Web Application Attacks and Countermeasures

LakshanePooja[1] Pawar Puja[2] SoniVarun[3] Sulakshana Mane[4]
[1,2,3]Student of B.E [4]Assistant Professor
[1,2,3,4]Department of Computer Engineering
[1,2,3,4]BharatiVidyapeeth College of Engineering, Navi Mumbai-400614

Abstract— Adoption of internet for business operations has changed the face of business interactions in business 2 business (B2B), business 2 consumer (B2C) or now popular consumer 2 consumer (C2C) models. Applications are built to adapt this changing business perspective and thereby giving more and more control to the end user/consumer to define the way they want to make the business happen. But as we say, each coin has two sides, with flexibility there comes a great responsibility of maintaining the sanity of the information, which these applications access and publish on the internet. The thought of increase of cyber terrorism and espionage incidents has triggered a worldwide uncertainty of ever increasing dependency of internet as a business medium. This paper hereby attempts to address these problems by taking references from the well-known attack vectors and thereby providing counter measures to identify, analyze and mitigate the problem. This paper primarily takes the reference from Open Web Application Security Project (OWASP) guidelines.

Key words: Web Application Attacks, Injection Attack agents, OWASP

I. INTRODUCTION

Web browser typically is an *n-tier* application, which can scale out from a single operating layer to multiple operating layers.

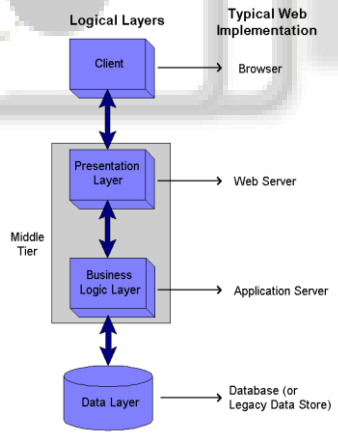

Fig. 1: Basic web application architecture

Typically, the anatomy of a web application spreads across multiple interaction layers (Figure 1. Basic Web Application Architecture).

A. Browser – Responsible for accessing the UI (User Interface) of the application

B. Middle Layer – Responsible for the interaction from Business to Presentation Layers

C. Database or legacy Layer – Responsible for acting as a data repository for storing the business data

There are individual risks attached at each and every layer and hence it is quite difficult to address to each and every loophole proactively, making the entire approach as "Reactive" than "Proactive". As per Open Web Application Security Project (OWASP)'s latest analysis, code injection attack is the most familiar and fatal attacks among the top ten web application vulnerabilities followed by broken authentication, session management and cross-site Scripting attacks.

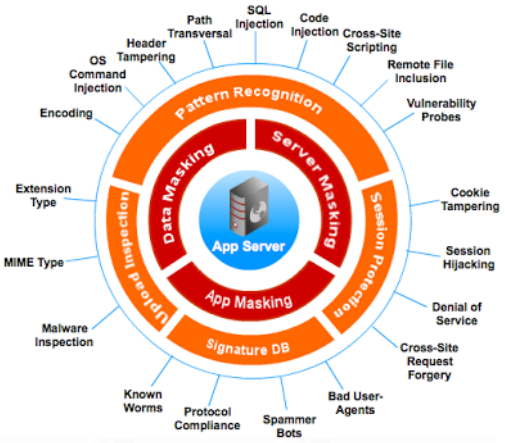

Fig. 2: Injection Attack agents

It is interesting to note that all above stated injection attacks are not singular in nature but they are most devastating when clubbed with multiple agents (Figure 2. Injection attack agents) there by making it as a more hard to detect and mitigate the same. To add to the pain, these attacks manipulate the trust relationship of the application and browser relationship more than just focusing the user input.

II. BACKGROUND

By definition, authentication is "the use of one or more mechanisms to prove that you are who, you claim to be and access will be given based in the same". Basically most of the web applications use client – server architecture, where the clients remain connected to the server making them highly session dependent applications. The primary challenge which has not changes since the origin is the attack vectors. Some of the identified ones are:

(1) Applications entering "legacy" state without change.
(2) The development of the applications are done on the outdated platforms
(3) Access control of the application isnot controlled
(4) Applications are not smart enough to judge the invalidated/malicious requests.

According to OWASP 2013 there are ten dominant categories specified for web application attacks in its latest 2013 release.

OWASP Top 10 – 2013 (New)
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 – Unvalidated Redirects and Forwards

Though this is the observed and analyzed list of application attacks but there are lot of attacks which gets un-noticed and thereby many more attacks do not get reveal to the outer world. As per 2014(3 year analysis) edition of "Verizon Data Security" Report, it is clearly evident that 2013 evidenced a rise of application attacks on a global front. Some of the main contributors in this rise were: POS Intrusions, Card Skimmers, and Cyber-espionage. Our scope in this research is to map the basic configuration level mistakes* with the attacks in a web application. Primarily from the bucket of OWASP2013 observations, we are focusing on

(1) Cross Site Scripting (CSS/XSS)
(2) Insecure object reference
(3) Cross Site Request Forgery (CSRF)

The main objective of this project is to make the audience aware of the ongoing web application threats and to guide them, to safe guard their critical personal data.

– Though OWASP highly talks about the application attacks but we understand that this is more about vulnerabilities which facilitates the possibility of an attack.

III. CHALLENGES

A. *Some Design Challenges:*

(1) TCP dependency,
(2) Data flow in clear text,
(3) Synchronous mode of authentication
(4) Browser compatibility issues
(5) Cookie dependency
(6) Use of known ports without port forwarding,
(7) Lack of multi-tier deployment.

B. *Some Development Mistakes:*

(1) "WebDev" is kept open,
(2) HTTP methods like Track/Trace are kept open
(3) Inheritance of platform vulnerabilities,
(4) Process breaks are not coded,
(5) Open ended procedure calls,
(6) Session identifiers are hardcoded and not nullified.

IV. CONCLUSIONS

(1) With more flexibility in the operations, comes with a fear of exploiting the Confidentiality, Integrity and Availability of the critical data
(2) In the offset of evaluating the scalability and flexibility, often security measures are rolled out.
(3) Security should be embedded in all phases of application build cycle than putting it as a topping.
(4) There is no 100% security but we can safeguard our data 100%, by understanding, "how much to retain? And how much to reveal?"
(5) Security cannot be overpowered with functionality but it should be adequate to retain the sanity of the data.
(6) Security does not discourage anyone from doing anything but encourage everyone from doing what they want in a "Controlled" environment.

Our future works will focus on giving the users and the business a pleasant, yet secure, feel of leveraging the power of "The Internet of things" to grow their business without compromising on the Confidentiality, integrity and Availability of the important / critical data. We also wish to formulate certain guiding principles, with their help of our analysis, which will help the community to phase off their uncertainty regarding the sanity of their data.

V. ACKNOWLEDGMENT

Sincere appreciation and warmest thanks are extended to many individuals who, in their own ways, have inspired us to complete this project.

Firstly, we are thankful to our principal DR. M. Z. Shaikh for his help. We are extremely grateful for his friendly support and professional attitude. Secondly, we express our heartfelt gratitude to our Head of Department Prof. D. R. Ingle & project coordinator Prof. Rahul Patil of Computer Department for their help and support.

Thirdly, be believe, this was because of the guidance and supervision of our esteemed professor, guide and mentor, Prof. Sulakshana Mane, we have been able to take up our project to this stage.

We also convey special thanks to all staff members of Computer Engineering Department for their support and help.

Last but not least, we are very much thankful to our friends who directly or indirectly helped us in completion of the project report.

REFERENCE

[1] https://www.owasp.org/index.php/Top_10_2013-Top_10
[2] Nilesh Kochre, Satish Chalukar, Santosh Kakde, "Survey On SQL Injection Attacks And Their Countermeasures ", International Journal Of
[3] Computational Engineering And Management, Vol -14, October 2011
[4] https://www.owasp.org/index.php/Top_10_2013A2 Broken_Authentication_and_Session_Management
[5] Hossain Shaihriar and Mahammad Zulkernine, "S2XS2: A Server Side Approach To Automatically Detect XSS Attacks", Ninth International Conference on Dependable, Automatic Secure ReferencesFor how to output encode properly, read the
[6] For how to output encode properly, read thehttps://www.owasp.org/index.php/XSS_(Cross Site Scripting)

