural type-based evaluation that this method employs often leads to the production of erroneous positive and negative conclusions, which is the approach's primary shortcoming [38]. In addition to that, this method considers the outcomes of user-created filters to be safe. Because of the high probability that a malicious payload will not be discovered by the filtering approach that has been used, the genuine vulnerabilities may not be notified[39].

iv. Prevention

The point of deployment, which may be either the client's computer or the server's computer, is one of the most essential considerations when comparing different XSS protection approaches.There are two different kinds of testing tools, which include Both white-box and black-box vulnerability scanners have been proposed in earlier studies, and both have been successfully implemented in real-world scenarios. Even while these tools may normally aid in the detection of cross-site scripting vulnerabilities, there is a compelling argument for adopting additional security measures for online applications. This argument is based on the fact that there is a good need for doing so. It is suggested to use a firewall at the application level. This kind of firewall is placed on a security gateway that sits between the server and the client. It is responsible for performing all transformations and checks that are connected to security. The technology assists Web developers in the deployment of countermeasures against cross-site scripting (XSS) attacks by separating the security-relevant code from the rest of the application and offering a specialised Security Policy Description Language to build it [31].

Hybrid Approaches of Mitigation Some treatments use both traditional and modern hybrid methods, such as using the web browser. The server annotates the content that is sent and gives data on the validity or degree of rights that scripts have. These annotations are the responsibility of the web browser, which must validate and enforce them[40]. BEEP (Browser-Enforced Embedded Policies) [41]suggests using a modified browser that blocks any attempts to launch scripts and compares them to a policy that must be given by the server. This approach is recommended by the authors. There are two distinct policy types that come highly suggested. The scripts were first approved for use after a whitelist of hashes that the browser might use to check them against was created. Second, the HTML source nodes that are intended to include user-provided content should be labelled so that the browser can determine if the location of a script within the DOM tree is inside user-provided content[42]. This will allow the browser to determine whether or not user-provided content is being used. The updated browser performs a check of each script to ensure that it complies with the policy, and it prevents scripts from running if the scripts fail the check. Since the information flow-based job would result in more false positives in 2004, [27]a high information flow rate is not a sign of strength.Validation strategies and scanners have been put up as potential solutions to the problem of XSS vulnerabilities [43].

Additional software engineering strategies, such as WAVES for security analysis, have also been published since their inception [44]. However, every one of the proposed methods is antiquated and would be rendered useless if tags were allowed in web applications. Even though Jayamsakthi Shanmugam's solutions are based on financial and non-financial applications, they do not manage XSS attacks that come from multiple interfaces[43]. This is because of the breadth of the solutions. The information divided into multiple different trust classes by using randomized XML namespaces inside Nonce spaces. It is the client's responsibility to comprehend namespaces, and the rights to the content must be governed in accordance with a policy that is supplied along with the website. By using XPath expressions, the owner of the website can determine the necessary trust levels and disallow the inclusion of JavaScript code in HTML subtrees that are designed to hold user-submitted material[45]. The previously described hybrid risk mitigation methods provide the most effective attributes and the best cost-to-protection ratio for parameterization. Nevertheless, they suffer from the same limitation as systems that are dependent only on the client computer, namely the need for user machine deployment.

## V. DISCUSSION

One of the main challenges in malware injection attacks is avoiding detection by security software and other defensive measures. Attackers may use obfuscation techniques, such as encryption and packing, to make the malware code harder to detect. Attackers must carefully select the target application or system component to inject the

malware into, as not all components are equally vulnerable. The target must be one that is used by many users and has a large attack surface but is also not too well protected. Attackers may need to first gain access to the target system before they can inject the malware. This can involve exploiting vulnerabilities in the system, social engineering attacks, or other methods. Once the malware has been injected, attackers need to ensure that it remains in the system and can survive reboots and other system changes. This can involve hiding the malware code, modifying system settings, and other techniques. Malware injection attacks can interfere with the normal operation of the target system, causing crashes and other errors. Attackers need to carefully manage the injection process to avoid these types of issues. Antivirus software is designed to detect and remove malware, so attackers need to create malware that can evade detection. They may use techniques such as polymorphism and metamorphism to make the malware code look different each time it is executed.Malware injection attacks are complex and require a high level of skill and knowledge to carry out successfully. However, as these attacks can be highly effective, they remain a significant threat to computer systems and networks.

We have compared the methods used today to identify and stop SQLIAs. We initially determined the different kinds of SQLIAs that are currently known in order to do this evaluation. We next assessed the strategies under consideration on the basis of our ability to spot and prevent such threats. Additionally, we also delved into many ways that SQLIAs may be included into applications and determined which strategies might be used with specific mechanisms. We outlined each technique's implementation needs and assessed how much its prevention and detection processes might be entirely mechanized. Numerous solutions struggle to defend against threats that use badly written stored processes and are concealed by alternative encodings. Based on the differential between preventive-focused and broad detection and prevention strategies, we also discovered a broad difference in prevention capacities. Future assessment effort ought to concentrate on assessing the precision and usefulness of the methodologies.Both academic institutions and private companies are actively engaged in ongoing research on the detection or prevention of XSS. Even if automated tools and security systems have been put up in order to accomplish those goals, none of them is comprehensive or precise enough to ensure a consistent level of safety for online applications. One of the primary factors that contributes to this problem is the absence of a generally adopted and thorough method for evaluating system performance. Another factor is the need to regularly update the system's source code, which brings with it additional burden. It is necessary to have a system in place that can be promptly implemented and functions properly in order to detect and prevent attacks using cross-site scripting (XSS). These attacks may be particularly damaging since they can compromise sensitive data [46]. The method of detecting code injection may be used to any other programming language that you like. It would be sufficient to simply swap out the encoding module and make use of an encoder that was developed particularly for each new language [47].

## VI. CONCLUSION

The worrisome rise in the number of cyberattacks in today's world, in which almost everything is conducted online, becomes a major cause for worry. To protect the confidentiality of sensitive information that belongs to both the public and the government, certain preventive measures, such as the creation of new laws or the revision of existing ones, should be taken by both the general public and the government. Insufficient awareness of cyber security measures is one of the primary factors contributing to these attacks. Nobody, not even the government or any other industry, wants to admit that they have been the target of these attacks or breaches because it will damage their reputation in the eyes of the public and make them wonder how they can protect us and our sensitive information if they themselves are not protected. This is because admitting that they have been the target of these attacks or breaches will cause the public to question how they can protect us and our information if they themselves are not protected. This study describes how to recognize a cyberattack, how hackers get access to networks, and how we can secure our computer systems from cyberattacks or stop them from happening altogether. The purpose of this post is to raise awareness and provide information for readers who have a significant concern about the safety of their networks, application, websites or the data they store.

## VII. ACKNOWLEDGEMENT

The authors would like to thank all School of Computing members who were involved in this study. This study wasconducted for the purpose of Ethical Hacking & Penetration Testing Research Project. This work was supported by Universiti Utara Malaysia.

## REFERENCES

[1]. M. Hasan, Z. Balbahaith and M. Tarique, "Detection of SQL injection attacks: a machine learning approach," In 2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA), pp. 1-6, November 2019.

[2]. H. Maurel, S. Vidal and T. Rezk, "Comparing the Detection of XSS Vulnerabilities in Node. js and a Multi-tier JavaScript-based Language via Deep Learning," In ICISSP 2022-8th International Conference on Information Systems Security and Privacy, February 2022.

[3]. A. Bhardwaj, C. Saheb Singh, B. Aniket, M. Shubham and U. Deepak, ""Detection of Cyber Attacks: XSS, SQLI, Phishing Attacks and Detecting Intrusion Using Machine Learning Algorithms."," In 2022 IEEE Global Conference on Computing, Power and Communication Technologies (GlobConPT), pp. 1-6, September 2022.

[4]. G. E. Rodríguez, J. G. Torres, P. Flores and D. E. Benavides, "Cross-site scripting (XSS) attacks and mitigation: A survey," Computer Networks, vol. 166, p. 106960, January 2020.

[5]. W. G. Halfond, J. Viegas and A. Orso, "A classification of SQL-injection attacks and countermeasures," In Proceedings of the IEEE international symposium on secure software engineering, pp. 13-15, March 2006.

[6]. F. Q. Kareem, S. Y. Ameen, A. A. Salih, D. M. Ahmed, S. F. Kak, H. M. Yasin and N. Omar, "SQL injection attacks prevention system technology," Asian Journal of Research in Computer Science, pp. 13-32, July 2021.

[7]. P. Roy, R. Kumar and P. Rani, "SQL Injection Attack Detection by Machine Learning Classifier," In 2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC), pp. 394-400, May 2022.

[8]. L. Ma, D. Zhao, Y. Gao and C. Zhao, "Research on SQL injection attack and prevention technology based on web," In 2019 International Conference on Computer Network, Electronic and Automation (ICCNEA), pp. 176-179, September 2019.

[9]. Y. W. Huang, F. Yu, C. T. C. H. Hang, D. T. Lee and S. Y. Kuo, "Securing web application code by static analysis and runtime protection," In Proceedings of the 13th international conference on World Wide Web, pp. 40-52, May 2004.

[10]. M. Alenezi, M. Nadeem and R. Asif, "SQL injection attacks countermeasures assessments," Indonesian Journal of Electrical Engineering and Computer Science, pp. 1121-1131, February 2021.

[11]. Y. W. Huang, S. K. Huang, T. P. Lin and C. H. Tsai, "Web application security assessment by fault injection and behavior monitoring," In Proceedings of the 12th international conference on World Wide Web, pp. 148-159, May 2003.

[12]. C. Gould, Z. Su and P. Devanbu, "JDBC checker: A static analysis tool for SQL/JDBC applications," In Proceedings 26th International Conference on Software Engineering, pp. 697-698, May 2004.

[13]. S. Chopra, H. Marwaha and A. Sharma, ""Cyber-Attacks Identification and Measures for Prevention."," In International Conference on Cybersecurity and Cybercrime, pp. 83-90, 2022.

[14]. G. Wassermann and Z. Su, "Static detection of cross-site scripting vulnerabilities," In Proceedings of the 30th international conference on Software engineering, pp. 171-180, May 2008.

[15]. W. G. Halfond and A. Orso, "Combining static analysis and runtime monitoring to counter SQL-injection attacks," In Proceedings of the third international workshop on Dynamic analysis, pp. 1-7, May 2005.

[16]. V. Nithya, S. L. Pandian and C. Malarvizhi, "A survey on detection and prevention of cross-site scripting attack," International Journal of Security and Its Applications, pp. 139-152, 2015.

[17]. H. B. S. Reddy, "A Proposal: For Emerging Gaps in Finding Firm Solutions for Cross Site Scripting Attacks on Web Applications," Journal homepage: www. ijrpr. com ISSN, pp. 3982-3985, July 2022.

[18]. J. Kumar, A. Santhanavijayan and B. Rajendran, "Cross site scripting attacks classification using convolutional neural network," In 2022 International Conference on Computer Communication and Informatics (ICCCI), pp. 1-6, January 2022.

[19]. L. K. Shar and H. B. K. Tan, "Defending against cross-site scripting attacks," Computer, pp. 55-62, 2011.

[20]. P. Panwar, H. Mishra and R. Patidar, "An Analysis of the Prevention and Detection of

Cross Site Scripting Attack," International Journal, pp. 30-34, January 2023.

[21]. R. Kadhim and M. Gaata, "A hybrid of CNN and LSTM methods for securing web application against cross-site scripting attack," Indonesian Journal of Electrical Engineering and Computer Science , pp. 1022-1029, February 2021.

[22]. M. D. Ambedkar, N. S. Ambedkar and R. S. Raw, "A comprehensive inspection of cross site scripting attack.," In 2016 international conference on computing, communication and automation (ICCCA), pp. 497-502, April 2016.

[23]. A. S. Christensen, A. Møller and M. I. Schwartzbach, "Precise analysis of string expressions," In Static Analysis: 10th International Symposium, SAS 2003 San Diego, CA, USA, June 11–13, 2003 Proceedings, May 2003.

[24]. M. Mohri and M. J. Nederhof, "Regular approximation of context-free grammars through transformation," Robustness in language and speech technology, pp. 153-163, 2001.

[25]. Y. Minamide, "Static approximation of dynamically generated web pages," In Proceedings of the 14th international conference on World Wide Web, pp. 432-441, May 2005.

[26]. F. M. M. Mokbal, W. Dan, W. Xiaoxi, Z. Wenbin and F. Lihua, "XGBXSS: an extreme gradient boosting detection framework for cross-site scripting attacks based on hybrid feature selection approach and parameters optimization," Journal of Information Security and Applications, p. 102813, May 2021.

[27]. Y. W. Y. F. Huang, C. T. C. H. Hang, D. T. Lee and S. Y. Kuo, "Verifying web applications using bounded model checking," In International Conference on Dependable Systems and Networks, pp. 199-208, 2004.

[28]. J. Feist, G. Grieco and A. Groce, "Slither: a static analysis framework for smart contracts," n 2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB), pp. 8-15, May 2019.

[29]. J. Li, "Vulnerabilities mapping based on OWASP-SANS: a survey for static application security testing (SAST).," Annals of Emerging Technologies in Computing (AETiC), 2020.

[30]. Ö. A. Aslan and R. Samet, "A comprehensive review on malware detection approaches," IEEE Access, pp. 6249-6271, December 2020.

[31]. D. Balzarotti, M. Cova, V. Felmetsger, N. Jovanovic, E. Kirda, C. Kruegel and G. Vigna, "Saner: Composing static and dynamic analysis to validate sanitization in web applications.," In 2008 IEEE Symposium on Security and Privacy (sp 2008), pp. 387-401, May 2008.

[32]. Z. Su and G. Wassermann, "The essence of command injection attacks in web applications," Acm Sigplan Notices, pp. 372-382, 2006.

[33]. P. Bisht and V. N. Venkatakrishnan, "XSS-GUARD: precise dynamic prevention of cross-site scripting attacks," Detection of Intrusions and Malware, and Vulnerability Assessment: 5th International Conference, DIMVA, pp. 23-43, July 2008.

[34]. S. D. Samarin and M. Amini, "Preventing SQL injection attacks by automatic parameterizing of raw queries using lexical and semantic analysis methods," Scientia Iranica, pp. 3469-3484, January 2019.

[35]. D. Balzarotti, M. Cova, V. V. Felmetsger and G. Vigna, "Multi-module vulnerability analysis of web-based applications," In Proceedings of the 14th ACM conference on Computer and communications security, pp. 25-35, October 2007.

[36]. E. Kirda, C. Kruegel, G. Vigna and N. Jovanovic, "Noxes: a client-side solution for mitigating cross-site scripting attacks," In Proceedings of the 2006 ACM symposium on Applied computing, pp. 330-337, April 2006.

[37]. T. Pietraszek and C. V. Berghe, "Defending against injection attacks through context-sensitive string evaluation," In Recent Advances in Intrusion Detection: 8th International Symposium, pp. 124-145, September 2006.

[38]. Y. Xie and A. Aiken, "Static Detection of Security Vulnerabilities in Scripting Languages," In USENIX Security Symposium, pp. 179-192, August 2006.

[39]. M. A. Kausar, M. Nasar and A. Moyaid, "SQL Injection Detection and Prevention Techniques in ASP .NET Web Application," International Journal of Recent Technology and Engineering (IJRTE, pp. 7759-7766, September 2019.

[40]. Z. C. S. S. Hlaing and M. Khaing, "A detection and prevention technique on sql

injection attacks," In 2020 IEEE Conference on Computer Applications (ICCA), pp. 1-6, February 2020.

[41]. T. Jim, N. Swamy and M. Hicks, "Defeating script injection attacks with browser-enforced embedded policies," In Proceedings of the 16th international conference on World Wide Web, pp. 601-610, May 2007.

[42]. E. Uzun, "A novel web scraping approach using the additional information obtained from web pages," IEEE Access, pp. 61726-61740, March 2020.

[43]. J. Shanmugam and M. Ponnavaikko, "A solution to block cross site scripting vulnerabilities based on service oriented architecture," In 6th IEEE/ACIS International Conference on Computer and Information Science, pp. 861-866, July 2007.

[44]. T. Pattewar, H. Patil, H. Patil, N. Patil, M. Taneja and T. Wadile, "Detection of SQL injection using machine learning: a survey," International Research Journal of Engineering and Technology (IRJET), pp. 239-246, November 2019.

[45]. H. Wu, Z. Yu, D. Huang, H. Zhang and W. Han, "Automated enforcement of the principle of least privilege over data source access," In2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 510-517, December 2020.

[46]. S. Abaimov and G. Bianchi, "CODDLE: Code-injection detection with deep learning," IEEE Access, pp. 128617-128627, 2019.

[47]. M. Alghawazi, D. Alghazzawi and S. Alarifi, "Detection of sql injection attack using machine learning techniques: a systematic literature review," Journal of Cybersecurity and Privacy, pp. 764-777, September 2022.