Cyber Security Risks For Modern Web Applications: Case Study Paper For Developers And Security Testers

Devanshu Bhatt

Abstract: "May you survive in fascinating times" can be an English phrase claiming to be considered an interpretation of the traditional Chinese curse. Cyber Security risks are becoming top concerns as we find out frequent data breach occurrences on regular basis now a days from organizations like Equifax, Anthem, JP Morgan Chase and other large corporations. As per IBM this year's global average cost of data breach is $3.62 million. Findings from NIST (National Institute of Standards and Technology) shows that 92% of security vulnerabilities exists at the application layer not in the network layer. In this research paper; subsequent subject areas discussed -Introduction to Cyber security, Web applications security challenges, Top web applications vulnerabilities and conclusion with approaches and mindset to comprehend for developers and security testers.

Index Terms: Cyber Security, Cyber Security Risks, DAST, SAST, SQL Injection, Top 10, vulnerabilities, Web Application Security Risks

1. INTRODUCTION

The problem of vulnerable software program is potentially the most crucial complex concern of our era. The remarkable increase of web applications permitting organizations has only worsened the requirements to build a powerful technique to creating and locking down our Internet, Web Applications and Data. Vulnerable software application program is undermining our financial, healthcare, and other crucial system. As our software program gets significantly complicated, and linked, the problems of attaining software security raises tremendously. The fast velocity of current software development techniques can make the most frequent threats crucial to uncover and solve rapidly and precisely. This paper is broken down into four primary components - "Introduction" presents general introduction about cyber security threats in today's era for software applications. The "Web Application Security Challenges" and " Top Web application vulnerabilities" present the detail approach of different challenges and number of security risks. At last the "Conclusion" explains reasons and paths to follow which also includes the future considerations for the improvement.

2. WEB APPLICATION SECURITY CHALLENGES

2.1 Industry Security Exploits Statistics

As per industry numbers collected in 2017 Application security statistics report:

- Out of total breaches reported: 30% involved attacks on web applications, 62% featured hacking to exploit vulnerabilities.
- Out of total attacks on web applications reported: 93% financially motivated and executed by organized criminal groups, 77% carried out by botnets and 32% exploited SQL Injection errors.

Methods accessible to cyber criminals are unbelievable. Numerous advanced hacking resources are incredibly openly accessible on the web including software and hardware In accordance to Gartner, corporations invested $719M on software security in 2016, up from $630M in 2015. Which is excellent news; on the other hand relative to other cyber security methods, web application security still isn't enjoying nearly enough investment.

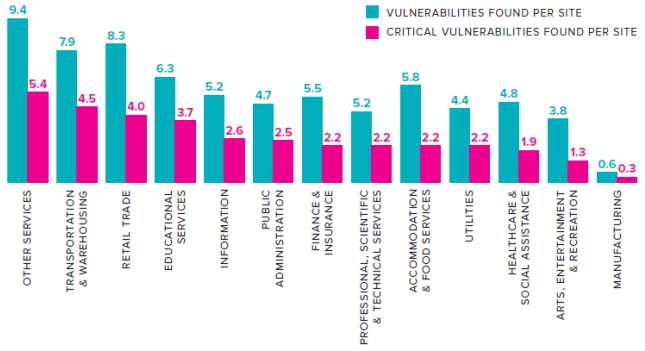

Fig. 1. *VULNERABILITY PROFILE BY INDUSTRY*

Fig. 2. *VULNERABILITY PROFILE BY INDUSTRY*

- *Devanshu Bhatt is currently working as Consultant, IT App Development at Nationwide General Insurance, Columbus, OH, U.S.A. PH-6784683152. EMAIL: devanshu20@gmail.com*

Fig. 2. *exhibits that in several important market sectors –*

Utilities, Education, Accommodations, Retail and Manufacturing - about 60 % of their web applications continue to be "always vulnerable". In all but a single industry, at very least one-third of web applications are usually vulnerable. This indicates that organizations are not in a position to solve all of the "serious" vulnerabilities discovered in their applications, and it takes them a extended time to remediate significant vulnerabilities.

2.2 Web Application Security Risks

Only several web applications are developed securely from the surface, or maintained with procedures developed to maintain them secure over the time. Actually the web itself never designed to securely run mission critical applications. Actually web itself was never designed to securely run mission critical applications. Web applications protocols were designed to be light weight as well flexible; and lacked privacy, integrity and security features in the past when designed. Many web applications are mission critical, but may not have been when they were designed. Not many web applications were designed securely from ground up. Web applications are complex combinations of platforms and services from multiple providers, each with its security challenges. The web applications depend on remote web browsers which is neither secure nor trusted, since it adds more complexities and flaws. As well many users have a tendency to click on links without considering the risks of their actions. Web page addresses can be disguised or take you to an unexpected sites. Many web browsers are configured to provide increased functionality at the cost of decreased security. New security vulnerabilities may have been discovered since the software was configured and packaged by the manufacturer. Computer systems and software packages may be bundled with additional software, which increases the number of vulnerabilities that may be attacked. Many web sites require that users enable certain features or install more software, putting the computer at additional risk. Many users do not know how to configure their web browsers securely. Many users are unwilling to enable or disable functionality as required to secure their web browser. Hackers can possibly use numerous various routes via your application to do damage to your organization. Every single of these routes symbolizes a threat that might, or might not, be significant enough to bring about consideration. Occasionally these routes are insignificant to discover and exploit, and sometimes they are incredibly challenging. Likewise, the damage that is triggered might be of no outcome, or it might place affected organization out of business. To ascertain the threat to organization, once can assess the probability linked with each and every threat agent, attack vector, and safety weakness and blend it with an estimate of the technological and organization impact. Collectively, these aspects determine overall risk.

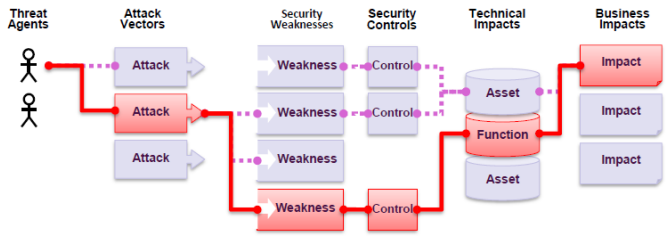

Fig. 3. *VULNERABILITY ACCESS ROUTES*

2.3 Web Application Security Threat - Real time

As per Fig. 4, it shows the real time threats and attacks which is occurring on constant basis round the clock, Attackers are constantly trying to get in system and take benefits from potential system flaws, as per example recent data breach in 2017 from large organization like Equifax, which has compromised millions of personal data of credit consumers. As per kaspersky report In 2017, "the DDoS Intelligence system recorded DDoS attacks against resources in 72 countries, which is eight less than in the fourth quarter of 2016. The Netherlands and the UK replaced Japan and France among the top 10 countries with the most DDoS victims. South Korea remained the leader in terms of the number of detected C&C servers. The US came second in this respect, followed by the Netherlands, which dislodged China from the top three for the first time since monitoring began.

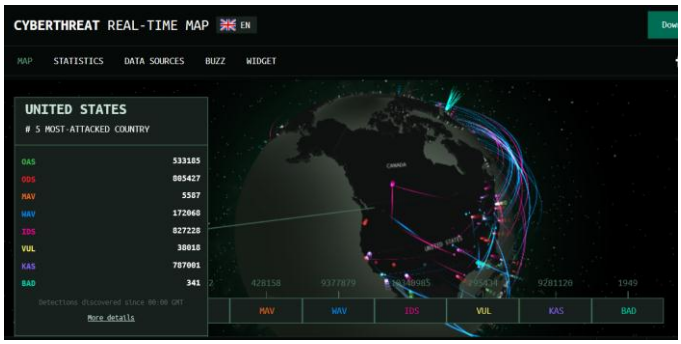

Fig. 4. *REAL TIME CYBER THREAT MAP*
https://cybermap.kaspersky.com/

3. TOP WEB APPLICATION VULNERABILITIES

Fig. 5. shows comparison between top vulnerabilities of 2013 and 2017.The analysis shows that there is not much difference between top vulnerabilities even though after few years passed, top exploits still remains on the top, even with all advancement in technologies, languages, Machine learning techniques, Artificial Intelligence and many more. These vulnerabilities are so prevalent and many of these vulnerabilities are very easy to exploit. By understanding most common of these web application vulnerabilities - Developers and Testers gets confidence on how to securely code web applications and what strategies and procedures to implement for security testing. Here is a listing of widespread web application security vulnerability types, as well as the objective of this list is usually to produce an knowledge and understanding for everybody who is associated with building software program. It really is fundamentally some tips about what to avoid; if one don't

want to get compromised. One can find, obviously, many different ways to compromise and breach applications which go beyond the OWASP Top 10, however the list is a fairly nice beginning. Security exploits are constant changing and emerging as time passes, there have already been a number of changes towards the listing. This paper concentrates on the OWASP Top Ten 2017 Release Candidate 2. Up to now, the Release Candidate 2 is among the most current version of the OWASP Top Ten available. This is what evolved through the 2013 listing towards the 2nd Release Candidate for 2017. As you can tell, most items remained exactly the same, when a few them combined and some brand new ones have been included.

OWASP Top 10 - 2013		OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	∪	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	∪	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

Fig. 5. *TOP VULNERABILITIES COMPARISON TABLE*

3.1 A1: 2017 - Injection
Injection weaknesses, this sort of as SQL Injections, NoSQL Injections, LDAP Active Directory injection, take place when non-trusted information is delivered to an interpreter as portion of a command or query. The hacker's aggressive and unpredictable data can trick the interpreter into carrying out accidental commands or being able to access information without having appropriate permission.

3.2 A2:2017 - Broken Authentication
Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.

3.3 A3:2017 - Sensitive Data Exposure
Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.

3.4 A4:2017 - XML External Entities (XXE)
Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.

3.5 A5:2017 - Broken Access Control
Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

3.6 A6:2017 - Security Misconfiguration
Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion.

3.7 A7:2017 - Cross-Site Scripting (XSS)
XSS flaws occur whenever an application includes un-trusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

3.8 A8:2017 - Insecure De-serialization
XSS flaws occur whenever an application includes un-trusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

3.9 A9:2017 - Using Components with Known Vulnerabilities
Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

3.10 A10:2017- Insufficient Logging & Monitoring
Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

4. Conclusion

4.1 Static Application Security Testing (SAST)
SAST functions like a compiler, It evaluates code line by line. It also generates a logic flow and information flow design, attempting to determine vulnerabilities that are present in an application, and could be exploited by some online hackers. The power of this technology, especially when you identify vulnerability, you know precisely in what

line of code the vulnerability is present and Identify vulnerabilities pretty earlier in the software development life cycle. What one just examined was not a runtime application, but just a code of it. many organizations, even if they have static application security testing, they cannot apply it, because they don't have access to source code of vendor 3rd party applications.

4.2 Dynamic Application Security Testing (DAST)

DAST in fact stimulates hackers' behavior, and It roll-outs simulated attacks on an application, and evaluates application responses, as well it can identify quite a few more vulnerabilities in comparison to SAST. It Can check 3rd party vendor applications due to the fact that it won't require application source code. one problem that it can identify Vulnerabilities late in the software development life cycle which is more expensive.

4.3 Approaches And Mindset

Even though there are number of tools and technologies available now a day's which utilize the machine learning and Artificial Intelligence (AI) approach to block or search and destroy exploits and vulnerabilities automatically to defend system; above top threats and current environment of cyber attacks and data analysis should be known and understood by every software application developer and tester to utilize secure coding practice and secure testing methodologies to implement effectively. Software application can be functional without being secure and can perform every specified action flawlessly, still can be exploited by a hackers. In order to identify security vulnerabilities, developers and testers have to think differently. There are difference between traditional web application defects and security web application defects. Traditional defect: Application behaviors which won't work as per requirement by business. While Security defect meaning: additional behaviors, their side effects, implications and interactions between the software and the environment; whatever it supposed to do, it also did something that may look harmless and irrelevant, but it is not. When conducting security testing, the goal is to find out if it can stand up to exploiters. It is more difficult to derive exploitation scenarios derived from an application's technical specification, but anticipating these scenarios is vital when planning for security testing. Functional testing involves testing an application under realistic user scenarios, while in security testing the test engineer frequently deals with scenarios that may not be realistic for the common user .A developer and test engineer must rely on information supplied by the application itself and. Must learn to extract that information from the application. During security testing the developer and test engineer has to consider from where the inputs are coming, how much trust can be placed in an input source, and what could go wrong if this trust is misplaced the application and its I/O interfaces. When carrying out security testing, however, deciding when to stop is more complicated because the developer and test engineer needs to help to quantify the risk and decide whether enough testing has been done to declare the product "safe". Assigning the proper severity level to a security defect is also important, so that the defect gets addressed appropriately.

REFERENCES

[1]. https://www.lifelock.com/education/history-of-data-breaches/

[2]. https://www.ibm.com/security/data-breach

[3]. https://info.whitehatsec.com/rs/675-YBI-674/images/WHS%202017%20Application%20Security%20Report%20FINAL.pdf

[4]. Source: Building a web application security program from Securosis.com

[5]. Source: CERT Securing your Web Browser

[6]. https://usa.kaspersky.com/about/press-releases/2017_kaspersky-lab-report-on-ddos-attacks-in-q1-2017-the-lull-before-the-storm

[7]. Source: Gartner presentation on SAST and DAST

[8]. https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project