Web Application Security – What You Need to Know

Bhushan B. Gupta
BG Consulting
bhushan.gupta@comcast.net

Abstract

There have been some significant web security breaches in “Corporate America” Sony, Target, and Home Depot, to name a few. Such breaches not only impact corporations financially, they also tarnish the brand image. The customers loyal to the corporations lose their confidence in their private data security and take their business to safer pastures, thereby financially impacting the business. Both corporate America and Government agencies are working towards controlling cyber security threats.

This paper is focused on raising awareness about Web application security. It starts with an exposure to the fundamentals of Web security (Vulnerability, Threat, Risk, Exposure, and Controls), discusses control types, and touches on the principles of the Zachman Architecture and The Open Group Architecture Framework. It then dwells into the “Top 10” OWASP (Open Web Application Security Project) most critical Web application threats including, SQL Injection, XSS (Cross Site Scripting), and CSRF (Cross Site Request Forgery). Furthermore, it provides some approaches to mitigate risks to make a Web application more secure.

The paper is a self-study of Web Security and is intended to provide the necessary information to raise audience awareness.

Biography

Bhushan Gupta has 30 years of experience in software engineering, 20 of which have been in the software industry. For the past several years, Bhushan has been involved with agile processes, quality methods and metrics, and general process improvements and now with the cyber security. Bhushan has been a presenter and a reviewer for PNSQC for several years and has also presented at other conferences. As a change agent, Bhushan volunteers his time and energy for organizations that promote software quality.

Bhushan Gupta has a MS degree in Computer Science from New Mexico Institute of Mining and Technology, Socorro, New Mexico, 1985.

1 Introduction

The data breach at Target Corporation, the largest retail hack in US history, exposed 70 million records with the customer information and some 40 million stolen records with credit card information [Reuters, 2014]. Another security breach labeled as worst security breach of the year 2014 [NetworkWorld, 2014], was at Sony Pictures Entertainment. The attackers obtained access to documents, emails, and movies yet to be released. This attack also caused some political backlash since the allegations seems to have suggested that the attack originated from North Korea. The other notable attacks that took place in 2014 were Home Depot and JPMorgan Chase. There have been fewer security breaches at the Federal Govt. level but were of similar magnitude.

Clearly, cyber space has been invaded by the “Bad Guys” and our security and privacy has been threatened. The Web security of big corporations, and the Federal Govt. has been compromised, and the consumer has suffered identity theft as well as loss of credit card information. Web security is becoming increasingly important and to protect ourselves, we must have a sound understanding of security and how breaches are manifested.

2 Basic Elements of Security

A secure system should provide Availability, Integrity, and Confidentiality of critical assets; also known as the Security Triad. Availability refers to reliable and timely access to the data. Integrity is the data protection against unauthorized modification. Finally, confidentiality is assuring that the data is not disclosed inappropriately as it moves around in its environment and finally stored in its destined source. The five basic elements of security are, Vulnerability, Threat, Risk, Exposure, and Control [Harris, 2013]. Each of these are briefly described in this section.

2.1 Vulnerability

Vulnerability is the extent of unauthorized exposure of your valuable assets in combination with the countermeasures that are in place. A vulnerability may be a process in the system, a table that contains the system data, a database table, or a web service running on your server. A high degree of exposure may not be a serious problem if you have countermeasures in place. A simple analogy would be, keeping your house door wide open but guarding it.

2.2 Threat

Threat is the potential of exploiting the vulnerability by an agent and could be external or internal. An intended activity or mistake by someone working with the data is also a threat.

2.3 Risk

Risk is likelihood of exploiting the system vulnerabilities and causing negative impact on business. The risk can be made up of multiple factors such as an inadequately configured firewall, an undocumented critical manual system process, or an untrained system administrator.

2.4 Exposure

An instant of exploiting a vulnerability and thus causing a negative business impact.

2.5 Control

A control is a mechanism to reduce the potential risk. Control could be preventive, detective, corrective, deterrent, and compensating. An example of a compensating control is buying an insurance policy against the damage to your house.

2.6 Security Standards

In the early days when there were no standards for security, British Standard 7799 (BS7799) became a de facto standard but no standard body was enforcing it. The standard laid out how security should cover a wide variety of topics. The BS7799 was expanded by International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) which established ISO/IEC 2700X where X took a digit representing a necessary component of the security systems. For example, ISO/IEC 27000 provides Overview and Vocabulary and ISO/IEC 27001 provides standard for security requirements.

3 Access Control

While all the elements listed in section 2 are important, the exposure is actual incidence in which the security is compromised using an unauthorized access. The following discussion highlights the necessary elements of access control.

3.1 Identification

Identification is the process of uniquely recognizing an entity before letting it use the system. In most cases it is a unique string of characters such as a name, an email address, or an account number. In the simplest scenario the security involves verifying the string with the access control list (ACL). The methods to assure that the string is not programmatically generated are now emerging, enter a system displayed text, where the consumer has to perform an additional task to prove the physical existence.

3.2 Authentication

Authentication is the process of assuring that a user has successfully proven to be a legitimate entity to utilize the system. Authentication is based on three aspects – something specific you know, something specific you have, and a characteristic unique to you. Something specific you know is normally a password the user has setup. The password is sometimes reinforced with a set of questions and answers, which also is a form of something you know. Something you have can be a string of characters randomly generated by a hardware device such as a pin or a physical object, a card that you swipe to get access. A physical characteristic unique to you is a biometric aspect such as your figure print or cornea. A strong authentication should include at least two of these three factors.

3.3 Authorization

Not every user needs access to all system resources, programs, processes, or data. The access to these resources should be controlled by an access criteria based upon the security policy. These criteria may vary from “No Access” as a default to “Need to Know”.

3.4 Audit

A sound security system must include an audit log of failed and successful access attempts. Repeated unsuccessful access attempts are of particular interest to avoid a potential security breach. The successful attempts are of critical importance if a breach has occurred to perform a detective analysis.

4 Web Application Security Threat Scenarios

A Web application is a software program that runs in a browser on the client machine [Wikipedia, 2013]. It is a convenient way to provide access to numerous clients without installing it on client systems. Since these programs run on the client systems the server side does not control the processes and the sessions they run under and can be a victim of injection, inadequate authentication and session management, improper exposure of data and functions, and week security configuration. On the server side there may be improper data and object exposure as a web application serves the data to the client.

Figure 1 shows the main components of a Web application environment. The engagement and the security aspects are described below for each component.

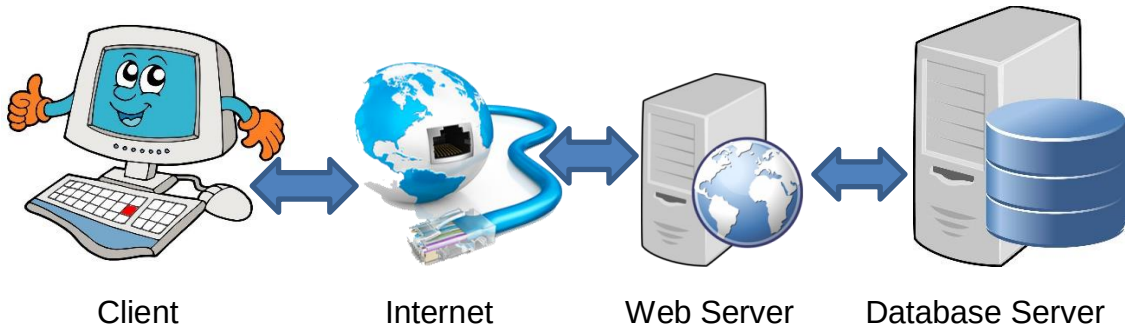

Figure 1: Web Application Environment

Client: The client system provides the run time environment for the Web application and is, therefore, responsible for managing the execution process which prominently includes, process and memory management. An application should not cause memory overflow and write data into unassigned memory. The potential security threats that arise from the scenario are remnants of code or data on the client system that can compromise the security of the client.

Internet and Web Sever: An internet networking stack has seven layers; Application, Presentation, Session, Transport, Network, Data Link, and Physical [Beal, 2015]. The session layer controls application to application communication, the communication between the client and the server, and the data transfer. The Web server manages the session ID for the multiple applications that are communicating with it. An inadequate session management can provide opportunity for an attacker to hijack a session and compromise the network security.

The transport layer establishes the protocol between the client and the server to assure the data is received in its entirety and its integrity is maintained. A security compromise at this level can cause data manipulation providing the attacker access to unauthorized data.

Software Practices: It is also meaningful to view the risk scenarios from the perspective these risk are originated. Inadequate identification and unauthorized access to processes, sessions, and data is a scenario that is internal to development. Adding to this list is also a lack of consideration of security configuration, use of known vulnerable software components, and improper exposure of sensitive data.

The Bad Guy: The attacker is always thinking about beating the system. The practice of injection, phishing, spoofing, social engineering, and fuzzing are here to stay and new once will be added to the list as the ‘Bad Guy” is always on the lookout.

5 OWASP Security Risks

The Open Web Application Security Project (OWASP) [OWASP, 2103] foundation is an independent non-profit entity mostly supported by volunteers. The OWASP foundation encourages creating application security programs that are compatible with the organization goals, culture, and technology. The OWASP Top 10 security risks were first released in 2003. This list was based on prevalence. The next major follow up release was in 2010 and the items were also based on risk and not just on prevalence. The 2013 version consolidated some items while it reprioritized others. This section describes major security risks as prioritized by OWASP in its 2013 release.

5.1 A1 – Injection

Injection is an attack technique which is common, easy to exploit, and can result in a serious security breach. In an injection an attacker normally inserts malicious data into client queries, OS commands, XML Parser, program arguments etc. [OWASP, 2013] which are executed by the server side. The most wide spread injection is SQL injection, which comes from an untrusted source such as a Web form. The malicious data entered in the Web form is used to construct a dynamic query that results into security breach. An attacker can exploit this technique to:

- Get access to data thereby compromising data confidentiality
- Modify data resulting into loss of data integrity
- Changing data authorization if stored in the database.

In its simplest form, the user can make the database throw an error by simply adding a single quote to the URL. The following example is taken form Web Application Injection Vulnerabilities [Couture, 2013] with a slight change to input. In the scenario, a Web form is requesting a name and the password from which it will create a SQL query which should like:

SELECT id FROM users WHERE username = 'Foo' AND password = 'QWERTY'

The attacker submits Username as 'OR 1=1 --/ and Password = anything. The server side will create a query by concatenating the values which looks like:

SELECT id FROM users WHERE username = 'OR 1=1 --/ AND password = 'anything'

Since the input field in this case, is not cleansed of escape characters, the double dash is interpreted by the parser that everything to right is a comment, and thus dropped. The parsed query that gets sent to the DB is: SELECT id FROM users WHERE username = '' OR 1=1

Which is interpreted as "Return all user ID's where the username is a null value, or 1=1" (which it always does). The string will always be true and thus dump all the stored user IDs.

The attacker can generate inputs like these by using automated tools like sqlmap [Moon, 2012] and can obtain a variety of information by embedding simple commands into an URL.

5.2 A2 - Broken Authentication and Session management

An article titled Survive the Deep End: PHP Security [Brady, 2014] has provided a detailed account of how risk can be attributed to insufficient transport layer security. A communication between a client browser and a Web server is maintained by a unique session using a session cookie on the client browser and the details with the Web server. It is critical that the privacy of both the client and the Web server, as well as the integrity of any data exchange, is maintained. The transport layer in the OSI stack is responsible to achieve this goal and its security is critical.

The negotiation about the encryption key between the two parties, client and the Web server, is carried out at the time of initiating a secure connection. An attacker can get in the middle and pretend to be the server thereby obtaining the encryption key. This is called the "Man in the Middle Attack" (MitM) attack. It is therefore essential that the initial negotiations happen with a trusted server that has an authentication ID. In addition, the session ID should be protected to avoid session hijacking. Brady recommends that the transport layer should be sufficiently secure between the client and the Web Server and the Web Server and a third party server and have proposed programmatic ways to achieve it. OWASP has cited some simple examples such as passing raw (un-encrypted) data in a URL, not properly logging out from a public client system, and improper hashing of database user password that can lead to this problem.

5.3 A3 – Cross Site Scripting (XSS)

Same-origin-policy is a concept of trust that states that if contents form one site are granted permission to access resources on the system then any content from that site will share those same permissions [Wikipedia, 2015]. The cross site scripting occurs when an attacker injects client side script into a web page. Due to the same origin policy, the script is considered to be from the trusted code as the Web server has already been granted permissions. If the attacker can get access to client session ID the effect may range from simple nuisance to a serious breach

There are two types of XSS attacks, non-persistent and persistent. A non-persistent attack occurs when the data provided by the client in http is immediately used by the Web server without sanitizing and an output is immediately generated. This leads to stealing of the client data. A persistent attack occurs when the non-sanitized data is stored in the database or any other locations such as blogs, forums and gets executed when other clients use the page. Here is a brief scenario of how XSS attack can take place:

User Gdguy always logs to site www.bgc.org using his login name and password and thus is a trusted user of the site. An attacker, Bdguy determines that the site www.bgc.org is vulnerable to XSS and creates a URL embedded with a JavaScript, http://www.gbc.org?q=cheaptickets<script%20src**=”http://attaker.org/steallogin.js”**>. The attacker sends this URL in an email about cheap tickets to the Gdguy. Gdguy does not pay attention to the URL and clicks on it. The JavaScript gets executed in the Gdguy’s browser and steals login credentials form the session cookie. With the stolen login credentials Bdguy now logs in as a trusted user and has control of Gdguy. Using an email spam the Bdguy can take control of www.bgc.org.

Detailed examples of session hijacking for both types of XSS vulnerabilities are provided on Wikipedia [Wikipedia 2015]. For a more comprehensive discussion on XSS refer to Survive the Deep End: PHP Security [Brady, 2014].

5.4 A8 – Cross Site Request Forgery (CSRF or XSRF)

In this attack an attacker lures a victim to click on an image and send unauthorized commands to the web server that the web server trusts once the session cookie has been established. Thus, the attacker exploits the trust of the web server in the client browser. An HTML image element is crafted that attracts victim’s attention and include the malicious state change request. The following example is taken from the OWASP Top 10.

<img src=http://example.com/app/transferfunds?amount=1500&destinationAccount=attackersAcct#
width=”0” height=”0” />

If the victim is already authenticated to example.com and visits any of the attacker’s sites the forged request will already include the session info, authorizing the attacker’s request. CSRF has not been widely reported but has been used for malicious bank activities.

5.5 A5 – Security Misconfiguration

Security misconfiguration is any part of the web server environment that is not adequately hardened with respect to security. This could be an update to the operating system or any open source or third party software, user accounts and passwords, and databases. The application developers should use and rely on a secure framework and keep it up to date. Hunt [Hunt, 2010] has discussed .Net framework scenarios specific to keeping it up to date with special aspects that include actions to take if a security flaw has been discovered in a framework, customizing error messages to not expose any system information, turn the debugging off, and encrypt sensitive configuration data.

6 Approach to Risk Mitigation

The exhaustive methods to protect yourself by building a security perimeter will not be cost effective. An organization should diligently address security based upon its goals and strategize its approach to risk mitigation to best achieve these goals. The following are some guidelines to make your environment less prone to risk.

6.1 Protect Your Most Valuable Assets First

As a basic principle, prioritize all your assets by value based upon your liability. When you are managing your customer data that might have legal consequences if compromised, it should be ranked higher on your priority list compared to the list of employees in your organization. Security breaches that involve customer data get significant public attention. The least vulnerable scenario is when there is no sensitive data to exploit for an attacker, in which case securing a Web application may not even be necessary. In the early days of the internet, the Web servers only served static pages with well thought out contents and there was no or only minimal security risk.

6.2 Information Classification

Information classification is an effective way to protect your environment internally. An organization can develop its own classification scheme or adopt one from the existing schemes depending upon it is requirements. Harris [Harris, 2013] has discussed effective access control models and methods to administer access control. The proper access control methods should be implemented so that business sensitive information is exposed only to those it is intended for.

6.3 Sound Development Processes

To deal with security today, we are using duct tape, band aids, and ointments that come in the form of firewalls, Intrusion Detection Systems (IDS), anti-virus software, and vulnerability scanners along with the security patches to the operating system. Just like achieving product quality is an integrated development activity and not an afterthought, so should be security. This concept can be best enforced if the Software Development Life Cycle (SDLC) proactively includes security activities and deliverables for each phase. Some early lifecycle activities should include; classification of sensitive data in the requirements phase, recognition of vulnerabilities and development of strategy in the design phase and security specific code reviews and testing in the implementation phase. The scope of security should be well understood and documented throughout the lifecycle to build a secure Web application. Harris [Harris, 2013] has provided a list of security activities and deliverables in each SDLC phase.

Although a Web application executes on the client system, it primarily serves the needs of the Web server by capturing critical data and providing the requested data back to the client. A secure Web server environment is essential for keeping both the server and the client trouble free. On the client side, since a Web application runs on the client system, the development methods should include practices that do not leave the client system vulnerable to attacks. Although the operating system can provide support for activities such as access control, the application itself must use robust development methods to protect the client system. The following discussion highlights some development practices that will result in a more secure operational environment for both the Web server and the client system.

6.3.1 Web Server Environment

A Web server environment is quite complex and includes numerous variables such as an operating system, web server software such as Apache, application development tools, databases, 3rd party support such as credit card support system, etc. Each of these variables requires dedicated attention for a successful operation. Listed below are some guidelines that will make your environment more secure:

Environment Upgrade/configuration – The web server environment should be up to date at all times. Any security updates to the operating system, development environment upgrades, as well as database updates should be up to par with the latest release. By default, all operating system and 3rd party software services may be turned on and configurations set; they should be reviewed and their status should be adjusted to match security requirements. The development environment tools may also need scrutiny as well. The network protocols are gateways for intrusion and only the necessary protocols should be enabled.

Authentication and Access Control – Making sure that only an authorized client is accessing the Web server is very critical. Section 3 describes the main elements of access control - identification and authentication being related to client's identity. Once the user provides her/his credentials they should be transported securely over the network using HTTPS or some other secure mechanism. In case of failed login attempts using UserID and Password as credentials, the error message should not reveal which component of the credentials (userID or Password) was incorrect. Only a limited number of failed attempts should be allowed in an applications that deal with sensitive information. There have been recent trends where additional authentication is performed via security questions. To avoid data corruption due to spam generated by an attacker the websites now use Captcha standard where a client enters a slightly distorted text after logging in.

Access control is very important even if you have a fool-proof authentication mechanism. Access control should enforce a mechanism that has been designed based upon business requirements. Both vertical segregation (where users at the different level have access to different data set) and horizontal segregation (where users at the same level have access to different data set) should be considered when designing the access control lists. This is critical when the access for administering the system is being considered. In addition, the permissions (setting a flag value to True or False) should not be controlled by the HTTP request which can be manipulated by the client.

Input Data Validation – In a Web application the data can come from a client, generated by a third party application, server's own environment, as well as from a database. Failure to validate the data could lead to serious vulnerabilities. In its guide, OWASP [Frenchi, 2014] has provided a detailed discussion on data validation including the characteristics the data should be validated against and validation strategies. The data validation should, at least, include a check for data integrity – any tempering during transport form the client to browser or 3rd party to browser. The data characteristics such as data boundaries, data type (numeric, alphanumeric etc.), sign of data (positive or negative) and validation against business rules, for example a price discount cannot be 100%, should also be adequately carried out. These checks should be made at the layer the data is generated. For example, a form data should be checked at the client side before sending it to the Web server. It may be too late from the security perspective if the bad data has already been sent to the server.

The data validation strategies recommended by OWASP range from accepting only known good data, sanitization such as eliminating HTML special characters, to void SQL injection discussed in Section 5.1, and take precaution to syntactically validate the data in the context of its grammar. The OWASP guide also provides steps to assess if you are vulnerable and how to protect yourself.

6.3.2 Session Management

A session is a period of time a client is interacting with the Web server including the time the client is not active. The web server creates a session upon the first HTTP request from the client and stores it locally. This session creation takes place for every single client connecting to the Web server and this could amount to thousands of client sessions. A session consists of Session ID along with other information such as login and password. The Session ID is unique to a client and is used to identify the client each time he/she makes a HTTP request in a session until the session expires.

The two aspects of the Session ID that the developers need to be diligent about are its generation and its transport back to the client. The algorithm used for the Session ID generation should be complex enough so that it cannot be reverse-engineered by an attacker. If an attacker is able to determine the Session ID, he/she can take over all the clients currently on the server and ultimately control the entire Web server.

Off-the-shelf Session ID generation algorithms are found to be prone to attack. If a new algorithm is implemented it should at least be better than an off-the-shelf algorithm. Highly sensitive applications such as the ones used in the banking environment, generate a new Session ID for each HTTP call. Additionally, the sessions are terminated either after a fixed time period or if there is no activity from the client for certain time.

The second aspect, session ID transport, is equally important. The session ID should be encrypted as it moves between the client and the web server. Once again, a strong encryption should be used for transportation and the encryption key should be well protected.

6.3.3 Programming Language

Web applications are created using a programming language such as Javascript. Programming languages have their shortcomings which can be exploited by an attacker if she/he can find out the application programming language. The developer needs to be aware that the programing language is not displayed in error messages or any other communication that is taking place between the client and the Web server. The first step an attacker takes is to map the application that includes attempts to identify the programming language and the development environment of an application. Some programming languages such as C, leave the remnant information in the memory due to stack overflow on the client system, which can be harmful if an attacker gets hold of this information. If a programming language has such a behavior, the developers need to be aware of it and must take adequate countermeasures.

6.3.4 Use of Proper Tools

Even though Web application security is a relatively new challenge, its compromise can cause serious harm to the consumer. There are enough resources available to proactively circumvent the problem. OWASP has a vast amount of information that provides guidelines and methodologies to build more sure applications. Stuttard and Pinto [Stuttard, 2015], in their Web Application Hackers Handbook, have discussed numerous tools such as Burp Proxy, WebScarab, and Paros, as well as methodologies that can be utilized for application testing and make testing more robust.

7 Conclusion

In general, the business of information security is complex and application security is an integrated component of it. On top of it, the “bad guy” enjoys the challenge of being ahead of the creators of applications and has gathered an arsenal of sophisticated tools. The industry has been fortifying the computing environment from outside by using a variety of tools mentioned earlier. But, once the attacker penetrates the security perimeter, she/he is capable of doing serious damage as evident from recent events. We have two choices; become numb to the sequence of events or take application security seriously as the consequences are dire. The software security should be an integral part of the SDLC and built-into development practices. What has been presented in this article is the tip of the iceberg. The information technology community has to come together to win the war against the bad guy.

8 Acknowledgements

The author would like to extend his gratitude to his ex-coworker Joshua McKinney of Nike Inc., for inspiring him and building his passion for Information Systems Security. He would also like to recognize Laura Bright and Satish Yogachar of McAfee, part of Intel Security, for their invaluable review efforts.

References

1. Reuters, http://www.reuters.com/article/2014/03/13/us-target-breach-idUSBREA2C14F20140313
2. NetworkWorld, http://www.networkworld.com/article/2861023/security0/worst-security-breaches-of-the-year-2014-sony-tops-the-list.html
3. Beal, Vangie, 7 Layers of OSI Model, http://www.webopedia.com/quick_ref/OSI_Layers.asp
4. Harris, Shon, All in One CISSP, 6th Edition, McGraw Hill, 2013
5. Wikipedia, http://en.wikipedia.org/wiki/Web_application
6. Couture, Erik , http://www.sans.org/reading-room/whitepapers/application/web-application-injection-vulnerabilities-web-app-039-s-security-nemesis-34247, 2013
7. Moon, Silver, Sqlmap tutorial for beginners – hacking with sql injection, http://www.binarytides.com/**,** 2012
8. Padraic, Brady, Survive the Deep End: PHP security, http://phpsecurity.readthedocs.org/en/latest/Injection-Attacks.html, © Copyright 2014
9. Wikipedia, https://en.wikipedia.org/wiki/Cross-site_scripting, 2015
10. Stuttard, Dafydd and Pinto, Marcus, The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws, 2011
11. Troy Hunt, OWASP Top 10 for .Net developers Part 6: Security Misconfiguration, http://www.troyhunt.com/2010/12/owasp-top-10-for-net-developers-part-6.html, 2010
12. Frenchi, https://github.com/OWASP/DevGuide/blob/master/old/OWASP%20Guide%203.0.docx , 2011