Proceedings of the International Conference on Inventive Research in Computing Applications (ICIRCA 2018)
IEEE Xplore Compliant Part Number:CFP18N67-ART; ISBN:978-1-5386-2456-2

Web Application Vulnerabilities – The Hacker's Treasure

Nirmal K
NIT Trichy, India
405913007@nitt.edu

B. Janet
Assistant Professor
NIT Trichy, India
janet@nitt.edu

R. Kumar
Sr. Consultant
Wipro Technologies, India
rajagopal.kumar4@gmail.com

Abstract – In today's online era, a web application is an integral part of every business. A web application may be a single page HTML website or a large web portal that offers various services on a web browser. There are many tools and methodologies that are relied upon to develop a web application. The development methodologies incorporate specialized frameworks, libraries in order to have the application more standardized and have it developed at a rapid pace to meet market demands. Web applications (web app) are hardened to mitigate security issues which are commonly referred as web application vulnerabilities. A web app security vulnerability is any kind of loop hole that allows an attacker to break into the web application to perform undesired actions on the target website. This may range from a cross site scripting (XSS) to vulnerabilities like Server Side Request Forger (SSRF) and its implications like XML External Entity (XXE). Though web applications are hardened to mitigate vulnerabilities, large scale web applications are still vulnerable post release in most cases. As a part of security research, critical vulnerabilities on large scale web applications were identified and the same were reported to the concerned security research team. The reporting was acknowledged and mitigated through appropriate channels. Common Vulnerabilities and Exposures (CVEs) were filed on Microsoft and CISCO products and the same were logged in National Vulnerability Database (NVD). Insights and tenets regarding web application and its vulnerabilities are highlighted in this manuscript.

***Index Terms* – Online Security, Web application vulnerability, Phishing, Pharming**

I. Introduction

Web Applications are the gateways for most of the business in today's cyber era. These web applications are developed using various technologies and programming languages. State of the art software technologies incorporate security framework within the Software Development Framework which enables software developers to incorporate security aspects to the web application. Considering the vast area of online security, it is practically impossible for an out of the box software development framework to address the entire security needs of a web application that is developed. A web application vulnerability might also refer to a loop hole in the implication of a business logic that is often not addressed by secure frameworks. An important phase of the software development life cycle addresses a major part of the security aspect of a web application. This is the web application vulnerability assessment/ Penetration testing phase. Two major methods are used in this phase. One is the manual web application software vulnerability assessment and the other is the automated software vulnerability assessment method. Tools aid in the web application vulnerability phase to identify common vulnerabilities that have sailed through the development phase. In spite of having secure web application development framework for software development and application vulnerability assessment phases for identification of web vulnerabilities, open security researches and responsible disclosure ventures have proved that web applications are susceptible for vulnerabilities that cannot be fully vetted merely by using secure frameworks. Once a web application is on the internet, it attracts the interest of a group of cyber Security researchers who constantly analyze the web application with a security eye. Their keep interest on having secure web applications result in the identification of security vulnerabilities on production environment which are then reported through appropriate channels for mitigation.

This being on the positive side, on the negative side, there are hackers who constantly crave for web application vulnerabilities to exploit and attack the target web application for personal, financial and political gains. Once a hacker identifies a loop hole on a target web application, he tends to exploit the same in a repudiated manner leaving no traces in server logs.

Hackers constantly hunt for web app vulnerabilities. On discovering a vulnerability on a web application, they do not report it as required. Most of the large scale web application companies and businesses lay down clear rules and policies on how a web application vulnerability should be handled and reported if found in their web portal. Failing to adhere to responsible vulnerability disclosure policies often violates the Terms and Conditions with regard to the usage of the web application portal. Triggering automated security vulnerability scan using tools on a live portal violates the vulnerability hunt policy in majority of the web applications. Web app administrators and owners should take extreme precautions before opening up their portal for public vulnerability hunting, bug bounty programs. Hackers tend to take advantage of such programs to evade criminal consequences of their illegal actions on web portals. Web Applications are written in both, weak typed and strong typed languages [2].

II. VULNERABILITY ASSESSMENT VS EXPLOITATION

For security research on a web application, researchers are encouraged to conduct web application vulnerability assessment within the frame of the security vulnerability disclosure guidelines laid out. Exploitation of the loop hole as a part of security research is often illegal and might result in negative consequences. Vulnerability assessment refers to the procedure followed in order to identify the security weakness on a target web portal. Exploitation is having the security weakness leveraged to accomplish actions that are not permitted by the web application usage policy.

A. *Example –SQL Injection terms of assessment vs exploitation:*

Vulnerability assessment on a parameter name might include the following.

&name = X

&name = X' AND 1=1—

&name = X' AND 0=1—

&name = <blank>

The above Boolean type of blind SQL Injection statements will assist in identification of SQL Injection which can be termed as Vulnerability Assessment on name parameter on the target portal.

Vulnerability exploitation on a parameter name might include the following. In this case, there is a clear indication that there is an attempt to steal information from the database by exploiting SQL injection.

```
&name = Daniel union all SELECT
NULL,NULL,COLUMN_NAME,NULL,NULL FROM
INFORMATION_SCHEMA.COLUMNS WHERE
TABLE_NAME = 'staff_password'--
```

III. WEB VULNERABILITY EXPLOITATION - THE HACKER's TARGET

Hackers assess a target portal for web vulnerabilities under the banner of vulnerability assessment or through repudiated methods where their identity is hidden or kept clean. Once a vulnerability is identified, they often target for a repudiated way of exploitation of the assessed vulnerability. Leveraging the security loop hole in the web application, they exploit the weakness of the web application for financial gains in most scenarios. Common web application vulnerabilities and their exploitation aspects are discussed. Session sniffing, Session prediction, Session Fixation, HTTP Response splitting, etc. are some of the prominent session based vulnerabilities [6]. Leaky or Guessable Session Identifiers have been a major factor for session related web vulnerabilities [8]

Vulnerability exploitation may result in undesired consequences for the web portal users. Hackers use the exploited vulnerability in various ways which include direct and indirect ways of exploitation.

Examples of common web application vulnerabilities includes SQL injection, XSS, CSRF.

A. *Cross Site Scripting (XSS):*

An XSS vulnerability may be exploited directly or indirectly. Hackers steal sensitive information which is not limited to cookie data by exploiting XSS vulnerability. In general, there are three kinds of XSS.

i. Reflected Cross Site Scripting
ii. Stored/ Persistent Cross Site Scripting
iii. DOM Based Cross Site Scripting

The fourth category is Universal XSS (UXSS) which is less common where the client browser or a browser extension is vulnerable for script injection.

Statistics and security research proves that XSS vulnerability is very much prevalent in today's web applications though they are hardened to mitigate the vulnerability.

An instance of XSS was reported on CISCO ACNS [1]

Hackers target web applications that are vulnerable for XSS and inject scripts onto the vulnerable web application. When a victim lands on the web page, the injected script executes in the context of the victim. The victim cab be vulnerable for session cookie stealing, key logging, etc. as a result of the attack.

B. *Cross Site request Forgery:*

Cross Site Request Forgery (CSRF) web vulnerability is exploited by hackers to perform undesired actions on user's authenticated sessions. A CSRF attack can be carried out once a user is authenticated to a legitimate target web portal. In today's online era, most of the websites and web portals deal with authentication mechanisms to protect user's data. Exploiting CSRF, an attacker will be able to craft and trigger a HTTP request on behalf of the victim.

Hackers target web applications that are vulnerable for CSRF for various operations, including complete account compromise.

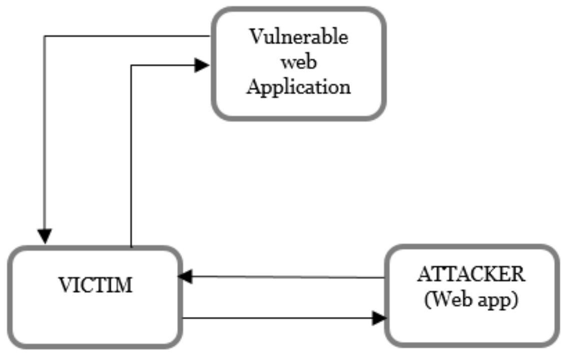

Fig 1. Working of a CSRF attack

To exploit a CSRF attack, an attacker tricks the victim to visit a malicious web application or a website that has been hosted by the attacker. On visiting the malicious web application, on behalf of the victim, leveraging the authenticated session cookies on the victim's web browser of the legitimate target page, the attacker triggers a HTTP request to the vulnerable web application from the victim's browser. The vulnerable web application responds to the request normally not knowing that it was triggered by the attacker on behalf of the victim.

IV. BUILDING A SECURE WEB APPLICAITON

Security research in the area of web application vulnerabilities have paved way for several state of the art software development frameworks which incorporates security as a part of the framework. One such framework is Spring security framework which is a powerful and customizable security framework. Research and practical analysis on web application vulnerability proves that, though a product is developed using a secure framework, it can be still vulnerable for a wide range of web app vulnerabilities. Reasons for this includes the fact that secure frameworks are not always fool proof and it depends on the way in which it is customized and used. Security education and awareness model is booming in today's software industry where web application developers are trained regarding the security aspects. Hands on lab exercises are used for computer security education [7]

Apart from secure frameworks, training and other methods, it is evident that security is everybody's responsibility during the development process. During the Software Development Lifecycle, from design to delivery, every aspect is to be viewed from a security perspective to have the end product hardened to the maximum level.

V. SECURE CODE REVIEW AND PENETRATION TESTING

Secure Code review and penetration testing of web applications are two important security phases in software development lifecycle that helps to identify and mitigate major web application vulnerabilities. Secure code review is the process of reviewing the raw source code for potential software vulnerabilities. Hybrid program analysis and machine learning techniques are also used to predict web application vulnerability [5]

Example of a Secure Code Review instance:

During a secure code review, it will be possible to identify vulnerabilities including instances of potential SQL Injection by evaluating the raw source code.

Server side code for a login box which is potentially vulnerable for SQL Injection

```
userName = getRequestString("UserName");
userPwd = getRequestString("UserPass");

sql = "SELECT * FROM Users WHERE username ='" +
userName + "' AND Password ='" + userPwd + "'"
```

On reviewing the above code snippet, a security code reviewer will be able to infer that the code might be vulnerable for SQL Injection. An attacker can insert the following malicious string into the username and password box in order to access usernames and passwords from the database.

Malicious string: " or ""="

The resulting SQL query will be:

```
SELECT * FROM Users where username ="" or ""=""
AND Password ="" or ""=""
```

The use of parameterized query is a proven method to mitigate SQL injection attacks on web applications [9]

Penetration testing phase, also known as security assessment is a key phase during the SDLC process. During a web application penetration testing phase, the web application is subjected to fuzzing and scanning where various combinations of malicious payloads are injected into the HTTP request stream and their corresponding response is analyzed. Vulnerabilities that have escaped and sailed through the development process can be identified in this phase. Further, organizations like Microsoft, Google, Facebook etc. have their production environment open for security researchers to conduct security testing within their policy to identify vulnerabilities. Injection of realistic attacks in production Web Applications have proved to be a successful methodology in to have the application hardened [3]. Black box penetration testing is also used to detect vulnerabilities in web services [4] alongside web applications.

V. TENETS IN LIEU WEB APP SECURITY

i) A software developed using a secure framework might not be 100% secure. Careful evaluation of the security specs along the SDLC (software development life cycle) process is required.

ii) The general specs of a SDLC involves phases from Requirement gathering to product release. Security should be embedded in each phase of the web application development.

iii) Web application developers should be aware of vulnerabilities and their consequences prior to software development

iv) Web application security should not be viewed as the job of InfoSec professionals. Every software developer should be involved.

v) Post release, the application should be iteratively evaluated for vulnerabilities using various channels including Responsible Disclosure – Web app vulnerability assessment should not be 'point in time' based.

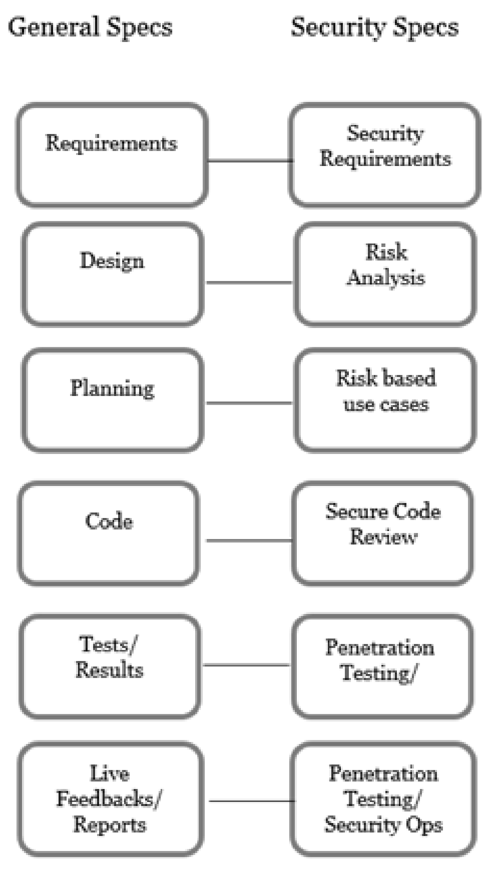

Figure 2. Security during SDLC of web application

IX. WEB APPLICATION SECURITY ACROSS DOMAINS

Web Application vulnerabilities are not restricted or specific to a particular domain or industry. This threat spans across every industry. It is a common misconception that web vulnerabilities are a threat only to banking and other financial domains. Web vulnerabilities can be disastrous for all domains ranging from banking, social media to the smallest domain one can imagine of. Likewise, there is no specific category of vulnerability that can be associated to a particular domain. Vulnerabilities span across all domains and the common vulnerabilities across major domains was released by Whitehat Security.

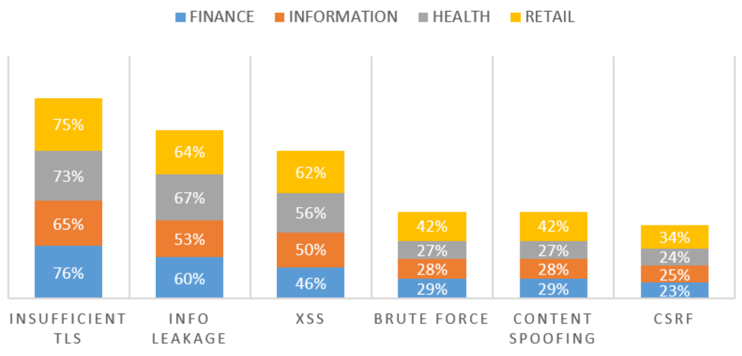

Figure 3. Likelihood across Major Industry with regard to common vulnerabilities [11]

Statistics also shows that the primary driver for resolving vulnerabilities was to avoid risk that may arise due to the exploitation of the vulnerability.

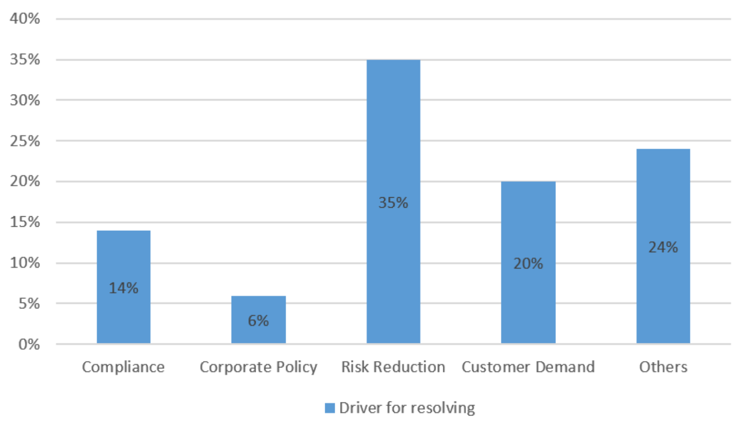

Figure 4. Driver for resolving web vulnerabilities [11]

VII. CONCLUSION

With the rise in need of web application for every business in today's cyber era, the need for hardening web application in terms of security is very critical. A vulnerable web application affects not only the business, but end users whose information will be subjected to risk. Hackers are always on the look-out for the presence of software vulnerabilities on the web applications which can be leveraged for personal, financial or political gain. It is the responsibility of every software professional to have in-depth understanding of web application vulnerabilities and their practical implications. At an organizational level, securing products from attackers who are on the constant look out for web application vulnerabilities is a key task. Programs including responsible disclosure and security research initiatives have proved to be a successful strategy in securing a product post release. CVE 2015-0774 and CVE 2016-0031 were filed on CISCO and Microsoft products as a part responsible disclosure. The reporting of CVE 2016-0031 was placed in Microsoft's Honor Rolls considering the criticality and the end user impact.

VIII. REFERENCES

[1] 'CISCO ACN returns Cross Site Scripting Vulnerability', site: https://nvd.nist.gov/vuln/detail/CVE-2015-0774

[2] Jose´ Fonseca, Nuno Seixas, Marco Vieira, and Henrique Madeira, "Analysis of Field Data on Web Security Vulnerabilities", IEEE Transaction on dependable and Secure Computing, VOL. 11, NO. 2, MARCH/APRIL 2014

[3] Jose Fonseca, Marco Vieira, and Henrique Madeira, "Evaluation of Web Security Mechanisms Using Vulnerability & Attack Injection", Dependable and Secure Computing, IEEE Transactions (Volume:11 , Issue: 5)

[4] Palma Salas, M.I.; Martins, E. "A Black-Box Approach to Detect Vulnerabilities in Web Services Using Penetration Testing", Latin America Transactions, IEEE (Revista IEEE America Latina) Year: 2015, Volume: 13, Issue: 3 Pages: 707 – 712

[5] Lwin Khin Shar; Briand, L.C.; Hee Beng Kuan Tan, "Web Application Vulnerability Prediction Using Hybrid Program Analysis and Machine Learning", Dependable and Secure Computing, IEEE Transactions Year: 2015, Volume: 12, Issue: 6 Pages: 688 – 707

[6] Vlsaggio, C.A.; Blasio, L.C., "Session management vulnerabilities in today's web" Security & Privacy IEEE Year: 2010, Volume: 8, Issue: 5 Pages: 48 - 56

[7] Wenliang Du "SEED: Hands-On Lab Exercises for Computer Security Education", Security & Privacy, IEEE Year: 2011, Volume: 9, Issue: 5 Pages: 70 – 73

[8] 'Farrell, S, "Leaky or Guessable Session Identifiers", Internet Computing, IEEE Year: 2011, Volume: 15, Issue: 1 Pages: 88 – 91

[9] Lwin Khin Shar and Hee Beng Kuan Tan "Defeating SQL Injection", Published by the IEEE Computer Society, Year: 2013, Volume: 46, Issue: 3 Pages: 69 – 77

[10] Grobauer, B.; Walloschek, T.; Stocker, E., "Understanding Cloud Computing Vulnerabilities" Security & Privacy, IEEE Year: 2011, Volume: 9, Issue: 2 Pages: 50 – 57

[11] "Web Security Statistics Reprot 2015" Whitehat Security, Site: https://info.whitehatsec.com/rs/whitehatsecurity/images/2015-Stats-Report.pdf