Web Applications

Anthony Garcia, Michael Lee, & Phillip Waitkevich

COM-450-CA01

5 March 2019

Abstract

Web Applications are becoming a common place in most businesses. Despite being assets in most occasions, they have proven to be valuable resources for attackers and hackers. This paper will describe the use of Web Applications and common security practices. Providing the detailed idea behind security, it will open the idea of information being a target for many users. Additionally, it will follow the concepts of common attacks and the use of proper defenses.

Introduction

Web applications can secure confidential data that is stored online with modifications on the authorized access. With the advancement of technology, businesses have changed the ways of accessing and sharing information. Many companies have switched to online access to benefit employees to have remote access to operations. Many partners that live in different countries can interact online, working together to build and share sensitive data instantly until they both reach a common goal (Abela). Protecting websites and online services can withstand many different security threats that exploit vulnerabilities in the application's code (WEB). The nature of the internet follows a global scale, is exposing web properties to attack from different locations and diverse levels of complexity and scales. With web application security, it involves specifically with the website, web services, and web applications (Security?). Although web applications have proven to be very useful in many occasions, many users should take in consideration the security, possible attacks, and defense protocols.

Web application security is essential for the safety for companies and others that may use it. Web applications focus on stored data online while keeping them confidential and secure and while focusing on the principles of security for in web application security. They focus on the principles of safety, which are addressing the four conditions of security. Confidentiality is of the sensitive data that is stored in the web application is not to be exposed under any circumstances (Techopedia). This is determined to be the managers on the level that is placed on for its accesses where and how it able to accesses the data. Integrity is the data that is contained in the web application that is consistent, and it cannot be modified by an unauthorized user (Pearson). Integrity provides with the assurance that the data is reliable and accurate (Techopedia). Without having it, the cost of maintaining and collecting the data would not be justified. That is why the

procedures and policies are to ensure that the data can be trusted (Pearson). The availability is for web applications to be accessible to valid users that are on a specified period that is depending on the request (Techopedia). The users can access an information asset. However, data is not always available upon request. Some data is stored in media that may be required to access by the users. Data can be stored offline in the storage unit, not allowing immediate access. Nonrepudiation is the ability to prove that the originator of the message or communication is the actual sender by guaranteeing the authenticity of their digital signature. It ensures that the sender cannot deny what is sending or what is being modified, which is contained in a web application, but verifying with their digital signatures (Pearson).

Web Application Attacks

According to Web Applications, they have become the foundation of the Internet Infrastructure. This generally includes government services and banking. Despite being significantly used to create platforms that create content, it does attract "a large amount of sensitive data, privacy concerns also emerge" (Jian 1). Many web-based attacks can cause many communities, businesses, and individuals. This made users to take security very seriously to protect privacy and essential data. Web Applications are all shared within the same browser environment. This means that there is information based on other web applications through the browser environment. This exemplifies attackers or hackers to create malicious websites and content generally. They can determine if a site has been visited and use the chance to steal the user's information indirectly. With that said, many common attacks can occur in web applications.

Structured Query Language Injection (SQLi)

This attack injects malicious coding where a hacker can deliver SQL statements that will eventually "give them the control of a web app database server" (TablePlus). The Open Web Application Security Project, OWASP, stated that this attack is "a major source of concern for application and web developers" (Herbrandson). Some of the issues include protecting and utilizing stored usable information that is stored in a local database. They form into strings that will generally be set in many places such as login forms and search boxes. For instance, an attacker can create SQL commands that enable the database to give up essential information. Some of the data includes credit card numbers, usernames, and passwords. Furthermore, this attack can allow attackers to have access to a system.

Cross-Site Scripting (XSS)

Cross-Site Scripting (XSS) is an "attack where the front of the website acts as a launching point for attacks" and will put a huge effect on the user, visiting a website. This type of attack is what many users underestimate. Many developers tend to forget to check and test their coding. This will allow scripts to be injected. Scripts can be executed "without the site's original functionality intending them to be" (Herbrandson). Whenever there is an XSS vulnerability taking place, attackers will have an opportunity to create a code that can be executed when the victim can open the same website. Also, "the end user's browser has no way to know that the script" can be trusted and possibly execute the script (OWASP). Malicious scripts can access session tokens, cookies, or other sensitive information that is retained by the browser.

Brute Force

When a person wants to find a way to get through a wall, they will use everything in their power to penetrate until the wall falls. In a Brute Force attack, they "result in an

increased load on the underlying infrastructure" (Hofstede). They can follow malicious scripts that can be installed, which produce large amounts of memory. When a user tries to sign up by creating an account, it mostly requires a username and password. With this attack, it is designed to gain access by figuring out the exact username and password. Attackers would generally try to check if the user as a weak password. It is a long-term attack due to a giant list of possible passwords an attacker could try. They must require "all the time in the world to wait for the right password to work" (Herbrandson). Once the attacker has access to an account, it is already too late. This will lead to a wide range of illegal acts.

Distributed Denial of Service

Distributed Denial of Service (DDoS) attacks mostly intend "to take an organization or a service offline, or otherwise render resources unusable" (Kartch). Many internet users sometimes have trouble connecting too many popular websites: Wired Magazine, Reddit, and Netflix. In accordance with the Verisign Distributed Denial of Service Trends Report, it stated that this activity has increased by eighty-five percent along with thirty-two percent targeted to software service companies and IT services in 2015. DDoS attacks usually range from criminal hackers to crime agencies. These attacks exploit the vulnerability of a computer system. In addition, they use many computer systems and internet connection to flood the target's resources. In this case, it can identify other vulnerabilities and take the extra steps to gain control or resources and valuable data. This includes bypassing authentication controls or the use of malware.

Inclusion Vulnerabilities: Local File Inclusion and Remote File Inclusion

When insecure coding occurs, some malicious users are able to find functionality inside web applications. They can follow through the mechanics and be able to execute their codes.

This is an example of inclusion vulnerabilities. The Local File Inclusion involves inclusion attacks. This means that the attacker will try to trick a specific web application and be able to include files that can exploit the functionality that contains scripts and local files. By using parameters in a Hypertext Preprocessor code, many intruders can request alternative files instead of files needed to follow a specific program. With Remote File Inclusion, this attack can cause web applications to have an additional remote file by simply exploiting a particular web application that has scripts and external files. This inclusion is a "malicious software on the victim's server" and find dangerous content that can run from the same location. With that said, they can occur when there is a function that is improperly crafted. An attacker can activate a piece of code that enables them a shell that can go through the connection between the remote server and the victim's site.

Defenses against Web Application Attacks

There are many different types of attacks that can occur on web applications. However, there are many countermeasures that companies can put into place to prevent or mitigate these attacks. There specific ways to mitigate DDoS attacks and some general practices that prevent a variety of attacks. One way to stop a DDoS attack is through network devices such as routers, firewalls, and load balancers. Routers and firewalls can drop packets that they deem to be malicious, whereas load balancers will distribute a large number of requests created in a DDoS attack amongst multiple servers (Dobran 2018). The idea of various servers is part of a concept called redundancy. Redundancy describes a network that has multiple devices that do the same job, indicating that if one device goes down, the system can still do its job. Redundancy is a good security practice for DDoS attacks because even if one server goes down due to an attack, then the network can still function as it should (Dobran 2018).

Another way to mitigate DDoS attacks is to have a response plan. According to an article about DDoS attacks, a response plan is used. The response plan will indicate if "there is no time to think about the best steps to take. They need to be defined in advance to enable prompt reactions and avoid any impacts" if a DDoS hits (Dobran 2018). They allow companies to recover from an attack quickly. The Cloud is a great resource to fight against DDoS attacks, because it can immediately provide extra resources, such as bandwidth, spreading out the attack (Dobran 2018). Artificial Intelligence has proven to find ways to combat DDoS attacks. Researchers like Ming-Yang Su used a k-nearest-neighbor (KNN) algorithm (an AI algorithm used to classify things) that took inputs such as TCP flags and other packet header information to detect known DDoS attacks 97.42% of the time and unknown DDoS attacks 78% of the time (Su 2011). Because KNN is a classification algorithm, it could be useful for detecting other attacks on a web application.

In many security practices, the first form of prevention is input validation. One of the most common attack vectors for a web application attack is a user input that comes from places like the URL, the login form, and the cookie. Because of the variety of attacks that can come from an input, cybersecurity researchers began to tell people to assume "all input is evil" (Jovičić and Simić 2006). However, input validation puts a stop to most of these attacks (i.e. SQL injections, buffer overflows, Cross-site scripting, Path Traversal, Local File Inclusion, etc.) because it only permits inputs that are known to be valid and blocks everything else (Cook 2003). According to Steven Cook, author of A Web Developer's Guide to Cross-Site Scripting, "[m]any popular languages used for web development provide quick…methods for filtering or encoding special characters" (Cook 2003). Many attacks can be stopped at the programming level.

Input validation is excellent, but there are a few problems that have to be corrected by automation. One of the issues, according to Cook, is that "[t]he sheer number of possible places to check and possible permutations of user input can make checking everything exceedingly difficult" (Cook, 2003). The problem is that there are too many places to check input and so many possible inputs that it's almost impossible to validate it all manually. Automated unit testing software, such as that defined in Mohammadi et. al, can automatically check for possible injections. Intrusion Detection Systems can be programmed to statically scan user input for common malicious data (and one can note that it acts similarly to antivirus software that checks for matches in its database of viruses) (Mookhey, Burgate 2004). As mentioned in the above paragraphs, KNN and other Artificial Intelligence algorithms are useful to detect malicious inputs.

Conclusion

Web Applications have proven to be valid resources to businesses and companies as well as for attackers and hackers to play huge roles in their schemes. Along with common attacks are the defenses that can do their jobs. Web Applications will continue to be used in the future and are a technology that will not be going away for a long time. It is critical that companies become aware of these attacks and take the measures to defend them now. Otherwise, they will not be able to keep up. Because of the amount of attacks, a user must be mindful to their work in order to prevent essential data from being used from outsiders.

References:

Abela, R. (2017, December 01). Getting Started with Web Application Security. Retrieved from https://www.netsparker.com/blog/web-security/getting-started-web-application-security/

Cook, S. (2003, January 11). *A Web Developer's Guide to Cross-Site Scripting*(Tech.).

Dobran, B. (2019, February 21). 7 Proven Tactics To Prevent DDoS Attacks: Make a Security Plan Today! Retrieved March 2, 2019, from https://phoenixnap.com/blog/prevent-ddos-attacks

Herbrandson, J., Roo, Roo, & Taylor, P. (2017, May 10). Most Common Attacks Affecting Today's Websites. Retrieved March 3, 2019, from https://blog.sucuri.net/2014/11/most-common-attacks-affecting-todays-websites.html

Hofstede, R., Jonker, M., Sperotto, A., & Pras, A. (2017). Flow-Based Web Application Brute-Force Attack and Compromise Detection. *Journal of Network & Systems Management*, *25*(4), 735–758. https://doi-org.saintleo.idm.oclc.org/10.1007/s10922-017-9421-4

Jian Mao, Yue Chen, Futian Shi, Yaoqi Jia, & Zhenkai Liang. (2017). Toward Exposing Timing-Based Probing Attacks in Web Applications. *Sensors (14248220)*, *17*(3), 464. https://doi-org.saintleo.idm.oclc.org/10.3390/s17030464

Jovičić, B., & Simić, D. (2006). Common Web Application Attack Types and Security Using ASP.NET. *ComSIS*,*3*(2). Retrieved March 2, 2019.

Kartch, R. (2016, November 21). Distributed Denial of Service Attacks: Four Best Practices for Prevention and Response. Retrieved March 4, 2019, from https://insights.sei.cmu.edu/sei_blog/2016/11/distributed-denial-of-service-attacks-four-best-practices-for-prevention-and-response.html

Mohammadi, M., Chu, B., & Lipford, H. (2017). *Detecting Cross-Site Scripting Vulnerabilities through Automated Unit Testin*(Tech.).

Pearson IT Certification. (n.d.). Retrieved from http://www.pearsonitcertification.com/articles/article.aspx?p=30287&seqNum=2

Su, M. (2011). Real-time anomaly detection systems for Denial-of-Service attacks by weighted k-nearest-neighbor classifiers. *Expert Systems with Applications*,*38*(4), 3492-3498. doi:10.1016/j.eswa.2010.08.137

TablePlus. (2018, August 19). SQL Injection Attack explained, with example. Retrieved March 4, 2019, from https://tableplus.io/blog/2018/08/sql-injection-attack-explained-with-example.html

OWASP. (n.d.). Cross-site Scripting (XSS). Retrieved March 4, 2019, from https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

WEB APPLICATION SECURITY. (n.d.). Retrieved from https://www.incapsula.com/web-application-security/application-security.html

What is Web Application Security? - Definition from Techopedia. (n.d.). Retrieved from https://www.techopedia.com/definition/24377/web-application-security

What is Web Application Security? (n.d.). Retrieved March 4, 2019, from https://www.cloudflare.com/learning/security/what-is-web-application-security/