© 2017 IJSRST | Volume 3 | Issue 7 | Print ISSN: 2395-6011 | Online ISSN: 2395-602X
Themed Section: Science and Technology

Major Web Application Threats for Data Privacy & Security – Detection, Analysis and Mitigation Strategies

Varun M Deshpande*[1], Dr. Mydhili K. Nair[2], Dhrumil Shah[3]
*[1] PhD Student, Department of C.S.E., Jain University, Bangalore, India
[2] Professor, Department. of I.S.E., M S Ramaiah Institute of Technology, Bangalore, India
[3]Application Security Specialist, Bangalore, India

ABSTRACT

In the context of information security, privacy and data security are inseparable, interdependent and complement each other. This is truer in social networking and e-commerce where user's personal data including financial transaction data is at stake. Web application security threats have posed several challenges to ensuring data security of any web application hosted on cloud. These threats have been evolving in severity and the potential impact that it causes to service provider and the user's personal data that it hosts. Current work is an effort to educate the readers about major vulnerabilities that exist among security threats listed as part of Open Web Application Security Project's (OWASP) top ten web security threats. We provide detailed guidelines on how to detect, and analyse these vulnerabilities using tools such as Burp Suite. Recommendations and best practices for developing a secure development life cycle and following secure coding practices are discussed at length to empower developers to mitigate and avoid these vulnerabilities in their application at different stages of software development. This work is a timely and technically informative reminder for all the service providers to build trustable solutions for secure cloud based services and move towards trusted computing and to ensure user data's privacy and security.

Keywords: Privacy, data security, digital identity, OWASP, web application threats

I. INTRODUCTION

A. Privacy and Security Complementary concepts

In the world of information security, terms *privacy* and *security* are often used on behalf of each other. Although it is technically not the same, they are hugely inter-dependent and form 2 faces of the coin named "*information security*". To illustrate this thought, let us consider a service provider which hosts large amounts of its consumer's personally identifiable information, such as financial transaction records and other personal information which is ought to be kept in private and in reasonable safety. Suppose, an external malicious agent can hack into the service provider's servers and able to gain access to all these user data records. He can then use it for gaining unfair financial benefit or even cause harm to the users and the service provider. Private user information which had to be secured, is now an asset of the hacker. Hence, ***data security is the door keeper of the fortress of data privacy***. If the service provider's data security is breached, the private information which is stored in the server is vulnerable to data theft and loss of privacy.

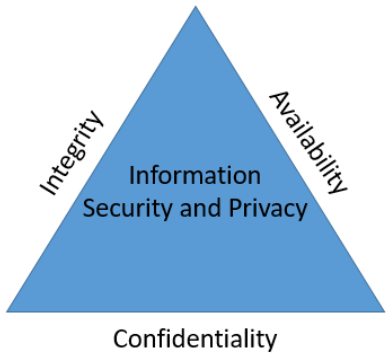
Loss of privacy due to data security breach has happened several times in recent past as highlighted by Varun M Deshpande et al. [6]. Yahoo (2016) announced that about 500 million of its user records were compromised. About 360 million records from MySpace (2016) was hacked by hackers. One concern with soft data is that, even after data is stolen, the data still exits on the hard drive making it extremely difficult to proactively detect a data theft. Therefore, in some cases, the data theft is detected several months or years after the actual breach. For example, Dropbox recently discovered that about 70 million user records were compromised years ago, back in 2012. Even Linkedin reported in 2016 that it had underwent a data breach in the year 2012, when 6.5 million passwords were stolen.

In each of these cases, certain lapses in data security practices of the companies had resulted in such massive scale of data leak and compromised user data. Thus, the companies find themselves in law suits and end up paying fines to government and/or to users whose data were leaked. Service provider eventually ends up paying a price for poor application security for years on.

Another parallel that we can draw from, to understand the relationship of privacy and security, is object oriented programming principles. From the days when C++ was first introduced, we have been familiar with concept of data hiding through encapsulation. Any object's data can be categorized into different levels of privacy required to be enforced on it. "public" and "private" access specifiers have been used to implement the same in most object-oriented programing languages. One of the main motivation for adopting such a strategy of data privacy is to facilitate data security of the overall application. If the external entities are not able to access the private data of the application, the possibility of causing harm to the application is greatly reduced. Hence, ***enforcing data privacy acts as an enabler for ensuring data security of an application.***

With above two illustrations, we understand that privacy and security are complementing concepts in the broad area of information security. *Data security is required to safeguard data privacy. Data privacy is necessary to ensure data security.*

B. CIA Triad & Security Design Principles in Information Security



Mark Rhodes-Ousley [8], in their book "Information Security", highlight the importance of information in current data driven era. They highlight that being an important asset, amount of information and how they protect and use it, differentiates between success of one company to another. Hence, information security is very important aspect in running a company successfully.

Over the past decades, computer scientists have come up with various models and best practices that ensure information security. In a context setting research paper published in 1974, Saltzer et. al. [10] proposed 11 security design principles for any computer based system. Even after 30+ years, their inputs are still being respected and followed. Some of security design principles proposed in the paper were: ***Least privilege, Separation of duties, Défense in depth, Fail Safe, Economy of mechanism, Complete mediation, Open design, least common mechanism, Psychological acceptability, Weakest link and Leveraging existing components.***

Adopting these principles while building web applications would ensure security of the service. Any discussion related to information related to information security is incomplete without mention of its 3 pillars – Confidentiality, Integrity and Availability. These three are popularly termed as "***The CIA Triad***" of Information Security as shown in figure 1. As we established that security and privacy are complimentary to each other, we may term that CIA triad are pillars for information security and privacy as well. Ronald L. Krutz et. al [7], in their book, discuss briefly about CIA triad while discussing about risks involved in providing a secure cloud based services. They regard CIA triad as fundamental tenets of information security.

When we study some case studies with respect to data privacy breaches, data access and misuse of personal data could be intentional or unintentional. Example for unintentional data access and misuse is Google being fined by several states in USA in 2013 for (unintentionally) scanning emails and other personal information from unsecure Wi-Fi routers using StreetView cars. Example for intentional data access and misuse is when AT & T was fined in 2015 for actions of some of their employees, illegally accessing and selling user data to third parties [11]. In this context, ***Confidentiality*** deals with proactive prevention of intentional and unintentional unauthorized disclosure of data which is supposed to be private with restricted access. Data encapsulation, encryption, network authentication etc. are some of the practical examples of confidentiality principle being applied in real world scenarios.

Integrity by principle is an assurance that the information which is sent by sender is the same when compared to the message received by the receiver. Effectively, it is a guarantee that the data which travels through the communication channels is not modified during its journey and it reaches the destination without altercations using *man in middle attacks*, which we will discuss later in the paper. Firewalls, intrusion detection and prevention systems are used to implement integrity in cloud based applications.

Any application, to be successfully accepted by the user community needs to be reliable and stable. The application needs to be accessible so that the authorized users can access the services over the internet when necessary and connection lines be open at all time. This tenet of cloud services is known as **Availability**. This is the third pillar of information security and privacy. Fault tolerant architecture, fail safe mechanism needs to be put in place to ensure highest availability of a cloud applications. Availability, though not coming under direct purview of web security, still is an important topic to be discussed and to take measures to uphold high availability and fight against distributed denial of service (DDoS) attacks which have become common mode of brute force attack these days.

C. Web Application Threats – A brief Introduction

Current applications paradigm includes web, mobile and cloud platforms. With rapid upgradation of technology in recent times, software organizations are having to keep on updating their technology stack.

Web Applications, being information collectors and disseminators, are the biggest source of information for customers and/or users and its usage is increasing exponentially. One must ensure that applications are available 24/7 and provide relevant requested information. Complexity of application is increasing to facilitate more feature making it more vulnerable by increased attack surface. With evolution in technology and availability of computing power, attackers have become highly intelligent and they can perform attacks with knowledge of application behaviour, network and tools.

With the knowledge of software development and hosting network, attackers create tools and customize the attacks based on the application to exploit security vulnerabilities, rules and policies to achieve their desired motive.

Security is never ending game. As soon as latest threat is mitigated, new threat emerges. To ensure that security is maintained, we need to implement security at different levels to achieve defence in depth. Application level attack rely on complex input scenarios. Attackers manipulates different application inputs and obtain sensitive information, bypassing network defence systems. In early 2010, SQL Injection type of application attack were common compared to malware distribution and DDoS. Other than this, OWASP (Open Web Application Security Project) Top Ten security threats are in the top list of attackers.

Web attacks are evolved to target non-technical people using social engineering attacks. Also, increase in usefulness and popularity of 'Smart' devices have become integral to the modern way of life which carry a huge risk. These devices act as a wallet; authentication means and communication portal. Access to mobile devices has significant control over the end user's life. Profile provided on social networking platform reveals lot of personal information which helps attackers in crafting targeted attacks using phishing (fraudulent practice of impersonating reputable companies by sending emails) and vishing (fraudulent practice of impersonating reputable companies by calling or sending voice messages) techniques to trick users to provide financial and personal information. To support different devices Web API's (Application Programing Interface) are exposed and that deals with all the communication between client and server. Implementing security in Web API's is challenging and sometimes application owners exposes whole lot of information to attackers.

Use of BYOD (Bring Your Own Device) has increased attack surface for an attacker looking to compromise enterprise security safeguards. Attackers can use proxy of a BYOD device to peep in to organizations network, systems and applications.

The addition of online access to sensors (like video cameras, surveillance systems) attackers have newer ways in compromising privacy, monitoring victims remotely and organizations to espionage and other risks.

Threats are not limited to application only; it can affect network and infrastructure level also. In recent times,

there were ransomware's in action which encrypts all your data and system and then asks for Ransom to recover of data. We will discuss this in more detail, later in the paper

D. OWASP – Top Ten Project

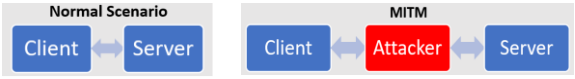
In any given field of knowledge engineering, there are several stake holders such as the industry leaders – service providers, academic researchers, subject matter experts from user community. For dissemination of knowledge on latest developments and proposals of best practices, academic publications and industry white papers alone is not a holistic solution for accessing unbiased, technically relevant, accepted as an open standard across all the stake holders of the knowledge community. It is recommended to have an independent body which publishes unbiased reports on the current happenings on the industry and acts as a watch dog for alerting the knowledge community from potential pitfalls. It also needs to guide all by publishing best practices for different aspects in the knowledge industry.

In "*Web Application Security*", we have such a community that provides open source and extensive documentation on standards, tools, methodologies, best practices, potential threats etc. The name of this community is **OWASP** (*Open Web Application Security Project*). This initiative was started in year 2001. OWASP runs as a non-profit organization which deals with helping the stakeholders of the technology space to improve the security of their software including cloud based web applications. It is an open forum where people can contribute and stay connected in the community.

One of the popular initiatives taken up by OWASP is the publication of "***OWASP Top 10 Most Critical Web Application Security Risks***" [12]. Popularly known as the "***OWASP Top Ten***", it is regarded as a highly dependable and powerful awareness report on the top web application security threats. The report lists the top ten vulnerabilities that any web application has faced in last 3-4 years of time. It sends out a formal questionnaire which is filled by stake holders of web application security, usually web based companies and researchers. The data is analyzed and the trend is discussed in open consortiums and the top ten web vulnerabilities are published. OWASP Top Ten web vulnerabilities have been published since 2003 and currently 2017 list is under discussion. For any changes done for application, of any new application being built; checking and safeguarding it from these vulnerabilities goes a long way in securing the web services. Top Ten has become a default standard in web application security industry for understanding top web security threats.

E. Man In Middle Attacks – A brief Introduction

In *man-in-the-middle* (MITM, MitM, MiM) attacks, an attacker intercepts communication between two parties. Attacker then impersonates both the parties and gains access to information of impacted parties. For example, an attacker is sitting between client and server to intercept, capture and modify any data client sends to server and vice versa, such as login credentials, financial information, transactions, conversations or transfer of data.



Modern MITM attacks involve malware distribution which provides attacker with access to victim's web browser and the communication data. This kind of attacks are called as *Man-in-the-browser*. An attacker can also exploit vulnerabilities of a wireless router's default or weak security configuration. For example, a malicious router (evil twin or malicious hotspot), can be setup in a public setup such a café or hotel that provides free Wi-Fi to intercept information communicated through the router.

Other ways of MITM includes Email Hijacking, Wi-fi eavesdropping, Session Hijacking, Sniffing, Spoofing (ARP – Address Resolution Protocol or DHCP – Dynamic Host Configuration Protocol or DNS – Domain Name System), Spanning Tree Protocol (STP) mangling, port stealing, traffic tunnelling and route mangling.

There are various ways to prevent MITM such as Authentication Certificates, HTTP Strict Transport Security (HSTS) on HTTPS, S/MIME, SSL/TLS Certificates, Use of Virtual Private Networks (VPN) and updating default configuration of devices/systems

F. Need for Vulnerability Assessment and Early Detection

No web application is free from vulnerabilities. It is better to be safe than being sorry later. Early detection and mitigation can save the company millions of dollars.

We have highlighted several examples of data theft and subsequent litigations that affected big names in the industry in recent past. "*A stich in time, saves nine*", says a popular English proverb. So, it is better to ensure that web application security is dealt with right from the design and conceptualization phase itself rather than building security layer on top of the application.

The level and sophistication of the web threats are constantly evolving. Hence, there is a very genuine need to ensure that, lack of web application security doesn't hinder the reputation of the service provider and make it vulnerable from attacks. Implementing the security design principles while developing any new system or updating an existing one is necessary step to ensure software security. CIA triad should always be kept in mind when efficient information safety measures are put into place. Best practices documents such as OWASP Top Ten can be taken as reference for building security defence in web applications.

While building web applications, at each iteration, security testing can help in identifying any loop holes in the system and the vulnerabilities can be addressed at its infant stage itself. It might be very costly to perform security patching at later stage. We are seeing several companies updating their encryption algorithms from MD5 or SHA1 to SH256 or higher algorithm. This is done to mitigate any future data security related concerns. There are several application security gauging tools available in market which can help in vulnerability assessment of web applications. We look at such tools, later in this paper.

II. EVOLUTION of OWASP TOP 10 WEB THREATS

A. Introduction

We have discussed briefly about OWASP Top Ten project in the previous sections. It is a document created and maintained by OWASP group which lists the most frequently occurring and most dangerous vulnerabilities in cloud based web applications [12]. In current section, we specifically consider 3 such OWASP Top Ten projects – 2010, 2013 and 2017. 2017 project is currently under discussion and list is not finalized yet. We will provide latest update regarding the ongoing discussions. OWASP's primary goal of publishing this report is to educate all the stake holders of developing secure software and web applications such as architects, developers, designers, management, testers etc. OWASP wants to help them realize the consequences of web application threats. The report also provides inputs and remediation measures to mitigate the web application vulnerabilities.

OWASP guides and alerts the readers not to be complacent and stop at only the top ten vulnerabilities. They recognized that there are hundreds of vulnerabilities that need to be addressed. Top Ten threats should be considered as a guiding light to start a successful security program in the organization.

B. OWASP Top Ten - 2010

All the major sectors such as defence, healthcare, finance, energy, commerce etc. are moving online. This makes the requirement of web application security, more important and relevant for discussion. OWASP Top Ten-2010 [5] was a major update when compared with its predecessor reports. It focused on the top ten most critical web application security risks at that time. The previous versions of OWASP Top Ten reports were designed to highlight only the most frequently occurring web vulnerabilities. However, in 2010, OWASP followed an updated model to assess the potential risks and business impact of such vulnerability exploitation may cause. As the risk rating methodology was upgraded, the report became a holistic and credible report. OWASP Top Ten – 2010 list contained below vulnerabilities.

A1 – Injection, A2 – Cross-Site Scripting (XSS), A3 – Broken Authentication and Session Management, A4 – Insecure Direct Object References, A5 – Cross-Site Request Forgery (CSRF), A6 – Security Misconfiguration, A7 – Insecure Cryptographic Storage, A8 – Failure to Restrict URL Access, A9 – Insufficient Transport Layer Protection, A10 – Un validated Redirects and Forwards

Some of the major vulnerabilities would be discussed in detail in next section. When compared with OWASP Top Ten list from 2007, *A6- Security Misconfiguration* and *A10 – Un validated redirects and forwards* were elevated. Previous risks which were de prioritized were *Malicious file executions* and *Information leakage and improper error handling*.

C. Updated OWASP Top Ten - 2013

Three years after the previous report, OWASP published an updated report in 2013 [4]. This version of OWASP Top Ten aimed to broaden some of the vulnerabilities to include common and more important vulnerabilities compared to 2010 version. It also reordered some of the threats based on risk rating reports. Data regarding security vulnerabilities from over 500,000 vulnerabilities which were collected from hundreds of organizations to arrive at the report. The updated OWASP Top Ten – 2013 is listed here: ***A1 – Injection, A2 – Broken Authentication and Session Management, A3 – Cross-Site Scripting (XSS), A4 – Insecure Direct Object References, A5 – Security Misconfiguration, A6 – Sensitive Data Exposure, A7 – Missing Function Level Access Control, A8 – Cross-Site Request Forgery (CSRF) , A9 – Using Known Vulnerable Components, A10 – Un validated Redirects and Forwards.***

The threat landscape continuously evolves due to new attack strategies of hackers and vulnerabilities introduced in new technologies as well as newly discovered vulnerabilities in legacy technology. *A2 - Broken authentication and session management* vulnerabilities were moved up the chain mostly because there we more sophisticated attacks on this aspect. *A8- Cross Site Script Forgery (CSRF)* was moved down from 3 to 8th spot. This may be because companies have effectively protected themselves from this vulnerability. *A7 – Insecure Cryptographic Storage* and *A8 – Failure to Restrict URL Access* were merged into single vulnerability: *A6 Sensitive Data exposure***.** 2010's *A8 - Failure to Restrict URL Access* vulnerability was renamed ad broadened as *A7 – Missing Function Level Access Control.* 2010 A7 and A9 were merged into *A6 - Sensitive data exposure.* A new vulnerability *A9- Using known vulnerable components* made it into the Top Ten List.

D. Update on OWASP Top Ten - 2017

Usually, OWASP Top Ten – web application security risk report gets published every 3 years. After 2013, next version was expected on 2016. However, this version has been an exception for this. OWASP Top Ten - 2017 report (RC1) was published in April 2017 [3] and opened for discussion. Based on constructive feedback, the RC1 was rejected and the updated OWASP 2017 report is due. Currently, discussions are ongoing and update is expected very soon. The list which was proposed in RC1 were as follows: ***A1 – Injection, A2 – Broken Authentication and Session Management, A3 – Cross-Site Scripting (XSS), A4 – Broken Access Control, A5 – Security Misconfiguration, A6 – Sensitive Data Exposure, A7 – Insufficient Attack Protection, A8 – Cross-Site Request Forgery (CSRF), A9 – Using Components with Known Vulnerabilities, A10 – Under protected APIs (NEW)***.

E. Other notable Web Threats

Considering different web threats, few are not fully related to web but those are caused due to the vulnerabilities in the core libraries and impacted major web applications. In this section, we discuss few such web threats detected in recent past by security community.

Heartbleed -a serious and a legacy defect in the OpenSSL cryptographic library introduced due to improper input validation during implementation of the TLS Heartbeat extension [16]. It allowed anyone on the internet to read memory contents of the system, exposing the security keys used for encryption making the secure transactions unsecure. This was subsequently fixed in the later versions (CVE-2014-0160) of OpenSSL libraries.

SSL Poodle - a man-in-the-middle vulnerability that uses protocol version negotiation feature available in SSL/TLS to force the use of SSL 3.0 (lower version of SSL) and then decrypts targeted content within the SSL session. [17] are the example of such attacks.

Recent ransomware attacks: **WannaCry** – was spread using **EternalBlue**, a vulnerability in Windows Server Message Block (SMB) protocol [18]. When run on a client, the ransomware reads the 'kill switch' domain name; if the domain name is not found, then the ransomware will encrypt the computer's data, then uses the SMB vulnerability to span out to random computers over the network and internet.

Petya - a ransomware that affects computer's master boot record (MBR), overwriting the Windows bootloader, and restarting it. On the next startup, the malware payload is executed, which encrypts the Master File Table of the NTFS file system, it then displays the

ransom message to the user demanding that payment be made in Bitcoin. [19]

Locky - a highly active ransomware being distributed by email with an attached Microsoft Word document containing malicious executables. If the user falls for this scam and opens the attachment, n execution, the malware downloads the encryption Trojan, which will then encrypt computer files configured in the Trojan.

III. ANALYSIS of WEB THREATS

A. Introduction

Based on recent trends, we will discuss on major three Web Threats (Injection, Cross-site Scripting, Cross-site script forgery) in detail and its countermeasures. This section will provide details on attack, attack scenarios and its remediation.

B. Injection Vulnerabilities

Injection vulnerability is when an application sends untrusted data to an interpreter and received data is interpreted as commands instead of data. There are different types of injection flaws such as SQL, LDAP, XML, XPATH, OS commands, SMTP Headers, HTML etc. Out of these injection flaws, *SQL injection* is widely spread and exposed vulnerability in any application. Injection vulnerabilities occur when

- Application receives unexpected data from any source
- Data is directly used to construct a dynamic Query

Let's understand this by example of SQL Injection: Basic SQL statement to validate username and password in which "***Request.Form***" accepts user-supplied data is **string query = "select id from login where username='" + Request.Form("username") + "' and password='" + Request.Form("password") + "'";**
The way SQL Query is constructed; the attacker can supply malicious input to make the original SQL statement execute additional actions of the attacker's choice. In the above SQL Statement user enters **username="*admin' or 1=1--*" and password="*P@ssword123*"**

So now query will be

select id from login where username='admin' or 1=1 --' and password='P@ssword123'

Here malicious user-supplied data changes the logic of the query and returns true due to "or 1=1". "--" considered as comment thus "' and password='P@ssword123'" part of query will not get executed by SQL Interpreter. Execution of this query by SQL Interpreter always returns the first record of the login table (in most of the cases it will be admin user) and attacker will be able to login into application with credentials of an admin.
In basic SQL Injection, application retrieves and displays relevant content in the response. So, by comparing output for normal input and malicious input, one can identify its existence. More advanced version of SQL Injection is *Blind SQL Injection*, which requires more knowledge and expertise to identify. Injection vulnerability leads to:

Data loss or corruption by insertion/modification/deletion of data, Data disclosure, Complete host takeover by executing OS Commands, Bypass authentication controls, Reputation loss

Injection attacks also depend on the privileges given to the application user to connect to the database server. Remediation techniques for Injection vulnerabilities are ***Use of prepared statements and parameterized query, Input sanitization using white listing, Escaping user inputs and Use of least privilege concept for database permissions and user access***

Injection flaw is a known and critical attack variant, but by implementing proper countermeasures you can protect applications and take necessary steps toward keeping your data secure.

C. Cross Site Scripting (XSS)

Considering Whenever application takes untrusted input/data and sends it to a web browser as a client side script without proper sanitization or escaping; client/user's browser will execute the script as it is, as it thinks the script came from a trusted source. This vulnerability where malicious man in middle attacker can get malicious code executed on client is called ***Cross Site Scripting*** popularly known as ***XSS***. There are three known types of XSS: ***Stored or Persistent, Reflected or Non-Persistent and DOM Based,*** We will discuss these with examples.

Stored/Persistent XSS: In this variant of XSS, the injected malicious code/script is stored in data store and

it will get executed every time when user accesses the data. *Example*: Assume that application is having two roles *Admin* and *User*. When *'Admin'* logs-in, he can see list of username and other administrative pages. When *'User'* logs-in, he can only update their display name.
Now attacker logs-in as *'User'*, and enters below as his display name

<a herf=#
onclick=\"document.location=\'http://www.attacker.com/session.aspx?s=\'escape\(document.cookie\);\">Hello

The above content entered by attacker as display name which will get stored in the database.
Now, when the *'Admin'* logs-in, he will see a link named "Hello" along with other usernames. When *'Admin'* clicks on link, it will send current logged in session cookie to the *www.attacker.com* which can be used in Session Replay kind of attack. This vulnerability will impact all *'Admin'* users of the application.

Reflected or Non-Persistent XSS:

In this variant of XSS, the injected malicious code/script is reflected from the web server in response as error message, search result or any other response that includes full or partial part of the input sent to the server as part of the request. This is also called as ***One-time Attack***.

Example: Web Application offers a personalized view of a application and greets a logged in user with *"Welcome, <Username>!"* message. The data referencing a logged in user is stored in the query string parameter of URL. Assume the URL of the Application is:

http://www.example.com/home.aspx?user=Bob
It displays,
Welcome, Bob!

Now attacker modifies the URL as follows:
http://www.example.com/home.aspx?user=<script>alert(document.cookie)</script>

It will show an alert message containing Cookie value in it. Non-persistent XSS is delivered to victim by email, phishing site or engaging user's in accessing maliciously crafted links.

DOM (Document Object Model) Based XSS:

In this variant of XSS, malicious code/script is executed through modification of the victim's browser DOM used by the initial client side script, so that client side script executes the malicious script. In this web threat, page content does not change, but the client side script present in the page executes differently due to modifications done by the attacker in the DOM. This can be considered as Client-side Attack. *Example*: Consider the following HTML page Welcome.html

<html>
<title>Welcome!</title>
<script>
var position=document.URL.indexOf("name=")+5;
eval("document.write(document.URL.substring(position,document.URL.length))");
</script>
**
**
Welcome to the Site!!!
… … …
</html>

Normally, this HTML page would be used for welcoming the users, example
http://www.example.com/welcome.html?name=Bob
However, a malicious request like

http://www.example.com/welcome.html?name=<script>alert(document.cookie)</script>

will show alert message containing Cookie value in it. Let's understand why XSS executed: the victim's browser receives this link and sends HTTP request to *www.example.com* and receives above HTML. Browser then starts parsing this HTML into DOM. When the parser arrives at the Javascript code, it executes and modifies the raw HTML of the page and malicious script also will get executed along with it. XSS vulnerability leads to

- ***Sensitive information disclosure through phishing, keylogging, identity theft***
- ***Hijacking of User's Session***
- ***Deface websites***
- ***Redirect user to malicious sites or malware***
- ***Accessing a user's geolocation, webcam, microphone and even the specific files from the***

user's file system. HTML5 APIs tried to address this with requirement of user opt-in for these features. However, XSS in conjunction with some clever social engineering can achieve target

Remediation Techniques for XSS vulnerabilities are:

- ***Enforce output HTML Encoding and Javascript***
- ***Encoding of all user supplied inputs***
- ***Input Sanitization using white listing***
- ***Use of AntiXSS libraries for Input Validation***
- ***HTML, Attribute, CSS and JavaScript Escaping***
- ***Use of JSON and HTML Entity Encoding***
- ***Use of HTTPOnly cookies***
- ***Implementing Content Security Policies***
- ***X-XSS-Protection Response Header***

Other than above remediation technique, there are other XSS defence which are emerging such as Javascript sandboxing tools, auto-escaping templates etc. which can be used by developers to stay ahead of the curve and avoid being vulnerable to these web threats.

D. Cross Site Script Forgery

Cross Site Script Forgery, in short ***CSRF*** is a client-side attack which eventually executes the attack on the server. It uses victim's browser feature to execute this attack by unknowingly sending request to the server. Malicious code is often not on the attacked site, thus it called as "***Cross-site***". Once user establishes trust with the application, this trust remains in the browser process until logout and thus browser does not require further intervention related to trust for subsequent requests. This vulnerability also known as *XSRF, Session Riding, One-click attacks, Cross-site Reference Forgery, Hostile Linking and Automated Attack.* Let's understand this by example.

Assume that user is authenticated on the E-Commerce site www.example.com. The attacker crafts and stores a malicious script on www.attacker.com and enforces victim to visit their site.

Malicious Link embedded on www.attacker.com is:

****

When Victim accesses www.attacker.com from another tab or instance of the same browser where authenticated session is already established, automatically request to www.example.com will be sent from www.attacker.com along with the all session cookies and required content required for www.example.com. Server considers this as a legitimate request and process the transaction. CSRF vulnerability leads to: **Critical functionality exploitation, Identity theft, Sensitive data disclosure, Initiation of unwanted transactions, ad Modification of data**

Remediation Techniques for CSRF vulnerabilities are:

- **Use of CSRF Token (Unique random token) in each HTTP Request as hidden form fields or URL**
- **Avoid using GET Methods for critical transactions**
- **Check referrer for before processing request**
- **Implement multi-factor authentication or CAPTCHA for critical transactions**.

E. Broken Authentication and Session Management

HTTP is a stateless protocol and in such scenario user's session becomes most critical for Web Applications. Any issue in authentication mechanism or session management such as unprotected account, credentials, session Token/IDs leads to impersonation of legitimate users. Improper implementation of logout, application timeouts, password management, account recovery options, account updates, session token generation algorithm results in Broken Authentication and Session Management vulnerability.

During authentication, application doesn't assign a new session ID, making it possible to use an existing session ID leads to Session Fixation attack. The attack requests user to authenticate himself with a known Session ID (existing/already used session ID), and then using the same session to impersonate a victim. This established session used in the victim's browser allows an attacker to access application functionalities without being asked for login. Let's understand this with Example:

- Attacker establishes a legitimate connection with Application www.example.com
- Application issues a Session ID for e.g. **www.example.com/login.aspx?sessionID=L2dBISEvZ0FBIS9nQSEh**
- Attacker sends a link to a Victim for e.g. **www.example.com/login.aspx?sessionID=L2dBISEvZ0FBIS9nQSEh**
- Victim clicks on this link and gets redirected to login page with pre-defined session ID
- Victim logs-in to the application with their credentials and user session is established by the Application and user details are stored in the **sessionID "L2dBISEvZ0FBIS9nQSEh"**
- Attacker also accesses same link and they will get redirected to home page as session contains valid user values as victim already logged in to the application

Broken Authentication and ***Session Management vulnerability leads to: Session Hijacking, Session Replay, Identity theft, Privacy violations, Information Disclosure***

Remediation Techniques for Broke Authentication and Session Management vulnerabilities are

- Use framework provided session IDs
- Invalidate existing session prior to generating new session for same user
- Implement session timeouts and periodically renew Session IDs
- Implement multi factor authentication of critical functionalities including
- Using OTP authentication
- Set proper domains, path and expiry time attribute for session cookies
- Do not implement SessionIDs in GET/POST, rather use HTTP Cookies
- Use HTTPOnly and Secure attributes for HTTP Cookies.

IV. DETECTION of WEB THREATS

A. Introduction

Web applications have long been victim of attacks. The sophisticated technology and computation power that is available for all, is sadly available for people with malicious intentions as well. We have seen a long history of smart hackers who have breached the digital security walls of organizations and even government websites and brought them to their knees. Recent exploits such as "***WannaCry***" and **"Petya"** ransomwares have run havoc on millions of un patched machines of not just end consumers but corporates as well. Living in an era where most of our work gets done digitally, in some way or other, it is almost a scary thought to have all our digital assets stolen by malicious attackers. As the saying goes, "*A chain is as strong as its weakest link*", we should ensure that even the weakest link in the application software is strong enough to face any reasonable security threat from external entities.

It becomes very important for us to identify all the vulnerabilities of the system and come up with action plan of how to address them. Detection of vulnerabilities and subsequent patching is a critical step before moving the code base to production servers. In this section, we discuss how security and penetration testing can help in getting an unbiased external perspective on exposed vulnerabilities of the product. Penetration tools and practices are discussed next. We then introduce ***Burp Suite*** tool which is one of the popular tools for web security assessment.

B. Web Security and Penetration Testing

A well-planned security testing is important before releasing the product to production environment. Security testing is apart from the regular functional testing which is to be conducted for verifying and validating the product functionality. Quality metrics that is tracked for each release should take into consideration, the security awareness and maturity of the team and get the applications tested for security. In many organizations, web surety team is created who work full time of product security and they are complimented by security leads in individual teams who analyse the product security right from the inception of a project till it is deployed to production servers. Security leads of the team need to analyse which methodologies need to be used for testing the software security of the product. In cases where minor updates are being pushed live, then security testing is concentrated on the delta changes that is done as part of that release. A good security test plan includes a mix of these methodologies [7]: ***Static analysis, Dynamic code analysis, Fuzz Testing, Binary Fault Injection, Vulnerabilities scanning, Byte code analysis, Penetration testing***

Penetration Testing is synonymous with term "*ethical hacking*". Developers, testers, designers or architects who work on a product release would have a certain notion of product security. This may result in false presumptions of product security. It is often seen that, when an issue gets reported from the field, it is because of overlooking certain parts of product which are assumed to be vulnerability free.

Hence, it always helps to assess from time to time, the level of product security from a person or group, external to an organization. They should have very little or no knowledge about the product and what it does to avoid any presumptions. There exist groups of certified security professionals who are equipped with knowledge and tools to assess the security strength, and overall vulnerabilities of the system. These professionals are called as "***Ethical Hackers***". Occasionally, a responsible organization should employ external groups to scan their website and provide a security report on overall product security. This process of security testing where the tester is given an opportunity to find the strength of product security and enlist all its vulnerabilities using tools and techniques is called ***penetration testing***. During this exercise, the external group signs an agreement with the organization that they would not disclose any information found during penetration testing. Once the report is provided, organization needs to work on it seriously and ensure that all vulnerabilities are patched. A fresh run of penetration testing may be required once organization completes the security patching of the product.

Ronald Kurtz et. al. [7] describe briefly about cloud penetration testing. Per authors, a penetration test should aim to determine the feasibility of a security attack from external entity or to check if anyone has already been successful in attacking the system. Components of a such a security audit includes: ***Level 1 – High level assessment*** **-**Review of data policies, procedures, guidelines and standards. ***Level 2 – Network evaluation*** **–** Information procurement including preliminary scanning of folder structure, open ports etc. ***Level 3 – Penetration test*** **-** systematically attacking the application from view of an attacker to find weak stops to compromise the system.

C. Penetration tools and practices

Some of the practices for systematically exploiting the system and finding the weak links as part of penetration testing are discussed below. Penetration testing is carried out in 3 phases [7]: ***Preparation*** (contract signing etc.), ***Execution*** (execution of penetration testing using tools) and ***Delivery*** (Security report evaluation and communication)

During and after penetration testing, the ethical hacker/s designated for conducting the penetration tests are required to exhibit ethical behaviour and not disclose and misuse any information found during penetration testing. Before performing actual penetration testing, a military based approach of finding maximum information about the attack surface is carried out. This is referred to as, *reconnaissance* process. It involves gathering information about attack surface, determining the network range and the active machines in the network along with their open ports which can be exploited as access points. Network mapping and knowing more about operating system which is running on the systems and the services which are running on them. The pre-tests which are carried out to achieve the above objectives are: ***Foot printing*** **–** Accessing the security profile of an organization., ***Scanning*** **–** Scanning network to find the range, open ports, access points, information on operating system etc., and ***Enumerating*** **-**mapping the network to give a bigger picture of the attack surface for systematic attack.

There are certain specialized tools available for penetration testing which have been discussed by Varun M Deshpande et. al. [6]. Some of them are listed here:

Visual Traceroute – is a freeware which visually depicts that path taken by an IP packet from source to sink. In the report, an approximate geo physical location of the networks along with hop details are provided. This helps in debugging issues with bottleneck, network congestion etc.**Visual Route** – is a freeware which has functionality of traceroute along with reverse DNS and whoIs look up, **Burp Suite** – A comprehensive web application security assessment tool which we will discuss later in the section., **Web cracker** – A brute force technique based password guesser., **OWASP ZAP** – a tool like Burp suite developed by OWASP community, **Wireshark** – a popular web packet sniffer which can sniff multiple IP protocol based packets,

Fiddler – HTTP/S proxy for packet sniffing and analysing.

There are several other notable penetration testing tools available in the market. However, the tester needs some amount of training to use them and we need to execute caution before choosing a tool as the tool itself might be eavesdropping into the application that we are testing. Hence, using software from known and reputed sources is recommended.

D. Detection of Web Threats using Burp Suite

Based Burp Suite from Portswigger [13] is a web security assessment tool. It is used for automatically crawl and scan websites for over 100 known vulnerabilities which includes OWASP Top Ten web security risks. Using Burp Suite, one can scan for vulnerabilities, intercept browser traffic, automate custom security attacks. It supports several attack insertion points inside HTTP headers, parameter names, cookies, URL path etc. Burp suite is an ideal tool for web security penetration testers. It has capabilities to perform static code analysis as well. Along with all these capabilities, it provides best in class automated security test report. We will share details on how we need to analyse Burp Suite report, later in the paper. Burp Suite edition comes in a free edition and in professional edition. Free edition has limited capabilities, while professional edition is a full-fledged tool. We will show some of the configuration and capabilities with professional version of Burp Suite.

Step 1: After creating a custom project, we need to add target scope listing all the web domains that we wish to scan. As shown in figure 2, navigate to "Target" -> "Scope" and click on "Add" and list the domains that needs to be scanned. When the target scope is defined, Burp suite will concentrate on the security testing on these domains only and not on HTTP/S traffic of other sub domains. Hence, it is important to list all required sub domains in the app.

Step 2: By default, HTTP traffic interception is switched on. For sake of convenience, we can switch it off by navigating to "Proxy" -> "Intercept" and switch off Interception as shown in figure 3

Step 3: We mentioned earlier that several attack insertion points can be added for performing security testing. This is done under "Scanner" -> "Options" tab. Figure 4 shows part of the screen grab from this tab's contents. A security tester can choose as per application requirements, various attack insertion points.

Step 4: One more recommended live scanning configuration is under "Scanner" ->" Live Scanning". Select "Use Suite Scope [defined in Target tab] as shown in figure 5. This will limit the vulnerability scanning to those sub domains which have been listed under Target/Scope in Step 1.

Step 5: Usually, Burp suite is not run on production servers. Instead, it is run specifically on selected production replica or QA servers. The host name resolution details of the non-production servers can be mentioned under "Project options" -> "Connections" tab. We need to ensure that all the scoped domains mentioned in Step 1 are covered under this host name resolution of current step.

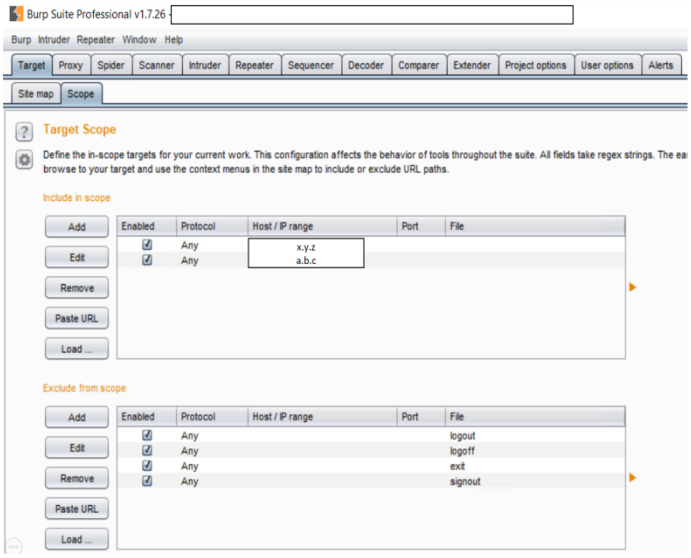


Figure 2: Burp Suite – Target scope addition

Figure 3: Switch off Interception

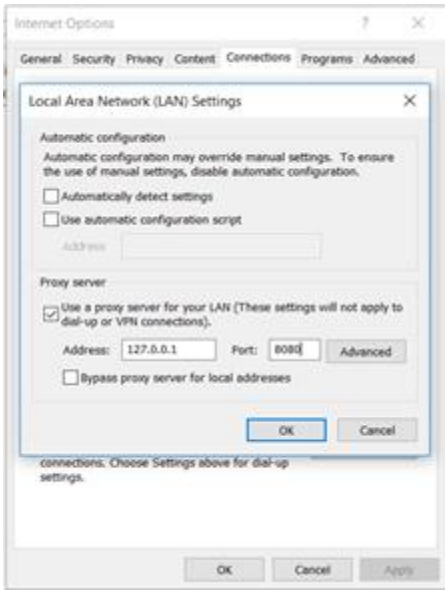


Fig 4: Add scanner options

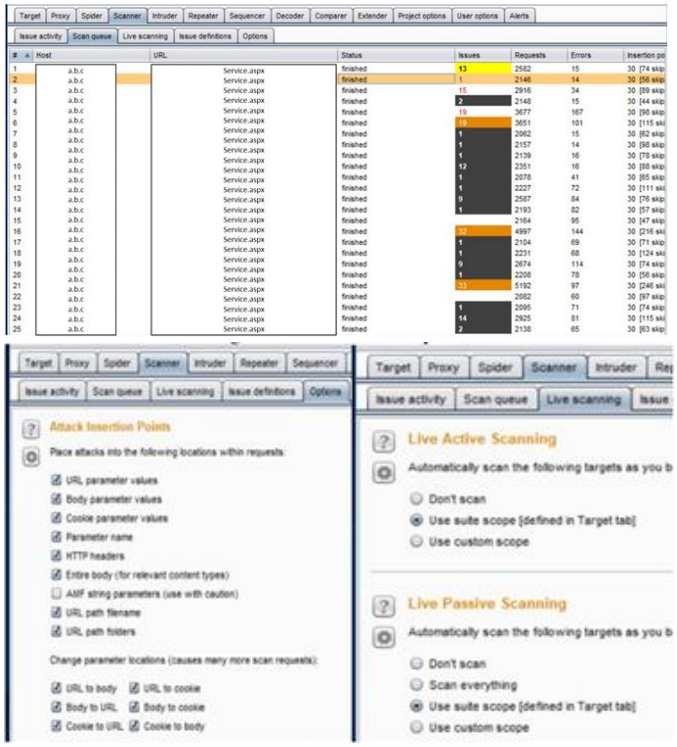


Fig 5:Live Scanning Options

With these steps, Burp suite project is ready for accepting HTTP/S traffic from client machine where manual security testing is being done. Security tester tries to access all the functionalities provided by the web application, end-to-end . He also tries to scan for any accessible web services which can be exploited. All these HTTP traffic needs to be redirected to the IP and port to which Burp Suite is listening. If Burp suite is configured on the same machine, proxy IP may be mentioned as "127.0.0.1" which is self IP. Usually, Burp suite listens to port "8080". This is configurable under "Proxy" -> "Options" -> "Binding" tab.

Step 6: From the client machine, where the security testing takes place, we need to set up proxy settings. It is to ensure that all the HTTP/S traffic goes through Burp suite. Burp suite takes care of forwarding the IP packets and returning the response back to the client. As mentioned in Step 2, intercept can be turned on or off based on project requirements. Burp suite acts as a "man in the middle" to simulate web security attacks. Proxy setting can be performed easily using Internet Explorer. In IE, under "Internet Options" -> "Connections" -> "LAN Settings" -> Proxy Server settings, update the IP address and port details of the server where Burp suite is configured to listen to web traffic as shown in figure on the left. In case, it is configured in same machine, home IP address can be mentioned.

Step 7: Once the security testing starts, HTTP requests and response start getting logged in Burp Suite under "Proxy" -> "HTTP History". For each request, automated security tests are initiated and status of the automated tests are updated in "Scanner" -> "Scan Queue" as shown in the figure below. Automated security testing takes a long time to complete spanning for few hours.

Step 8: Once security testing is complete, the vulnerabilities which are detected in the web application is listed under "Target" -> "Site Map" tab. These security issues can be exported to generate a security report for the test execution. The analysis of this security report is discussed later in this paper.

V. WEB THREATS- MITIGATION STRATEGIES

A. Introduction

Equating a well-run web application based service to real world country which consumes and provides various services; we find that security can be drawn as a parallel requirement. Like a country faces lot of internal and external security threats, so does a web application. From outside the web application, it faces constant threat from malicious entities trying to cause harm and break the system. While, usage of unsecured and vulnerable components in building critical functionalities might put the web application is serious threat. Similarly, OWASP Top Ten vulnerabilities can

be mapped to a corresponding security concern of a sovereign country. In the same way, that a country tries to protect itself from external and internal malicious forces, web application developers also need to spend effort in securing the website from all known vulnerabilities. Mitigating the web security risks will ensure that organization stays in business and stays competitive. In this section, we discuss some of the recommendations from security community to ensure web security vulnerabilities are mitigated or avoided at various level of application development.

B. Secure Development Life Cycle

Setting up a formal process which ensures high quality output is necessary for any process driven organization. Traditionally, security, being a non-functional requirement, has been sub consciously side-lined when compared to the functional requirement and performance. Lately, this trend is changing and organizations have started to adopt security best practices into their development life cycle. This inclusive security oriented development strategy is termed as "***Secure Development Life Cycle***" (SDLC). Microsoft was one of the pioneers who introduced, practiced and advocated a formal secure development life cycle framework for software delivery. This initiative was started in 2004 and first formal publication happened in 2008. It has evolved over time and it has helped Microsoft achieve great results in the process. Microsoft's SDLC [14] involves below steps which are briefly explained here. **Training**: Employees get core security training., **Requirement**s: understand security requirement., **Design**: establish security design requirements, prepare threat modelling and reduce attack surface, **Implementation**: Use approved tools and libraries, static code analysis, deprecate unsafe functions from code., **Verification**: Perform security testing, **Release**: After final security review, document and form incident response plan 7& team to monitor the release., **Response**: In case any incident is reported or detected, execute incident response plan. It is recommended that each organization should have a formal SDLC program which is properly funded. Each release should be certified by security experts within the team and any major change needs to be reviewed thoroughly before implementation.

C. Secure Coding Recommendations

With SDLC in place, it is important to setup efficient coding practices during design and implementation phase. The way we write code and handle data can make the product vulnerable. Writing clean and secure code is essential to mitigate security threats. Writing unsecure code may result in exploitation and data breach of the application and cause financial loss for the organization. Like we discussed earlier, the OWASP top ten vulnerabilities can be avoided by secure development practices. Hence, it is essential to educate the developers and designers to write clean and secure code.

OWASP has recommended secure coding practices [1] for application developers. Any organization can use this as a standard and practice writing secure code. They state that upholding CIA triad is the foremost responsibility of developing secure code. Per them, security vulnerabilities can be introduced at different stages of development such as not able to understand security requirements, poor design which have logical errors, poor coding and deployment of application etc. They give a secure coding practices checklist and recommendations for developers for writing clean and secure code. Some of them are ***Input validation, Output Encoding, Authentication and password management, Session Management, Access control, Cryptographic practices, Error handling and logging, Data protection, Communication security, System configuration, Database security, File*** **management and Memory management.**

Some of the other recommendations are to use HSTS (HTTP Strict Transport Security) in all the web pages and even in sub domains [2]. Using HSTS tag in the web page ensures that all the transactions are secured by HTTPS SSL communication and the message exchanges are encrypted. HSTS HTML tag looks like below. This should be included in the header section.

```
Strict-Transport-Security: max-age=31536000; includeSubDomains
```

As MD5 and SHA1 encryptions are becoming outdated, it is recommended to use SHA256 encryption logic in all the secure transactions. Ensure that security sign-off is done for each release, however small the release scope may be. Periodic, full vulnerability scans should be run by security experts and management should invest resources in fixing the vulnerabilities.

D. Analysing Burp Suite Report

Based Burp Suite helps in scanning and generating reports. Major work is to analyse this report and providing fix to the relevant team for resolution. Report generated by Burp Suite is very easy to understand. Issues reported by Burp Suite are classified based Confidence vs Severity Matrix.

Each Vulnerability is further classified in different parts based on ***Issue Background, Remediation Background, Vulnerability Classification based on CVE, Issue detail for*** **specified issue scenarios and Remediation detail for specified issue.**

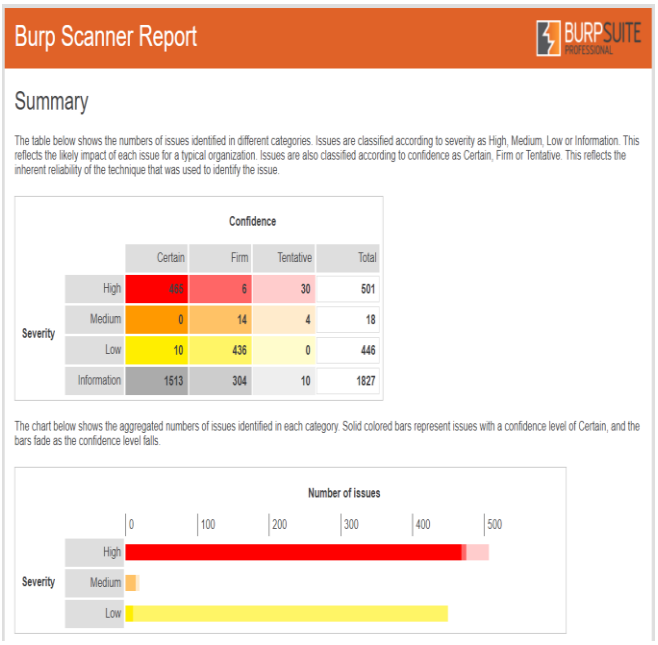


Figure 8 – Burp Suite report

This report helps reviewer in getting full details about the vulnerability and its resolution along with actual HTTP Request & Response of attack for evidences. Burp reports less number of False Positives compared to other tools but based on the individual's knowledge on the vulnerability, one can deduce that the issue is a false positive or not.

Figure 9: Defect Summary

E. Optimization of Web Security Practices

Security has always been a non-functional requirement. As an organization, management need to make trade-offs. Optimization of security needs to enable to cloud application rather than decreasing overall performance, increase cost and usability of system. Being over protective of web applications may have an adverse effect on the usability of the software. Providing high security should not impede the success of the company, but it should enable it to grow without any risks. In this context, security experts and business should sit together and discuss about the level of security practices that need to be put in place to feel reasonable secure. Analysing the risk rating and the security reports would help in deciding how much resources can be spent on optimizing the security of the system. As per recent Gartner predictions [15], cyber security spending is going to increase by 7.6% to reach $90 billion in 2017 and would like reach up to $113 billion by 2020. As the resources are limited, web security needs to be dealt with smartly. Security expenditures of fixing vulnerabilities should be avoided by following above said recommendations for developing secured cloud based web applications.

F. Working towards Trusted Computing

In the beginning of the paper, we discussed how data privacy and security are inter dependent and complimentary concepts. Ensuring data privacy and security is imperative for giving a sense of trust for the end users of system. News that service provider's data was breached does not help the users to trust the service provider. Trust is a trait which is a deciding factor for a user to make a purchase or stay connected with a brand name or service provider. Hence, it is very important for service providers to move towards trusted computing.

We have highlighted several companies coming under scanner for privacy breaches and data theft. This is an ongoing issue with fresh of privacy breach surfacing by passing of each week. As the threat landscape is increasing, hackers are managing to get hold of huge computing resources and technologies to employ sophisticated attacks on web applications. Hence companies need to relook at their security strategies and investments to reassure the user base about the trust that they have placed on them and move towards trusted computing.

Most recently, Equifax, a consumer credit reporting company reported a monument data breach which affected over 143 million customers' personally identifiable information including financial details and social security numbers [21]. Cyber security criminals used a vulnerability in their website, to hack into the system which eventually transpired into huge data theft. Equifax got to know about the data breach on July 29th. However, public disclosure happened on September 7th. Equifax is offering 1 year free extended subscription for all their customer in a bid to stopping them from suing them for data loss. This incident is a wakeup call for all security professionals to discuss about web security and trusted computing with highest priority.

VI. CONCLUSION AND FUTURE SCOPE

We began this paper, by understanding the interdependent relationship between privacy and security and their complimentary nature. We gave a broad overview of the web computing paradigm and web application threats. OWASP community was introduced to the reader along with man in middle attacks. By this, we established the need for vulnerability assessment of web applications and laid ground work for rest of the paper. OWASP top ten web application security risks give us a reference for identifying and be aware of the top web threats. We analysed the evolution of these web threats over the past decade. We also described other web threats such as "WannaCry" ransomware where which created havoc for millions of users worldwide.

We picked 3 of the major web vulnerabilities listed under OWASP top ten and broke it down to understand how to identify the threat and how to avoid the same by employing smart techniques. We later recommended use of secure coding practices and secure development life cycle to mitigate these vulnerabilities. We discussed in detail about detection of web threats through security testing and penetration testing. Along with best practices for security and penetration testing, we introduced several tools and techniques for performing the tests. Burp suite, the popular web security assessment tool was taken as a case study to show how to configure, and run automated security testing by adding various security attack insertion points. We recommend this usage of professional version of this tool for getting best results of web threats and vulnerability assessment. We described how the security report could be analysed by security expert and action taken accordingly.

We reminded the readers on the importance and need for moving towards trusted computing and why user privacy and data security play a vital role in ensuring success of any organization. Especially so, in the light of security incidents such as Equifax. We believe that investment on security should be made smartly by use of best practices and vulnerability assessment tools so that security is optimized to enable the organization to be successful and make users feel safe while they are accessing the product. We need to continuously put our efforts to find trustable software solutions for developing secured cloud based services; to stay ahead of the bad guys and be cyber guardians of the digital world.

VII. REFERENCES

[1]. OWASP Secure Coding Practices Quick Reference Guide Link: https://www.owasp.org/images/0/08/OWASP_SCP_Quick_Reference_Guide_v2.pdf (Last accessed on 4th Sep 2017)

[2]. HTTP Strict Transport Security Cheat Sheet Link: https://www.owasp.org/index.php/HTTP_Strict_Transport_Security_Cheat_Sheet (Last accessed on 4th Sep 2017)

[3]. OWASP Top 10 2017 rc1 - https://github.com/OWASP/Top10/raw/master/2017/OWASP%20Top%2010%20-%202017%20RC1-English.pdf (Last accessed on 9th Sep 2017)

[4]. OWASP Top 10 - 2013 https://www.owasp.org/images/f/f8/OWASP_Top_10_-_2013.pdf (Last accessed on 7th Sep 2017)

[5]. OWASP Top 10 2010 - https://css.csail.mit.edu/6.858/2011/readings/owasp-top-10.pdf (Last accessed on 7th Sep 2017)

[6]. Varun M Deshpande, Dr. Mydhili K. Nair, Ayush Bihani, "Optimization of Security as an Enabler for Cloud Services and Applications", to be published by Springer in edited volume titled "Cloud Computing for Optimization: Foundations, Applications, Challenges", to be published in "Studies in Big Data" book series, Springer (2017)

[7]. Ronald L. Krutz and Russel Dean Vines, "Cloud Security: A Comprehensive Guide to Secure Cloud Computing," Published by John Wiley & Sons, 2010
[8]. Mark Rhodes-Ousley, "Information Security The Complete Reference, Second Edition", Published by Tata McGraw-Hill, 2013
[9]. Siani Pearson, George Yee, Book - "Privacy and Security for Cloud Computing", Computer Communications and Networks, 2013, ISBN: 978-1-4471-4188-4
[10]. Saltzer, J. H., and Schroeder, M. D., "The Protection of Information in Computer Systems," Fourth ACM Symposium on Operating Systems Principles, October 1974.
[11]. Varun M Deshpande, Dr Mydhili K. Nair, "Need for User Centric & Unified Privacy and Data Policies for Social Networking. Case Study: Google, Facebook, Amazon & Flipkart", to be published in International Journal of Latest Engineering Research and Applications (IJLERA) ISSN: 2455-7137, Volume - 02, Issue - 08, August - 2017, PP - 83-93
[12]. OWASP Top Ten Project https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project (Last accessed on 7th Sep, 2017)
[13]. Burp Suite - https://portswigger.net/burp (Last accessed on 8th Sep, 2017)
[14]. Microsoft Secure Development Life Cycle https://www.microsoft.com/en-us/sdl/ (Last accessed on 8th Sep, 2017)
[15]. "Cyber security spending to reach $90 billion in 2017, Gartner says" news report https://businessinsights.bitdefender.com/cyber-security-spending-2017(Last accessed on 8th Sep, 2017)
[16]. Heart Bleed Bug http://heartbleed.com/ (Last accessed on 9th Sep 2017)
[17]. SSL 3.0 Protocol Vulnerability and POODLE Attack https://www.us-cert.gov/ncas/alerts/TA14-290A(Last accessed on 9th Sep 2017)
[18]. WannaCry ransomware attack https://en.wikipedia.org/wiki/WannaCry_ransomware_attack(Last accessed on 9th Sep 2017)
[19]. Petya (malware) https://en.wikipedia.org/wiki/Petya_(malware) (Last accessed on 9th Sep 2017)
[20]. Locky (malware) https://en.wikipedia.org/wiki/Locky (Last accessed on 9th Sep 2017)
[21]. "Equifax data breach: Find out if you were one of 143 million hacked" news article https://www.cnet.com/how-to/equifax-breach-find-out-if-you-were-one-of-143-million-hacked/ (Last accessed on 9th Sep 2017)