Metadata: Last updated 11/05/18
Total number of pages so far: 52
Two page format.

SYNOPSIS

MINOR PROJECT

BROWSER SECURITY

Jishan Shaikh 161112013
Ankit Chouhan 161112051
Ankit Chaudhary 161112048
Mansoor Sediqi 161112101

This page intentionally left blank

Maulana Azad National Institute of Technology

(An Institute of National Importance)
Bhopal – 462003 (India)

Department of Computer Science & Engineering

S Y N O P S I S

M I N O R P R O J E C T

Topic based on Browser Security

Submitted by : Jishan Shaikh (Scholar no. 161112013)
Ankit Chouhan (Scholar no. 161112051)
Ankit Chaudhary (Scholar no. 161112048)
Mansoor Sediqi (Scholar no. 161112101)

Supervisor : Prof. Deepak Singh Tomar
Department of Computer Science & Engg.
MANIT, Bhopal (India)

Dated by : November 5, 2018 (Monday)

Subject : Minor Project
CSE-319, V Sem. (B.Tech. in CSE)

Session : Odd Semester 2018 (Year 2018-19)

This page intentionally left blank

Maulana Azad National Institute of Technology

(An Institute of National Importance)
Bhopal – 462003 (India)

Department of Computer Science & Engineering

Declaration of Authorization

We hereby declare that, all the work produced in this document is solely created by us as a part of underlying project based on Browser Security under the supervision of **Prof. Deepak Singh Tomar** with proper citations (In project report). We are fully aware that any violation of institute's work ethics or department's legacy will clearly made us victims, and we'll then have to bear the charges produced by institute/department to us according to their etiquette. We also declare that we've not copied anyone's work to be named as our work. All the sources are referred properly at the end of the section for appropriate authorization.

This declaration is the proof that the work produced as a complete document is first handed, and is not sent anywhere with confidentiality, before submission of it in the department/to the supervisor.

Group Members

Jishan Shaikh (1611112013)
Ankit Chouhan (1611112051)
Ankit Chaudhary (1611112048)
Mansoor Sediqi (161112101)

Supervisor

Prof. Deepak Singh Tomar
Dept of Computer Science & Engg.
MANIT, Bhopal (462003).

This page intentionally left blank

Maulana Azad National Institute of Technology

(An Institute of National Importance)
Bhopal – 462003 (India)

Department of Computer Science & Engineering

Certificate

This is to certify that **Jishan Shaikh, Ankit Chouhan, Ankit Chaudhary, and Mansoor Sediqi** all students of third year B.Tech. (Computer Science and Engineering) batch of 2016-20, are taking Minor project on a topic based on **"Browser Security"** in partial fulfillment of their Bachelor of Technology degree in Computer Science and Engineering.

Project Coordinators

Prof. Sanyam Shukla
Prof. Rajesh Wadhwani
Dept of Computer Science & Engg.
MANIT, Bhopal (462003).

Project Guide & Supervisor

Prof. Deepak Singh Tomar
Dept of Computer Science & Engg.
MANIT, Bhopal (462003).

This page intentionally left blank

Contents

Preface

This project based on Browser Security is one of the academic projects we (as a team of 4 members) take as Minor Project (CSE-319) in III year (5th Sem) of Bachelor in Technology (Computer Science & Engineering), here at Maulana Azad National Institute of Technology, Bhopal. This project was started in May 2018 and merely ends at the end of academic year 2018-19 (March 2019).

This detailed synopsis is organized into small chapters, sections, subsections, etc. Snapshots and pictures are taken for clear idea of what's going on. We've tried to cover all information regarding project to be assemble in synopsis, but still Plan document, Preliminary RMMM plan does not find their place in the synopsis. Further they can be merged into report in future as per requirements and their specifications.

The foundations of project started in August 2018, when new session starts and topic of project decided. Some points such that project management, scalability, risk analysis, version control, aims, etc. were discussed primarily. The major aims set at beginning of this project was -

- To gain the technical knowledge and experience and practical security and working of browser, and related technologies.
- A clear idea of working title; incorporating traditional accent but with some modern software engineering principles, practices, and standards
- Improving team skills under fully supervision (Project management including risk analysis and Soft-skills)
- To present a final result implementation, its analysis, and evaluation with proper documentation/project report.
- To present the work in form of a publication ready research paper.

Although the aims were set primarily and broadly, we as a team collaborate on multiple issues and try as much as we can to be stick on aims along with healthy communication from the supervisor. This synopsis is neither a fully formal synopsis nor a complete preliminary report, but it inherits many attributes of both in it. Here are some highlights of the project -

- A single document having combination of documentation, project report, reference manual, tutorial, user manual for the project output.
- Practical implementation of various attacks exploiting current browser's vulnerabilities.

- Proposition of new model based on comparative analysis of modern browsers and their parameters in form of new model with proper architecture and security diagrams.
- Incorporation of Software Engineering elements such as principles, practices, life-cycle, etc. to the project.

Most of the collected contents are properly referenced*. If any of work is not, please feel free to notify us via emails.

It is also planned to place this project on Github.com soon, where the version control of project is monitored properly and subject experts (actually everyone) will get authority to analyze, evaluate, and even collaborate on this project. This project will be licensed under Creative Commons ShareAlike Non-distribution 4.0 International licence 2018. From software engineering practice we know Software prototype maintenance never ends as no research/software is perfect, so the modifications to the first version/release of this project will be completely moved to github. Feel free to checkout the readme.md file at Github. Althrough we are saying output as a software tool, it might not be a fully functional SOFTWARE. We can say it as a working prototype of software which can further be enhanced in form of a software, and will surely be ready for industry and production.

It is very memorable to work on such a project with such a team and such a mentor. No work is perfect, remember to write your views to us! Healthy criticism, suggestions, feedback, and even your opinions are always welcome to enhance this project either at github or via emails.

Acknowledgments

With due respect, we express our deep sense of gratitude to our respected and learning guide, mentor, supervisor, instructor **Prof. Deepak Singh Tomar**, for their valuable help and guidance. We are thankful for the encouragement and motivation that he has given us in initialization phase boost to this project successfully. Their rigorous evaluation and constructive criticism were of great assistance. It is imperative for us to mention the fact that this project would not have been accomplished without the periodic suggestions and advice of our supervisor.

We are also grateful to our respected director **Dr. Narendra Raghuvanshi** for permitting us to utilize all the necessary facilities of the college.

Needless to mention is the additional help and support extended by our respected Head of the department- **Ms. Meenu Chawla**, in allowing us to use the departmental laboratories and other services on the period of time and also maintaining a healthy discipline in management, tutorials, eduventurisation, and other activities.

We are thankful to all the other faculties, Professors, Associate Professors, Assistant Professors, staff-members, teaching assistants, laboratory attendants, seniors, and our fellow branch mates of our work culture for their kind co-operation, periodic evaluation, help, and support. We thank all others who directly/indirectly involved in this project and will make it a successful. We also thank all other teams for a maintaining an active communication aside of healthy competition and sportsman's spirit.

We would also like to express our deep appreciation towards our family members for making us such kind to do 'more' by providing the much needed support and encouragement from all aspects of life. At last, we recall our gratitude to the one eternal almighty: The God.

This page intentionally left blank

Abstract

Security issues are critical in the software life cycle as it directly or indirectly affects the key functionalities of software and expose privacy privileges of user regarding data and information. These are primarily addressed during development phase of its life cycle, where functionality is usually given higher priority than security. An insecure software which is consistent does not worth much. Hence, there must be proper balance between the two. Special emphasis must be therefore given to security in project management activities for risks, quality assurance, configuration, estimation, planning, etc.

This paper gave an introductory idea for maintaining security of browsers with a clear, concise understanding as well as existing proper tools and techniques. A practical comprehensive methodology *(Agile methodology)* is adopted for that purpose. This project also focuses on vulnerability detection and testing of a web browser and web application. A software security framework (SOSEF) is then proposed as an application/tool/web browser extension for effective security management which includes newly developed browser security model, its security architecture, assessment and evaluation.

Index terms: *Browser, Security, Browser-Security.*

This page intentionally left blank

List of figures

This page intentionally left blank

List of tables

This page intentionally left blank

Section 1
Introduction

1.1 Introduction to Web browser

A web browser is an application software stack that works as an interface between server and client with utmost security.

1.1.1 Components of a web browser

It comprises of following utility modules/components -

- **User Interface:** This is the interface with which client communicates with server. This typically comprises of web-pages, web-applications, or simple documents/media according to its usage and application. Usually web-pages are developed with the help of HTML (Hyper Text Markup Language), CSS (Cascading Style Sheets), JavaScript, or other frameworks such as Django, AJAX (It is special related to browser security because of its fundamental *XMLHttpRequest*[1] object), React, and many more. It is usually graphic rich.

- **Browsing engine:** This component acts as connecting component of User interface of client and rendering engine. Rendering engine sends rendered data to browsing engine, it simply formats it to visually good looking and well organized format.

- **Rendering engine:** It renders the JavaScript data, along with processing of CSS file included in user interface. It is also connected with Networking components i.e. all the networking activities seen inside the user interface window passes through rendering engine before, for processing or specifically rendering, hence the name.

- **Networking:** Web browser is a social application, it means it is connected to world wide web using URL (Universal Resource Locator) through internet with the help of networking.

- **User interface back-end:** It comprises of various back-end files written with either PHP, SQL (Or any alternatives of it). It usually have codes for specific purpose for implementation in user interface.

1 *The main feature of this class enable a web-page to take data from server without clicking the refresh button i.e. a static paper is also capable of sucking data from server without any user activity.*

- **JavaScript virtual machine or JavaScript interpreter:** This is the component where JavaScript scripts are interpreted, and are bind-ed with byte code to execute on browser engine. There are multiple flavors of JavaScript available in the industry, each of which having its own advantages and disadvantages.

- **Data Storage or Database:** Each browser has its own data storage/database usually not accessible by end-user / client. It has its preferential applications such as buffer storage, file storage (from server), cookie storage, History and Bookmarks storage, Settings preferences, etc.

1.1.2 Working of a web browser

The main work of a web browser is as an interface between client (end-user) and server. But, its not that much easy. It is pretty well concerned with DNS server with website address to IP (Internet Provider) address translation, and fetching data from server with utmost security and encryption.

The working of a web browser can be summarized in following picture -

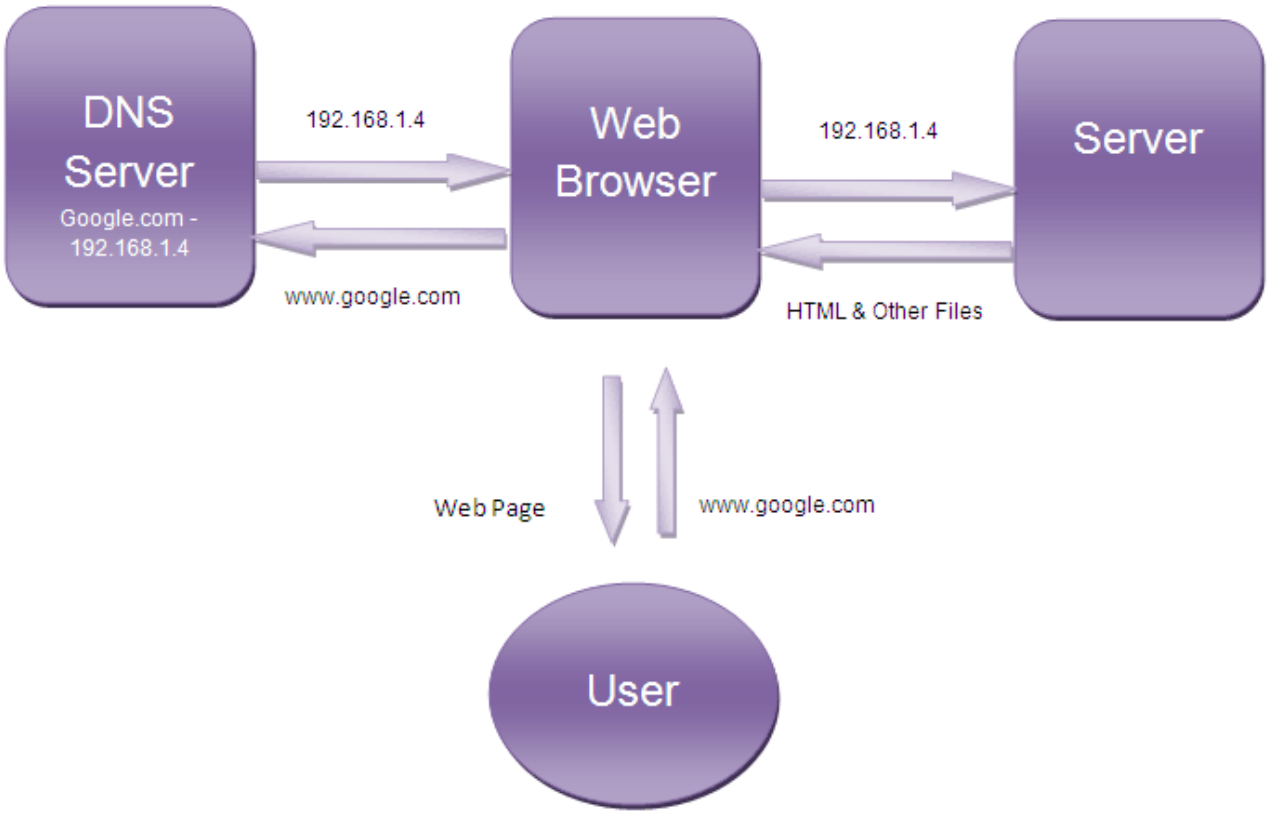

Figure 1: Working of a web browser. (Reference seng13.wordpress.com)

1.1.3 Architectural diagram of a web browser

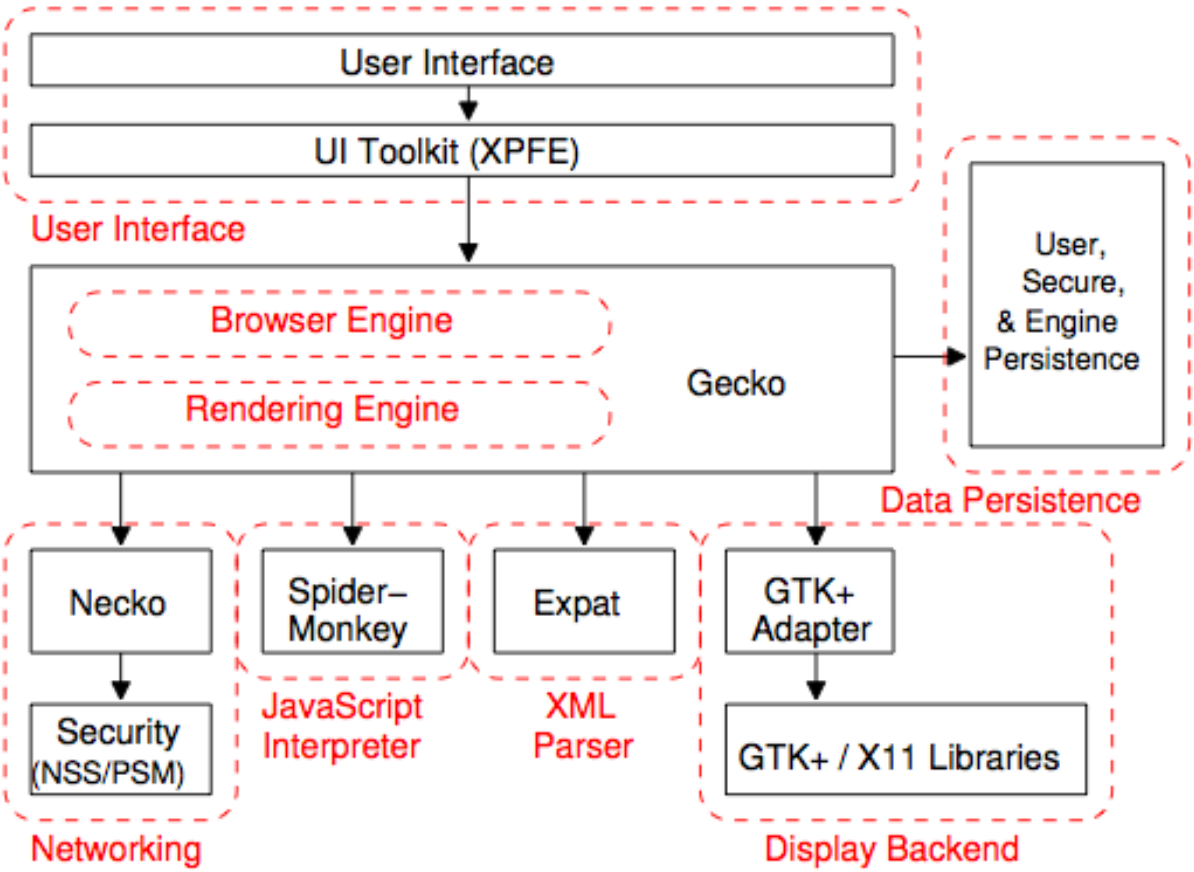

Figure 2: Architectural Diagram of a web browser (Reference engineersgarage.com)

1.2 Importance of Browser Security

One may say that, “Okay! A browser works as interface between me and server, and the data transferred between me and server are just encrypted files, so why would I care?” The answer of his/her question is very straightforward: “Importance of Information/Data security”. The risk of security of information/data today is as high as that of money. It affects privacy, confidentiality, authentication, and various other fundamental security concerns. Today, web is not just a collection of few web-pages shared between a “good audience”. Instead, web is a network of interconnected web-pages which are probably billions in number and is accessible to everyone using URL and internet. Electronic mails, Chat rooms, auctions, shopping, news, banking, Medications, etc. all are done at web. Why shouldn’t anyone care? Web contains person’s name, address, personal identification number (Aadhar Number or Social Security Number), Credit (and Debit) card numbers, Phone number, Date of birth, along with his/her favorite chocolate/color may be used for accessing/manipulating financial statements, trade, etc. Identity theft, script kidding,

and full disclosure antics are far way behind today, as Control of a nationwide examination, as well as state wise power grid control, hydroelectric dams operations, prescription fillings often happens online which are undoubtedly are of serious concern. And yes, browser is the interface to connect client and server, **with utmost security.**

1.3 Differences with Browser Security

Browser security usually concerns with its components, its working, and its vulnerabilities e.g. User Interface security (not of much concern), Network Security, Data Persistent Security, Browsing and Rendering engine security, Back-end security, as well as encryption and HTTP protocols.

1.3.1 Database security

Usually concerns specifically with data stored in a particular database e.g. of users including his/her personal information with some confidential information such as passwords, Bank statements, Transactions history, etc.

1.3.2 Network security

Usually concerns specifically with data transferring through a medium or communication wireless or with wires/electrics. It also has potential danger of attacks. threats, and anomalies.

1.3.3 Software security / Web Application security

Usually concerns specifically with software architecture, functions/modules, its objects, and its application based threats.

1.4 About the project

1.4.1 Aims and Objectives of the project

- To gain the technical knowledge and experience and practical security and working of browser, and related technologies.
- A clear idea of working title; incorporating traditional accent but with some modern software engineering principles, practices, and standards.
- Improving team skills under fully supervision (Project management including risk analysis and Soft-skills).

- To present a final result implementation, its analysis, and evaluation with proper documentation/project report.
- To present the work in form of a publication ready research paper.
- To learn concepts of security to apply them to web browsers.
- To enhance team work and individual responsibilities by following Software Engineering principles and standards.

1.4.2 Scope of the project

- Tool/framework can be used theoretically for testing of web browser.
- With a few feasible modifications, constructed tool can be used for testing the security aspects of a web browser as an extension, add-on, or as a plugin.
- Research oriented nature of project will definitely leads to better technological developments.
- Maintenance phase could compromise with further modifications and enhancements.

1.4.3 Motivation of the project

- Learning and applying the conceptual and fundamental knowledge to a work based on a vast technology i.e. Security and Browsers.
- Contribution to industrial-research community.
- To have practical “hands on” experience with security and testing.
- Enhancing team and soft skills of members, with research.
- Learning by doing; modern technologies and their challenges.

1.4.4 Project output and deliverable (s)

- Project report (with other documents including synopsis, plan document, design document, etc.).
- Seminar or presentation based on Browser security.
- A clean white paper on Browser security. Other research papers may include testing of browsers, comparison parameters of browsers, etc.
- **Proposition of new model with proper security architecture applicable to modern browser and development of a tool/framework based on it.**

1.5 Problem Statement

Browser security usually concerns with the security its components, its working, its interface, its networking and not whatsoever. Browser security is of utmost importance in modern culture and life style (See sub-section 1.2). Security analysis, comparison analysis, vulnerabilities detection, feature improvements, etc.

are therefore increased in demand with respect to web browser development. Preventing attacks by filling loopholes is a hard task- from a practitioner point of view as well as researcher point of view. Analysis of browser security is current primary statement of this project.

1.6 W_5HH Principles

Why is the project being developed?

- To gain the technical knowledge and experience and practical security and working of browser, and related technologies.
- A clear idea of working title; incorporating traditional accent but with some modern software engineering principles, practices, and standards.
- Improving team skills under fully supervision (Project management including risk analysis and Soft-skills).
- To present a final result implementation, its analysis, and evaluation with proper documentation/project report.
- To present the work in form of a publication ready research paper.
- To learn concepts of security to apply them to web browsers.
- To enhance team work and individual responsibilities by following Software Engineering principles and standards.

What will be done?*

**See Proposed work for this section.*

When will it be done?

- The project is supposed to be completed by March 2019.
- First release of report/paper/tool will be expected to be April 2019.
- Maintenance of that will be then referred to Open Source community (Github).

Where are the organizations located?

- This project is a single organization, single team project located at MANIT, Bhopal.

Who is responsible for the desired functionality?

- Each member has assigned his/her own responsibility regarding projects in sections and of risk analysis, they have to follow their rules.
- Team members are atomic elements of project and are solely responsible for their work.

- Timely supervision is responsibility of the instructor.

How will the job be done technical and management side?

Technical issues:

- For existing tools, Linux is preferred operating system.
- Tool can be constructed either among web application, android application, browser extension.
- Testing of vulnerabilities and loop holes by various plugins and add-on(s).

Management issues:

- Risk analysis.
- Cost estimation.
- Time scheduling.
- Project monitoring and controlling.

How much and what type of resources are needed?

- Human resource of 4 team members
- Monetary resource for software and hardware resources (if applicable).
- Hardware resource of a computer with required software installed.
- Software resources used (if applicable).

1.7 Summary

Web browser working and components are explained with the help of its architectural diagram. The question addressed thereafter about its relative importance and is compared to network, database, web application/software security. After there is brief discussion about the project and its structure using W_5HH principles.

▪▪▪

This page intentionally left blank.

Section 2
Literature Review and Survey

2.1 Top vulnerabilities and their possible implementations

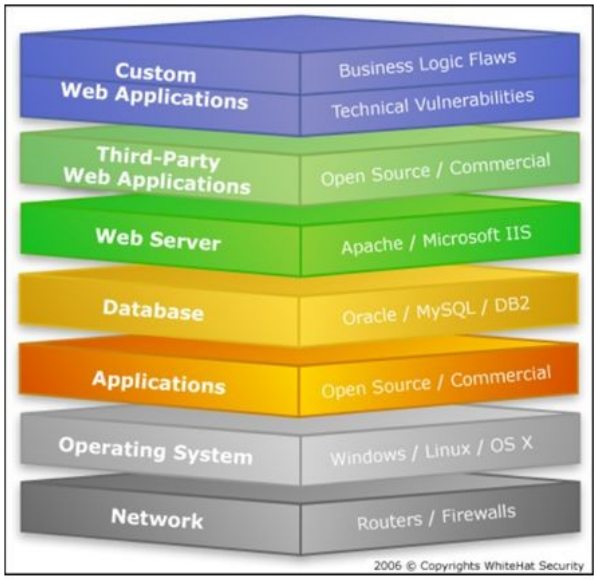

Figure 3: Vulnerability stack at server side.
(Reference: WhiteHat Security).

According to OWASP[2], followings are the top 10 vulnerabilities (Application Security Risks) that an attacker/hacker may exploit in order to get an attack successful -

- **Injection :** Injection flaws occurs when untrusted/unreliable data is sent to client machine using injection of malicious script usually database query language such as SQL, MySQL, NoSQL, MongoDB, PHP MyAdmin, etc. It can be prevented using parameters in SQL query inside user interface code.

- **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.

2 *The Open Web Application Security Project (OWASP) is an open community dedicated to enabling organizations to develop, purchase, and maintain applications and APIs that can be trusted.*

- **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.

- **XML external entities :** 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.

- **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.

- **Ill posted security configuration :** 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.

- **XSS (Cross Site Scripting) :** XSS flaws occur whenever an application includes untrusted 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.

- **Insecure de-serialization :** Insecure de-serialization often leads to remote code execution. Even if de-serialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.

- **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.

- **Insufficient logging and 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.

2.2 Browser attack scenarios

Followings are some scenarios/types where browser attack is possible -

- **Man in the browser attack :** It involves an attacker either secretly tapping into a browser to gather data, modify websites and/or manipulate requests, without the knowledge of the user. It cannot easily be detected by server as attack happens at browser (i.e. client side). Web integrating module helps user to create a security barrier against man in the browser attack. Similarly there exists a server side measure to detect MitB attack namely out of band verification. User behavior analysis may be helpful in detection of MitB attack.

- **Cross Site Scripting attacks :** Cross Site Scripting attacks (XSS) s a code injection attack that allows an attacker to execute malicious JavaScript in another user's browser. In this attack, attacker did not directly attack user/client instead he/she finds a crucial vulnerability in the browser/application from which he/she can inject malicious JavaScript code into the system. At first JavaScript code seems to be free from malicious concerns, but since JavaScript have access of user cookies, sending HTTP requests using *XMLHttpRequest*, and other DOM manipulation activities, therefore JavaScript scripts may be as malicious as injection attacks. The risks involves theft of cookies, key-logging, and phishing.

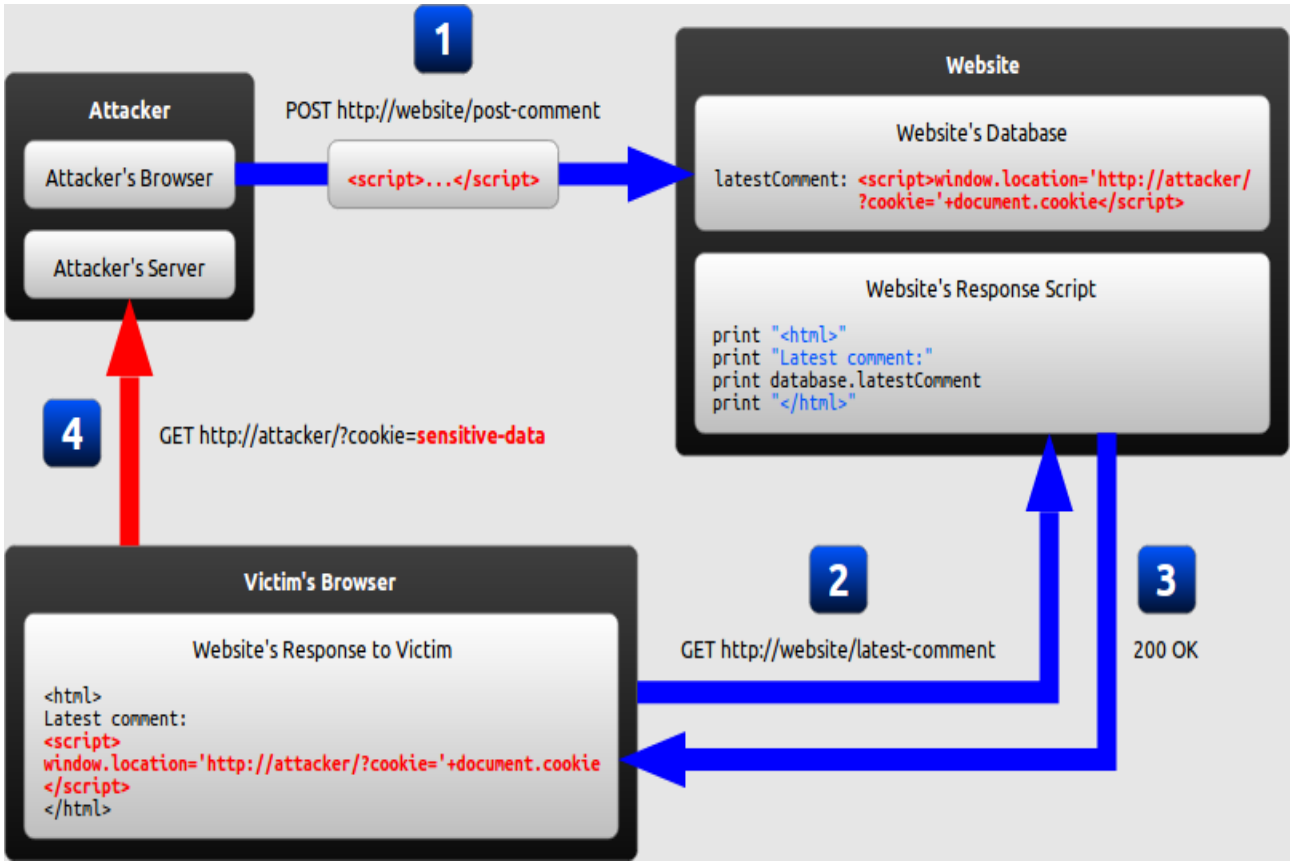

Figure 4: Example working of an XSS attack (Reference excess-xss.com)

- **Injection attacks :** Injection attacks refer to a broad class of attacks that allow an attacker to supply untrusted code/command to a program, which gets processed by an interpreter/virtual machine as part of a command or query which alters the course of execution of that program. Injection attacks are among the oldest and most dangerous web application and browser attacks. They can result in data theft, data loss, loss of data integrity, denial of service, as well as full system compromise.

- **Denial of Service attacks :** Denial of Service attacks abbreviated as DoS attacks are attacks in which a particular service is denied for a finite time or delayed. It is common attack in modern web browsers where a particular server of a website is exploited by some vulnerabilities of web browser. Some mechanisms for perfoming DoS attacks are webStorage Fill Up, webWorker Botnet, clientDoS, server-sent Event Botnet, and vibration Attack.

- **Request Forgery :** It is specific type of an attack where an authorized user send unrestricted commands to website which the website is not able to detect malicious and execute that commands on the website. It is also known as *one-side attack*. Some particular mechanisms for requesting forgery are Cross Site Request Forgery (CSRF or XSRF) with CORS, WebSocket Hijacking, and CrossPrinting.

- **Social Engineering attacks :** Social engineering attacks are those types of attacks that usually involves attack classes of phishing, pretexting, baiting, and tailgating. Phishing involves seeking of confidential information of user/client. Pretexting is another variation of social engineering attack where attackers focus on creating a good pretext, or a fabricated scenario, that they can use to try and steal their victims' personal information. Baiting is similar to phishing but involves promise of an etiquette item in replacement. Tailgating is another type also known as 'piggybacking' that involves someone who lacks the proper authentication following an employee into a restricted area.. Following are some measures of Social Engineering attacks - Phishing, Notification Autocomplete, Input Form Leak, and Fake Telephone Call.

2.3 Attack tree construction

Attack tree is hierarchical visualization of possible attacks on a system from external most vulnerable module to internal most vulnerable module. This depicts all the possible attacks that exists on a system/browser. It helps to find new vulnerabilities, as well as to organize existing vulnerabilities so as to give a clear pictorial view of security of browser. Number of attacks possible in any system/browser will then simply be number of nodes in that tree, and edges will depict functional/logical/physical hierarchy of modules where that attack is possible.

The procedure of constructing an attack tree is -

1. Determine all the existing vulnerabilities in a system/browser.
2. Predict possible attacks based on each vulnerability.
3. Organize all attacks (as nodes) to build a tree with edges depicting application hierarchy.

Here is an example of attack tree for a virus attack on system -

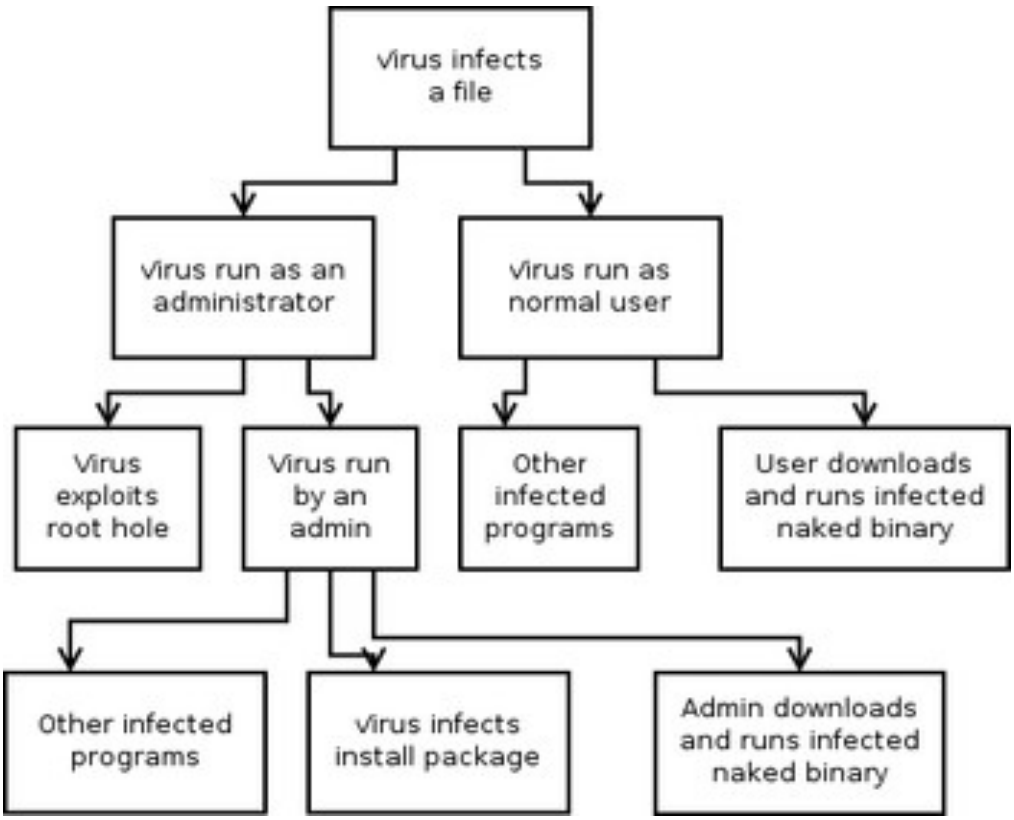

Figure 5: Sample attack tree - Virus Attack (Reference Wikimedia commons).

2.4 Implementation of Cross Site Scripting attacks (XSS)

Followings are some scenarios where Cross Site Scripting attack is possible -

- **Using HTTP URLs :** Examples -
  - www.serverspy.net/site/stats/mods.html?g=0%22%3E%3CSCRIPT%3Ealert(%22kefka%20was%20here%22)%3C/SCRIPT%3E
  - http://blogshares.com/blogs.php?blog=%3Cscript%3Ealert(document.cookie)%3C/script%3E
  - www.php.com/include/search/index.php?where_keywords=%3Cscript%3Ealert%28document.cookie%29%3C%2Fscript%3E
  - www.whiteacid.org/misc/xss_post_forwarder.php?xss_target=http://www.advfilms.com/search.asp&search=%3Cscript%3Ealert(String.fromCharCode(88,83,83))%3C/script%3E
  - www.vodafone.com/site_search_results/0,3062,CATEGORY_ID%253D200%2526LANGUAGE_ID%253D0%2526CONTENT_ID%253D0,00.html?section=all&company=all&KWD=%22%3B%3C

%2Fscript%3E%3Cscript%3Ealert%28%27XSS%27%29%3B%3C%2Fscript%3E%3Cb+&submitButton=%C2%BB

- **Exploiting JavaScript codes using <form> element of HTML :** Example -

```
<form action="http://www.example.com/login.php">
      <input type="text" name="username" value="foo" />
      <input type="text" name="password" value="bar';INSERT INTO user (username, password, accesslevel) VALUES('hackeradmin', 'evilhax0r', 99); SELECT userid FROM user WHERE username = 'foo">
      <input type="submit" />
</form>
```

- **Injecting malicious URL to inappropriate elements/links/media :** Example -

```
<a src = "https://www.exampleWebsite.com/preview.cgi?comment=<script>MALICIOUS%20SCRIPT</script>">Anniversary Images</a>
```

2.5 Open-Source tools to detect/prevent XSS

Following are some existing tools/techniques to discover XSS in a web browser -

- **Burp suite and Burp proxy :** This is one of the most useful, functional, and practical open-source tool for detection/prevention of XSS attacks. It is available in Linux, Windows, OS x since it is based on Java Runtime Environment (JRE). It comes in two editions – Community edition & Professional edition of which community edition is free.

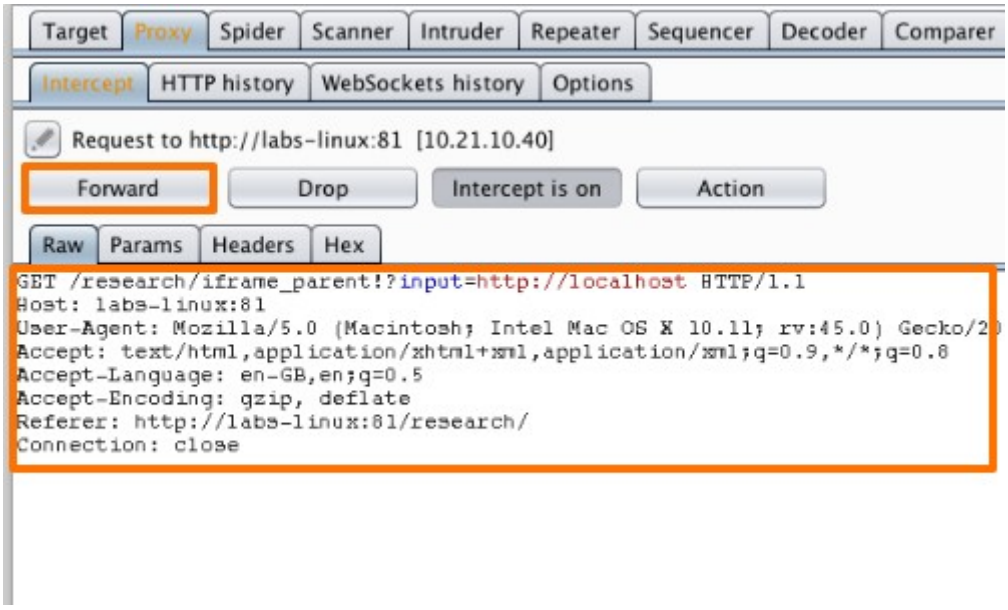

Figure 6: Burp Suite's interface provided by PortSwigger.

- **Dynamic HTML debugging; using extensions/plugins :** Dynamic HTML debugging means finding bug in HTML page at run-time. Several extensions and plugins are developed open-source for that purpose. For example - *IDA pro*, *Olly Debugger*, *GDB* are some extensions present in Mozilla Firefox browser. These extensions are generally used to research malware, examining protection scheme, and local vulnerabilities. But, there is one drawback that they can not be used to probe web applications because they are made to analyze dynamic HTML pages.

  JavaScript is based on object oriented features; attackers uses its feature to inject user scripts or XSS attack to web browser. This feature is also modeled as Document object Model (DOM). DOM based XSS attacks are relatively hard to detect, hence special extensions are used for this purpose. For example - DOM inspector, Webdeveloper, FireBug (web application debugger) and FireBug Lite are some Mozilla Firefox extensions for debugging a web application in web browser.

- **Analyzing HTTP traffic; using extensions/plugins :** Analyzing the traffic passes through HTTP can give us an estimate of level of XSS attack in a web browser. Various extensions/plugins are also available for analysis of HTTP traffic. For example – Live HTTPHeaders, ModifyHeaders, and TamperData are some Mozilla Firefox extensions.

- **GreaseMonkey scripts :** There exists an old technique that involves installation of application on local system and testing it against XSS attacks, but since technology develops with AJAX (and its *XMLHttpRequest* object); this technique fails as AJAX fails to load in local system. Here comes the

GreaseMonkey scripts. It is a dynamic tool that can run, modify, alter, and change the application at run-time. *GreaseMonkey* is available as Mozilla Firefox extension. It allows user to encode JavaScript code into image tag. *PostInterpreter* and *XSS Assistant* are some of GreaseMonkey based extensions available in Mozilla Firefox browser. They helps in running/disabling custom user scripts/XSS scripts. Simple web spiders and vulnerability scanner scans only simplest XSS vulnerabilities; for persistent and DOM based XSS vulnerabilities advanced technique are used for detection/prevention of such type of vulnerabilities e.g COM and XPCOM models are developed for the same.

- **Bookmarklets :** Bookmarks in a web browser generally stored in form of title, meta-information, description, URLs, etc. Some common URL prefix are: http:, https:, ftp:, file:, javascript:. The javascript: protocol is a simple way for storing multiple JavaScript expressions in a single line. This type of technique is widely used among AJAX developers. And it can simply be stored in <img src = "-"> tag. Bookmarlets are customized bookmarks with specific user scripts. Malicious scripts of JavaScript may be written in those Bookmarklets, and client may easily be attacked on clicking those bookmarks. There exists a feature of Mozilla Firefox that enables autorunnable bookmarks, this can be of serious activity.

- **Using Technika :** Technika is another tool from GNUCITIZEN that allows user to easily construct bookmarklets and automatically execute them, imitating the functionalities of GreaseMonkey. Technika is very small and integrates well with the Firebug command console, which can be used to test and develop custom bookmarklets. This extension can be found at http://www.gnucitizen.org/projects/technika.

2.6 Commercial XSS scanners

Here are some scanners used to scan XSS scripts in a web browser and web application -

- Sanctum's Appscan
- Rapid7's Nexpos
- Whitehat's Arsenal
- OWASP's Webscarab

The familiarity of these scanners may be a part of this project later on.

2.7 Methods of prevention of browser attacks

"Prevention is better than cure." Here are some of prevention measures against browser attacks -

- For prevention against social engineering attacks it is recommended to take special care about fake/untrusted emails, privacy protection, anti-virus software, etc.
- For Injection attacks, make sure proper protection/prevention/avoidance of vulnerability threats based on databases, and hosting server data.
- Use Virtual Private Networks (VPNs) for surfing internet for privacy protection.
- Make use of trusted web browser which provides frequent security patches and upgrades.
- Understands all types of attacks and take precautions for them correspondingly.
- Risk assessment procedures and security assessment audits must be seriously followed in analysis phase of development.
- Protection towards OWASP's security threats must be incorporated in web application / browser.
- CERT advisory should be taken special care at the time of development. Recent updates and security updates must also be taken care of.
- Disabling JavaScript and Flash player reduces threat of exploitation.
- Minimizing use of browser extensions and add-ons (plugins).

2.8 Summary and references

In this section, vulnerabilities and attacks are discussed alongside with special attack: Cross Site Scripting (XSS) with its detection/prevention tools. At last prevention measures of attacks are listed a few.

▪▪▪

Section 3
Proposed work

3.1 Problem statement

Browser security usually concerns with the security its components, its working, its interface, its networking and not whatsoever. Browser security is of utmost importance in modern culture and life style (See sub-section 1.2). Security analysis, comparison analysis, vulnerabilities detection, feature improvements, etc. are therefore increased in demand with respect to web browser development. Preventing attacks by filling loopholes is a hard task- from a practitioner point of view as well as researcher point of view. Analysis of browser security is current primary statement of this project.

3.2 Methodology

The methodology adopted in the project is simple and is easy to implement. **Functional research and Object oriented methodology** constitute a major portion of methodology. Project is divided into individual's tasks. Tasks are then taken as initial problem and solved individually and have to integrate them at last. Objectives are set first and then we tried to fulfill all objectives asserted.

We've first implemented preliminary Entity-Relationship diagram in 2 platforms namely- MySQL Workbench (Windows 10) and LibreOffice Base (Ubuntu 17.10). We then intuitively analyzed which platform is more suitable for this particular schema and required functionalities. After that we enhance that schema to further more relations, entities, relationships, forms, queries, etc.

The Project development model we used in project development is closely related to hybrid association of waterfall model, Rapid application development (Rapid delivery of project, time to time), Agile methodology (Simplicity and Swift evaluation), Some Extreme Programming practices, and 4^{th} generation tool usage (Usage of CASE tool: LibreOffice Base/Draw, etc.). The involvement of multiple model element in single project leads to higher level of customization, better management of project, better requirements adaptability, and improved rigidity of overall management procedure and development life cycle. Yet on a broad way, this project is closely resembles with **Agile methodologies**.

We've often used many Software engineering principles in the project to make it well organized, formal, and more productive. Use of Open-Source tools helps us to

contribute Free software foundation and Open-source community. We'll model the project structure into various design diagrams amongst Entity-Relationship diagrams, Use-Case diagrams, Sequence diagrams, Data flow diagrams. An Architecture diagram will also designed for an overview of tool which will be developed later on.

3.3 Work description

3.3.1 Major tasks

Followings are some works (papers) which may be created / taken as a part or in a whole, in this project -

- Understanding the working of some Open-Source engines for detection/prevention of vulnerabilities or user-scripts to avoid an attack.
- A complete guide of testing the web browsers against vulnerabilities.
- Unified comparison parameters for web browsers and comparisons of modern web browsers based on that.
- Proposition of a new model with a strong security architecture and development of a tool/extension based on that.
- Development of an engine for browser vulnerability detection and prevention.
- Possible detection of new vulnerabilities/bug in a modern browser such as Chromium, Firefox, or Tor (Big Bug Hunting).
- Case study of a particular web browser including security architecture, existing vulnerabilities/bugs, updates, bug-hunting, attack prone from plugins/extensions, happened attacks, countermeasures, and flaws in features with recommendation.
- A clean white-paper or a research article (may be a survey paper) on topic "Browser Security", or something like that.
- Advanced techniques for detection of user-scripts and XSS attacks.
- Performing security audits of web applications.
- DFM (Document *Function Model) based XSS (Cross Site Scripting).
- Development of complete browser attack tree, based on various attack scenarios.
- Automating the browser to perform HTML rendering through *a new model (instead of COM or XP-COM).
- Proposal of a new web browser architecture based on existing vulnerabilities, threats, and attacks.
- And, many more…

3.4 Summary

Possible output papers (white-papers, review papers, research papers, survey papers) are described as topics. They are sole collection of author and will certainly be included more than 2 papers in the project output.

▪▪▪

This page intentionally left blank

Section 4
Tools and Technologies

4.1 Software and Hardware requirements

Minimum System/Software/Hardware requirements :

- Windows/Linux/Mac OS/Chromium OS.
- Memory requirements: 64 MB (RAM).
  - Recommended 128 MB.
- Secondary memory requirements (Hard disk): 10 GB (ROM).
  - Recommended 256/512GB
- Latest versions of web browsers - Chrome, Chromium, Firefox, Opera.
- Writer and diagram designing software such as MS Visio or LibreOffice Draw.
- Clock Speed: 866 Mhz
- Virtual Memory: 32 bits (minimum).
- Cache Memory: 512KB, etc.

Resources Usage :

- Operating System: Windows 10/Ubuntu 18.04/Kali Linux 18.0.3
- Memory: 8 GB (RAM)
- Secondary memory: 1 TB (ROM)
- Firefox, Chrome, Chromium, and Opera web browsers.
- Microsoft Word 16/ LibreOffice Writer 5.4 (Linux).

Functional requirements for the tool :

- Cached data storage and retrieval.
- Proper User Interface as in accordance with use-case diagrams.
- Customized testing functionalities in accordance with the proposed model.

4.2 Open Source tools from OWASP

General Testing tools

- OWASP ZAP.
- OWASP WebScarab
- OWASP CAL9000

- OWASP Pantera Web Assessment Studio project
- OWASP Mantra – Security Framework
- Spike
- Burp Proxy
- Odysseus Proxy
- Webstretch Proxy
- WATOBO
- Firefox LiveHTTPHeaders
- DOM Inspector
- Grendel Scan

Testing of JS security, DOM XSS

- BlueClosure BC Detect

Testing AJAX

- OWASP Sprajax Project

Testing SQL Injections

- OWASP SQLix
- SQLNinja
- Absinthe 1.1
- Pangolin

Testing Oracle

- TNS Listener tool
- Toad for Oracle

Testing SSL

- Foundstone SSL
- O-Saft

Testing for Brute force password

- THC Hydra
- John the ripper
- Brutus
- Medusa

Testing Buffer Overflow

- OllyDbg
- Spike
- Brute force binary tester

Fuzzer

- OWASP WSFuzzer
- WFuzz

4.3 OWASP ZAP (Zed Attack Proxy)

Zed Attack Proxy (ZAP) is a free, open-source penetration testing tool being maintained under the umbrella of the Open Web Application Security Project (OWASP). ZAP is designed specifically for testing web applications and is both flexible and extensible.

At its core, ZAP is what is known as an “intercepting proxy.” It stands between the tester’s browser and the web application so that it can intercept and inspect messages sent between browser and web application, modify the contents if needed, and then forward those packets on to the destination. In essence, ZAP can be used as a “man in the middle,” but also can be used as a stand-alone application, and as a daemon process.

Figure 7: OWASP ZAP (Zed Attack Proxy).

ZAP Principles

1. Completely free and open source.
2. Cross platform (can run on windows/ Linux/Mac)
3. Very much internationalized.
4. Works well with other tools
5. Reuse well regarded components

Main Features

All the essentials for web application testing

- **Intercepting Proxy**
- **Active and Passive Scanners:**

Passive scanners- They just examines the requests and can't perform any attacks.

Active scanners- They perform wide range of attacks and should only be used on applications that you have permissions to test.

- **Spider:** Spider can be used to crawl the application. For example to find pages that you've either missed or which have been hidden from you.

- **Report Generation:** Generate reports on the issues it has found, including advice and links to more information about problem and solution.

- **Brute force (Using Owasp- DirBuster Code):** We can find file even if there is no link to them, using the brute force component.

- **Fuzzing:** We can use Fuzzing to find more subtle vulnerabilities that the automated scanner cannot find.

A Simple Penetration Test using ZAP

1. Configure your browser to proxy using ZAP

a. Open your preferred browser and set up the proxy as shown

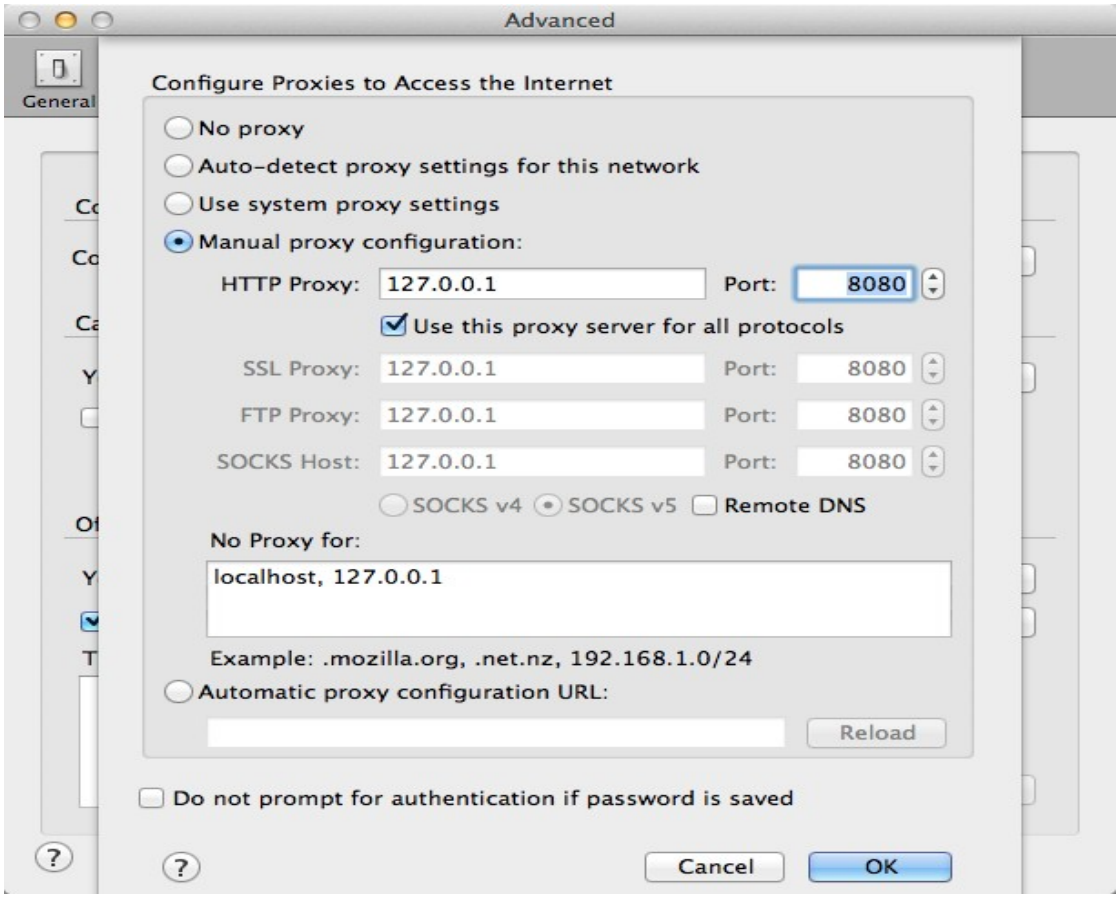

Figure 8: Configure browser proxy using ZAP.

b. In the ZAP User interface, go to Tools>Options>Local Proxy

c. Make sure the port is set to 8080(or the port you have configured in your browser)

d. Open any website using SSL in your browser and make sure the site shows up in the sites list.

2. Explore application manually

Spiders are a great way to explore your basic site, but they should be combined with manual exploration to be more effective. Spiders, for example, will only enter basic default data into forms in your web application but a user can enter more relevant information which can, in turn, expose more of the web application to ZAP. This is especially true with things like registration forms where a valid email address is required. The spider may enter a random string, which will cause an error. A user will be able to react to that error and supply a correctly formatted string, which will

cause an error. A user will be able to react to that error and supply a correctly formatted string, which may cause more of the application to be exposed when the form is submitted and accepted.

3. Use Spider to find hidden content

One way to expand and improve your testing is to change the spider ZAP is using to explore your web application. Quick Scan uses the traditional ZAP spider, which discovers links by examining the HTML in responses from the web application. This spider is fast, but it is not always effective when exploring an AJAX web application that generates links using JavaScript.

For AJAX applications, ZAP's AJAX spider is likely to be more effective. This spider explores the web application by invoking browsers which then follow the links that have been generated. The AJAX spider is slower than the traditional spider and requires additional configuration for use in a "headless" environment.
A simple way to switch back and forth between spiders is to enable a tab for each spider in the Information Window and use that tab to launch scans.

1. In the Information Window, click the green plus sign (+).

2. Click Spider to create a Spider tab.

3. Repeat step 1, then click AJAX Spider to create an AJAX Spider tab.

4. Click the push-pin symbol on both the Spider and AJAX Spider tabs to pin them to the Information Window.
Note that both of these tabs include a New Scan button.

4. See what issues passive scanner has found.

Since you have configured your browser to use ZAP as its proxy, you should explore all of your web application with that browser. As you do this, ZAP passively scans all the requests and responses made during your exploration for vulnerabilities, continues to build the site tree, and records alerts for potential vulnerabilities found during the exploration.

It is important to have ZAP explore each page of your web application, whether linked to another page or not, for vulnerabilities. Obscurity is not security, and hidden pages sometimes go live without warning or notice. So be as thorough as you can when exploring your site.

5. Use Active scanner to find vulnerabilities

So far ZAP has only carried out passive scans of your web application. Passive scanning does not change responses in any way and is considered safe.

Active scanning, however, attempts to find other vulnerabilities by using known attacks against the selected targets. Active scanning is a real attack on those targets and can put the targets at risk, so do not use active scanning against targets you do not have permission to test.

ZAP has several ways to perform an Active Scan. The most straightforward of these is to use the *Quick Start* welcome screen that is displayed by default when ZAP is launched.

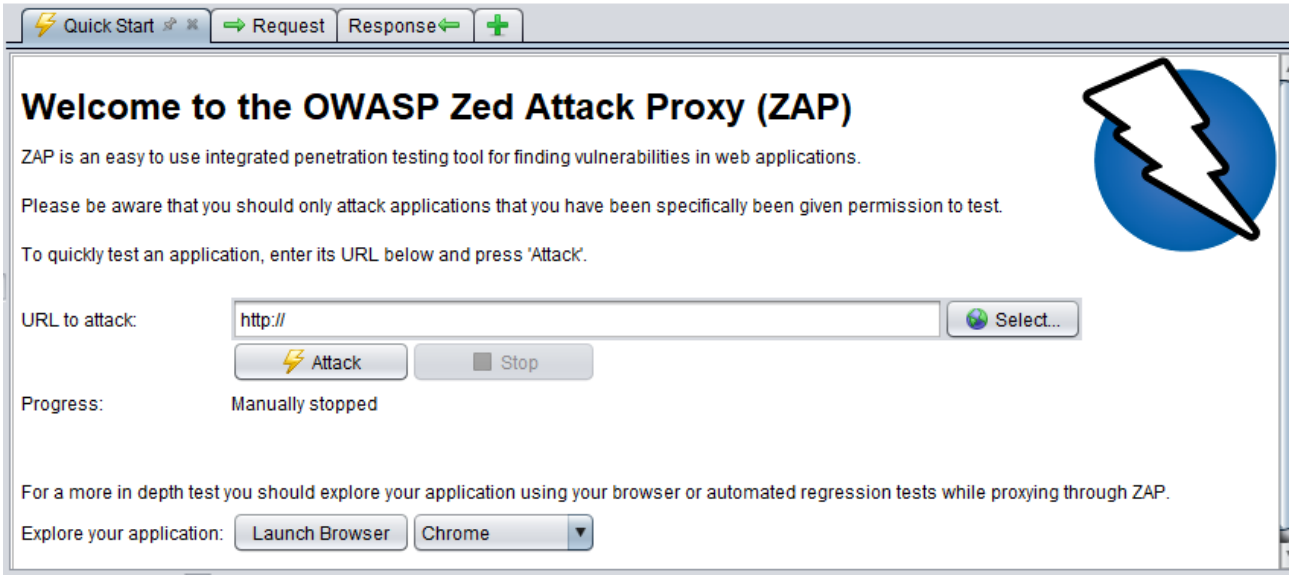

Figure 9: OWASP ZAP interface.

To begin, enter the URL you want to scan in the *URL to attack* field, and then press the *Attack* button. This will launch a two step process:

- Firstly, a spider will be used to crawl the website: ZAP will use the supplied URL as a starting point to explore the website to determine all of the hyperlinks within it (links that direct outside the domain will be ignored). The *Spider* tab at the bottom of the ZAP window will display the links as they are found. While this is happening, ZAP will simultaneously passively scan the links.

- Secondly, the Active Scan will launch: once the crawl is complete the active scan will start. ZAP will launch a variety of attack scenarios at the URLs listed in the *Spider* tab. The attack progress will be displayed in the *Active Scan* tab.

Once the active scan has finished, the results will be displayed in the *Alerts* tab. This will contain all of the security issues found during both the Spider and Active scan.

They will be flagged according to their risk - red for High Priority, and green and yellow for Medium to Low Priority, respectively.

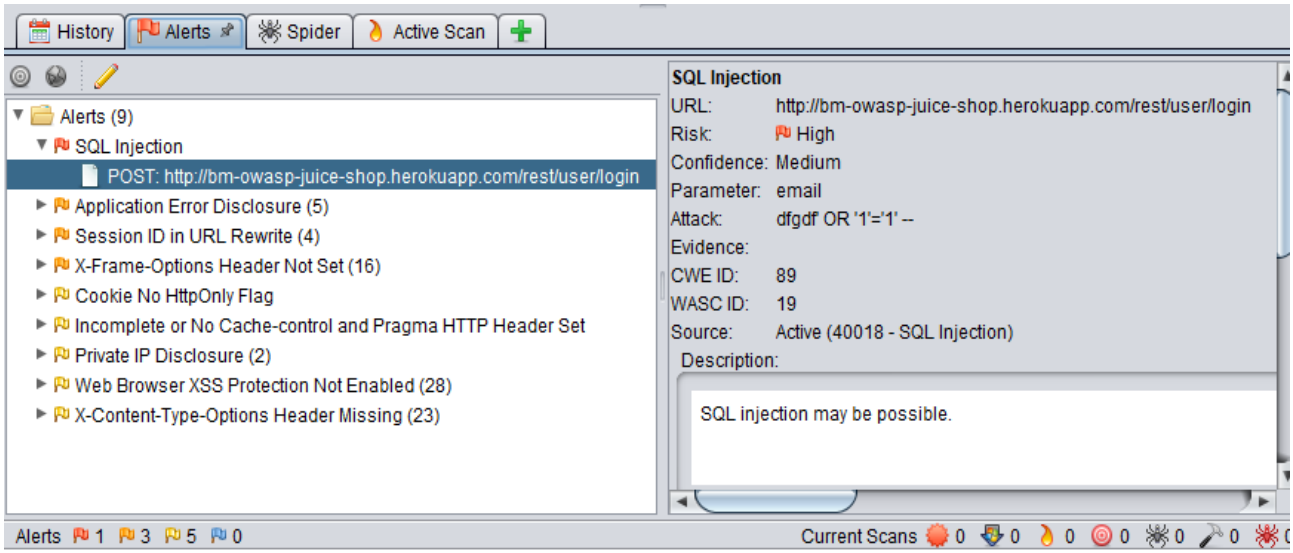

Figure 10: Result of testing with OWASP ZAP.

Depending on the size of your site, both the Spider and Active scans can take some time to perform. Fortunately, ZAP can save a session along with the results. This can be done from the *File | Persist Session* menu option.

If you need to share the results, ZAP can generate reports in multiple formats. To generate an HTML report use the menu option *Reports | Generate HTML Report*.

The above method is a simple way to obtain a vulnerability assessment of your site.

Limitations of ZAP

ZAP does have some limitations, particularly that it does not handle authentication by default.

This means that any links requiring authentication will not be found or scanned. To solve this, you can configure the **Session Properties** to include the login details that will allow ZAP access to the secure areas of your site.

4.4 Summary

In this section, first software and hardware requirements are enlisted. Then, a brief list of existing open source tools for testing of various case scenarios. Thereafter a detailed introduction of OWASP ZAP (Zed Attack Proxy) is discussed.

▪▪▪

Section 5
Software Engineering techniques

5.1 Project Management overview

- Embedded Project Planning.
- Project Scheduling (Internal in team).
- Risk management (Overview and Intuitive).
- Project monitoring and Controlling.
- W_5HH Principles (See Introduction section).
- Basic Project estimation
  - Cost estimation
  - Duration estimation
  - Effort estimation

5.2 Project Development life-cycle

The project is currently under requirements, design, and architecture phase.

Figure 11: Project development life cycle (Waterfall model)

▪▪▪

References

1] *The Browser Hacker Handbook*. Wade Alcorn, Christian Frichot, Michele Orru. Published by wiley. ISBN: 978-1-118-66209-0, ISBN: 978-1-118-66210-6 (ebk), ISBN: 978-1-118-91435-9 (ebk)

2] *XSS Attacks: Cross Site Scripting exploits and defenses*. Jeremiah Grossman, Robert Hansen, Petko D. Petkov, Anton Rager, Seth Fogie. Published by syngress.com ISBN-10: 1-59749-154-3, ISBN-13: 978-1-59749-154-9.

3] *Injection attack article*. https://www.accunetix.com/articles/.

4] *Cure53 Browser Security White Paper*. Dr. Ing. MarioHeiderich, Alex Infuhr, Fabian Fabler, Masato Kinugawa, Tsang-Chi, Dario Weiber, Paula Pustulka.

5] *Browser Security white paper*. Markus Vervier, Michele Orru, Berend Jan Wever, Eric Sesterhenn. Published by X41.

▪▪▪▪▪