Catalyst Research Volume 23, Issue 2, September 2023 Pp. 1226-1233

METHODS AND ALGORITHMS OF PROTECTING WEB APPLICATIONS AGAINST COMMON ATTACKS

Komil F. Kerimov, Zarina I. Azizova
Tashkent University of Information Technologies named after Muhammad al-Khwarizmi, Uzbekistan
Email: {kamil@kerimov.uz}, z.i.azizova18@gmail.com

ABSTRACT

The research of this paper focuses on attacks on web applications. String operations are the easiest to introduce during the web browsing and form transfer process, and consequently occur most frequently. We look at the characteristics and defense mechanism against these types of attacks to get a better solution in practice.

Database management systems are often installed on the backend of various types of web applications as a platform, to provide for fetching and writing all kinds of data. SQL is the most popular query language for a relational database.

This paper presents the best method for configuring system security using a firewall application, which is widely used by modern enterprises to secure the entire enterprise infrastructure. We investigate several common attack techniques such as cross-site scripting and SQL injection, and also present a new method for configuring system settings to enhance protection against common attacks from attackers.

Keywords: cross-site scripting (XSS), SQL-injection, web application firewall (WAF), SQL-query

INTRODUCTION

According to a network security incident research effort, 75% of the attacks were focused on web applications. However, these attacks couldn't be easily detected or prevented. Most of the problems occurred due to ignorance of software security during application or platform development. Application developers have a wide range of capabilities given their innate talents and professional skills. As a result, applications are frequently written in ways that are insecure from an information security point of view. In addition, some application platforms are developed and maintained entirely by a third-party vendor. These factors mentioned above not only lead to system or web application security vulnerabilities, as well as logical defects or transaction flow vulnerabilities, but also lead to loss of assets and the reputation of the enterprise itself due to these management issues.

The most fundamental solution for removing a web application vulnerability must be solved by making patches to the system or to the software itself. This can also be done using the white box testing method, which involves code analysis, system or web application vulnerability scanning tools, a penetration test, that requires higher level methods in order to find out about the problems

Catalyst Research Volume 23, Issue 2, September 2023 Pp. 1226-1233

of your own application. It is also possible to install a firewall for the web application in the front-end of the application node to ensure that the system is protected.

The web application firewall [1] is a security equipment that belongs to the application layer. But incorrectly configured security settings will result in an insufficient level of security for the software or hardware. In this research work, we use a hardware-based, industrial-grade application firewall as the hardware to verify the implementation of security models.

In addition to the security features in the system, we also deal with key characters for suppressive attack schemes such as SQL injection and cross-site scripts. Coding, conversion, deletion and other handling on the key symbols are required to avoid this attack behavior on the server or browser side. We also maintain a blacklist of keywords that need to be prevented in network traffic to improve system security. As a test environment, we create an e-commerce web application and install vulnerability scanning software of the application; we also test the servers by applying the most appropriate protection setting. The result is compared with a server scanning test without such setting to confirm the efficiency of the protection, which effectively prevents SQL injection and cross-site scripting attacks.

Characteristics and Defense Mechanism against Web Application Attacks

Web pages and web applications can be browsed by communicating between the user side ('client') and the host side ('server') via the HTTP or HTTPS protocols. The operation starts with the user entering a specific URL (or URI) into the browser, which sends a request to connect to the specified host. When the connection is established, the client sends one or more requests to access the web content. Once the application server has receives the request, it processes the request and responds with web content to the user. Usually, the request and response are paired. Most of the presented website not only uses a single Request/Response pair, but also a continuous Request/Response transmission in a single established connection session. Information can be transmitted via continuous Inquiry/Response pairs. Users can submit a request for information by clicking in a web browser or by filling out a form. The application node performs further processing regarding the user's request and responds accordingly. The web page displayed in the browser is a combination of the entire related Request/Response, as shown in Figure 1.

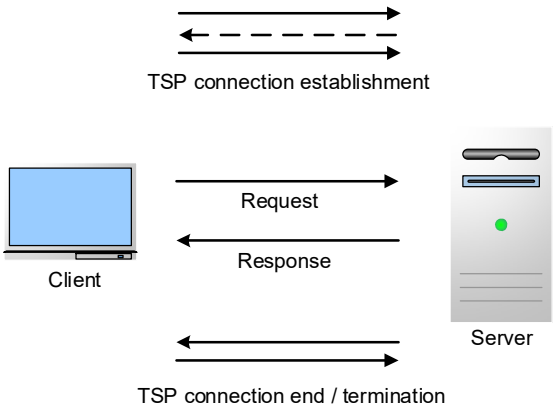

Fig 1. Diagram of interaction between client and web server.

In this way, malicious attacks on a website imitate the request response behavior used in normal

Catalyst Research Volume 23, Issue 2, September 2023 Pp. 1226-1233

browsing and perform a malicious action during a Request; or use certain programming languages or browser design features to perform a malicious action when the server sends a Response message to the client. Either way, the attacks perform mostly by transmitting, typing or combining special grammar, text strings, malicious links and semantic tautology information. These attacks commonly exploit the initial web processing procedure causing the server or browser to fail in parsing the grammar or content. Tautology is also used to evade the normal logic checking mechanism in order to execute the attack.

The implementation of structured query language (SQL injection), cross-site scripting attack (XSS attack) [6] and false cross-site queries (CSRF) are common attacks using a combination of sequences or programming language syntax.

Some attacks are produced by the system or the application environment itself. They use a URL or form fields as an entry point to enter attack data when browsing the application website. This data is malware or functions that can be executed on the server and attack the host application to get system information such as password, file, or other important system configuration information, etc. Sometimes, hackers can download other malware through specific features of the application system. For example, using wget to connect to a remote host and download a trojan, backdoor and other malware. Hackers can also introduce commands to add new users or elevate the privileges of existing users. For example, a network command can be used to add a new user account and password to a Windows system.

In addition, some attacks occur when connecting to or browsing a website. These are usually accompanied by the transmission of information entered or stored when data is entered into form fields or a session is established. This type of attack, which interrupts or interferes with transmitted information, is also known as a data validation attack. Another communication service-based malicious action is called a denial-of-service attack (DoS attack) [2-5]. It uses an emergency connection, data request, or large connection that results in an interruption or failure of normal service. Recent studies show that among all malicious actions, SQL injection and cross-site scripting attack are the most common and the most serious attacks.

This paper's research focuses on attacks against web applications. String operations are the easiest to inject during the web browsing and form transfer process, and as a consequence, occur most frequently. We examine the characteristics and defence mechanism against these types of attacks in order to obtain a more effective solution in practice.

Database management systems are often installed on the backend of various kinds of web applications as a platform, to provide sampling and recording of all kinds of data. SQL is the most popular query language for a relational database.

After the web server receives a frontend user request, it converts the required information into parameters and sends them to the backend database for further processing, together with the working commands of the database. It then retrieves the relevant information and passes the result back to the user. For example, a simple web page provides a user authentication mechanism by retrieving the username and password from form fields. The username field is called uid and the password field is called passwd. The username and password are stored in a database table called

Catalyst Research Volume 23, Issue 2, September 2023 Pp. 1226-1233

user table. The backend server can implement the following SQL syntax:

select * from user_table where uid='"&request("uid") &"'and password='"&request("passwd")&"'

The SQL command retrieves a string from the user_table where uid and password fields are the data entered by the user. In normal circumstances, if the data entered by the user is uid=user and password=abc123 and this information matches the data stored in the table, access to the website can be granted. However, if you instead enter a different SQL query in the uid and password fields, this will probably result in an unexpected script execution sequence that is different from the original server response as originally intended by the website developer.

In the same way, using a website property that allows users to enter data, a piece of HTML or Java/VB script syntax can be entered, so that the server will first save or process the input data. Later, when the user wants to view the relevant page, the server must respond to the request. Legitimate information automatically merges with the malicious information, fills the HTML or script, and then is presented to the user's browser. This will result in unexpected actions that are not embedded in the web application, leading to a degree of negative impact.

The target of such attack is not the server that provides the web application services, but the users browsing that site. When a user browses a web page containing attacker material, a certain Java Script, VB Script or other dynamic languages like ASP and PHP are initiated which are not embedded in the original web page. The implementation of further actions is unpredictable. They could consist of connecting to other websites, downloading unwanted data or injecting a Trojan that causes some degree of performance degradation. The degree of damage depends on the malware that was downloaded.

MATERIALS AND METHODS

The most dangerous attack, such as the injection of SQL commands [7], occurs because of careless input control. Input strings are executed because they are mistaken as part of the program or syntax. Similarly, hackers can exploit a validity problem in a logical programming language statement to imitate unexpected results and avoid a regular security check mechanism.

As shown in Figure 1 above, the user will be sent an HTTP request to the server. By using requests like GET, POST or other methods, the client can get pictures, text or other useful information from the web server. The user can also send the information to the server for further processing; once the server receives these requests and interacts with them accordingly, it passes the requested information back to the user, to be added to the complete web page.

If we will add application layer protection mechanisms to this client-server architecture, such as the application firewall as shown in Figure 2, during the browsing process, the HTTP request made by the user will be forwarded to the application firewall. The application firewall will perform filtering checks according to the settings. If there are no security concerns, the message can then be forwarded to the backend application server.

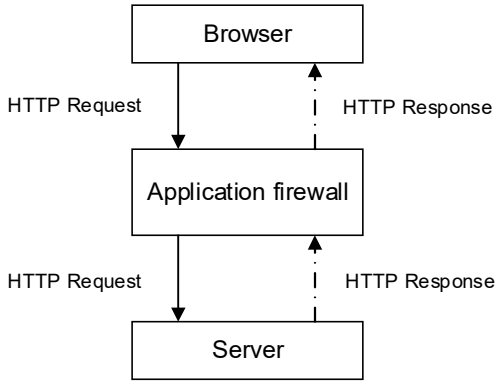

Fig 2. Scheme of client-server interaction with the security device.

After the user request is fully processed, it follows the same path back to the end user himself. In this case, we focus only on possible attacks from the user to the server and discuss defenses. That is, we would like to investigate a defense mechanism against server input.

Protection against SQL injection

Such attacks must use a logical approach and reasonable input values, together with the disruption of special characters of the source program accompanied by a normal SQL-query, to provoke the return of a tautologically correct value. If this type of attack was applied to the authentication login page, the identity authentication mechanism can easily be avoided and the login can be successfully logged in. It means that the hacker has a legal right to access the system resource. To figure axis labels, use words rather than symbols. Do not label axes only with units. Do not label axes with a ratio of quantities and units.

Protection against SQL injection

Such attacks must use a logical approach and reasonable input values, together with the disruption of special characters of the source program accompanied by a normal SQL-query, to provoke the return of a tautologically correct value. If this type of attack was applied to the authentication login page, the identity authentication mechanism can easily be avoided and the login can be successfully logged in. It means that the hacker has a legal right to access the system resource.

The most fundamental solution to defend against such attacks is to strengthen the verification mechanism of the application programme. All input requires detailed checks to determine the purity of the input data values before being passed to the downstream program for subsequent execution. In this way, we can eliminate the possibility of malicious input attacks.

Regarding database access, experience shows that passing an SQL-query by string concatenation is not the best method. This method not only leads to security problems, as mentioned above, but also inadvertently leaks information about the database structure and the logical way the program works. A better approach is to send parameters to gain access to the database, together with checking the input parameters. This is more effective in preventing security problems. Control of responses to error messages should also be strengthened. Too much information should not be shown to the user. Users can gather error messages from the database by trial and error, and then refine the attack.

Catalyst Research Volume 23, Issue 2, September 2023 Pp. 1226-1233

For input values of a parameter, if it contains special characters, the check must be strengthened. For example, further processing is required for single quotes ('), semicolons (;) and left slashes (\) so that the combination of SQL syntax with these special characters can be treated as a word or sequence rather than part of language syntax or grammar.

Taking Citrix as an example, in its special character handling mechanism, if " (single quotes) is encountered, an extra single quote is added before the character. Thus, the original characters will become purely symbolic because of this inverted comma. Combining the SQL-query in the process will not lead to unexpected results because of the logical judgement. For example, the original SQL looks like this:

select * from usertab where uid = ’1989’ …

If the malicious user input uid - with single quotes to replace 1989, such as 'or a = a--, the original sentence will become

select * from usertab where uid=’ ’or a=a --’ …

The backend server will treat uid as an empty sequence and the grammar structure will return tautological TRUE; After contacting special characters this will be turned into the following grammar

select * from usertab where uid=’ ’’or a=a --’ …

Despite the fact that there is still a grammatical structure problem, at least it does not make the uid parameter values larger than a true logical constant. Similarly, whenever "\" (backslash) is encountered, we can simply insert an extra slash to the left before the character to take it out of subsequent characters; in case ";" (semicolon) is found, we can simply remove the characters to prevent the SQL syntax character from being erased, so it will not cut off the normal and unfinished statement following the semicolon.

Protection against cross-site scripting

This type of attack is also caused by negligence in legitimate input validation. Malicious code can be entered as a sequence. When it is online, the browser treats it as a program and executes the code as an instruction. The severity of the impact depends on the functionality of this malware. It can direct the browser to execute a remote program or load an auto-executable program designed to do the trick when the drive is plugged into the system. The auto-exploiter might also download other malware from a pre-configured source and execute it on the local machine. Thus, the controller only needs to update the malicious code on the remote transfer station. The victim will be subjected to a lengthy attack. The other example is the theft of a victim's cookie. This cookie can be used successfully through a website's identity authentication mechanism, using the victims identity to gain access to the website.

Defending against this type of attack is similar to protecting against SQL injection

However, the syntax or symbols that are subject to protection checks mainly focus on the syntax of the web application that the browser can recognise. Within special characters, we must pay attention to angle brackets (<or >), the percent sign (%) or the corresponding conversion text. We can set the Citrix protection configuration to decode special characters once or several times, and

Catalyst Research Volume 23, Issue 2, September 2023 Pp. 1226-1233

then recover with real characters to determine if they are special characters. This will protect the backend server from replying to the user with malicious code embedded in the web page.

Despite the fact that the victim of such attacks is not the website itself, once the security incident becomes public knowledge, the negative image of the affected businesses or organisations will directly affect their profits or perception by the public.

Protection against the injection of commands

When protecting a system against the inclusion of a capability or command of the operating system or the database system itself, it is important to regularly refer to vulnerability alerts on linked sites. To strengthen the security of the system, it is highly recommended to update the system to the latest patch in a timely manner as soon as a significant security vulnerability is alerted.

In addition, the security of the web application itself also needs to be strengthened. A content filtering mechanism can also be installed on the web server. Through the web server firmware or third party software, the "Overwrite" function can be performed under certain circumstances. This feature applies regular expressions to set a black and white list of characters and commands to filter and inspect a website. If a message to be sent to the server or a request to access website content contains illegal characters, the browser will be redirected to the specific page with a mandatory warning or receive a specific error message. You can restrict link source access to avoid entering a specific URL directly to avoid a specific page or information.

Commonly, firewall applications contain appropriate content filtering features. Through input/output content filtering, they not only effectively prevent such access with special permissions, but also handle the response to certain web pages. For example, the default HTTP 404 response for a non-existent page does not exist and an error response is generated. Another erroneous page can be shown instead of the default page to reinforce information about hidden anomalies on the web-service.

System infrastructure protection

If an application firewall should be installed in an application website environment, it is recommended that in addition to the protection methods mentioned above, network infrastructure configuration adjustments should be performed. "Transparent" Mode should be prohibited; instead, the use of Reverse Proxy Mode is highly recommended. The so-called "Reverse" refers to a server that provides external services. Messages should be sent through the firewall and then to external users. This configuration requires an additional virtual IP to replace the real IP of the host. Otherwise, when users will browse the website, they will connect to the IP address of the virtual server assigned by the firewall. This virtual IP approach hides the real server IP information behind the application firewall. This allows the server to be relatively high degree of security.

CONCLUSION

This paper correlates the current prevailing methods according to their vulnerability to attacks and suggested protection attributes. We consider attacks based on string and command line operations and the corresponding protection mechanisms. If appropriate controls are implemented correctly, it will effectively reduce the injection of SQL, cross-site scripting and other attacks. This results

in a secure system environment.

In addition to the control and filtering of input/output information in the application, adjustments to the system infrastructure will improve the security of web applications.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Both the authors participated in research and wrote this paper like writing the Introduction, performing literature survey, Defense mechanism against web application attacks, Protection methodology and Conclusion.

REFERENCES

[1] Rustam Kh. Khamdamov, Komil F. Kerimov, Jalol Oybek ugli Ibrahimov, Method of Developing a Web-Application Firewall, Journal of Automation and Information Sciences Volume 51, No.6, 2019, P.61-65

[2] A.Yu. Sheglov, *Protection of computer information from unauthorized access*, Moscow, M:Nauka i tekhnika, 2004, pp. 384.

[3] K.V. Rzhavsky, *Information security: practical protection of information technologies and telecommunication systems*, Volgograd, V: Volgograd State University, 2002, pp. 50.

[4] M.K. Nizamutdinov, *Tactics of defense and attack on IT-applications*, St. Petersburg, St.P: BHV-Peterburg, 2005, pp. 30-60.

[5] V.V. Domarev, *Security of information technologies. Methodology of creating protection systems,* Moscow, M: DiaSoft, 2002, pp. 56.

[6] Veracode. Cross-Site Scripting: XSS Cheat Sheet, Preventing XSS Tutorial. Available: http://www.veracode.com/security/xss

[7] M. Sonoda, T. Matsuda, D. Koizumi, S. Hirasawa, “On Automatic Detection of SQL Injection Attacks by the Feature Extraction of the Single Character” in *Proc. International Conf. on Security of Information and Networks*, ACM, 2011, pp. 81-86.

[8] V.N. Opanasenko, S.L. Kryvyi, “Synthesis of Adaptive Logical Networks on the Basis of Zhegalkin Polynomials,” *Cybernetics and Systems Analysis,* vol. 51, no 6, pp. 969-977, Nov. 2015