UNIVERSITY OF NAIROBI

COLLEGE OF BIOLOGICAL AND PHYSICAL SCIENCES

SCHOOL OF COMPUTING & INFORMATICS

SECURITY OF WEB APPLICATIONS FROM CROSS SITE SCRIPTING ATTACKS ON BROWSER SIDE

ROTAH RICKY OMONDI

P53/79064/2015

SUPERVISOR

Dr. ROBERT OBOKO

NOVEMBER 2017

Submitted in partial fulfillment of the requirements of the Master of Science in Distributed Computing Technology

Declaration

Student

I hereby declare that this project is my own original work and has, to the best of my knowledge, not been submitted to any other institution of higher learning.

Signature: .... Date: ....

Rotah Ricky Omondi

P53/79064/2015

Supervisor

This project has been submitted as a partial fulfillment of requirements for the degree of Master of Science in Distributed Computing Technology with my approval as the University supervisor.

Signature: .... Date: ....

Dr. Robert Oboko

School of Computing and Informatics

University of Nairobi

Acknowledgement

I first thank the Almighty God for these far have reached in this project; I would like to appreciate my supervisor, the lecturers and fellow students for their continual support they accorded me in this journey.

I would have not reached this far if I didn't have the support of my Wonderful wife Nelly and my lovely daughter Brianna and son Bravyn. I thank you my family

My great parents Mr and Mrs Joseph A. Rota I thank you for support and great sacrifices to educate me.

My colleagues, thank you for always according me assistance in times when I needed them.

God bless you

Abstract

Cross-Site Scripting has been known to be the most common and serious attack against web based services. Confidentiality of browsers has been compromised in many ways since they support the execution of embedded scripts. With this capability of sites to be attacked, the attackers have the capacity to take control of the sites through cross site scripting attacks.. How to combat and prevent the attacks has to be looked at seriously. With this in mind I have come up with a tool to detect and deter these attacks. I elaborate how well overly we can solve the problem with a tool that deals with the cross-site scripting attacks. The network has developed from static sites to dynamic sites and the social media from Facebook, twitter, whatsapp, e.t.c are the leading online interaction sites. The participants in these networks use these malicious injected script codes without knowing. The current browsers are so limited in detecting the attacks and hence tools need to be developed to deal with this problem on the browser side. The tool is developed using python and java to deal with the attacks. In this work have come up with a tool that is secure, light in weight and fast in detecting malicious codes that appear within the script. The project is specific to look at the gap, build and test the components of the tool and evaluate the tool's performance, reliability and accuracy as compared with other tools. These project tries to answer most of the asked questions such as, is it possible to stop XSS attacks in a more secure and easy way, what do the current tools offer, what are the shortfalls of these tools and previous researches and can we develop a secure and light tool that can be incorporated within the web service scripts to reduce overhead load associated with current tools? The project covers the area of XSS related attacks more so on the client side of the browsers, malicious attacks and input area. In this project the research design method used is Design science. Design science is an outcome based information technology research methodology, which offers specific guidelines for evaluation and iteration within research projects. With the project intent in having a functional algorithm and an artifact, this design method best suits this project. The XSS tool will be based on other tools that have crawling, attack and analysis component: The crawling component looks for pages within the web application. It acts as the scanner and if its poor then it will miss major vulnerability. This component is the most crucial part since it must get all the pages within the web application. The attack component scans and extracts all linkstaht are within the web application and then all the page forms which have the URL parameters and injects some patterns of attack. The patterns have parameters

that can be either part of the HTTP POST request or URL query string. The two are not hard to exploit and can be so easily attacked. The analysis component determines the servers response and uses attack-specific keywords and pattern to determine how successful this was. An attack vector consists of a JavaScript code that is encoded into a algorithm and is reflected on the embedded HTTP response.. The sampling process involves the acquisition of known and reported attacks from the websites that have archives of these attacks. The sampling involved using of random data reported year by year. This is because the attacks evolve over period and hence this random collection achieved the best procedure. The results indicated over 85% accuracy in detection. This was quite impressive since the tool is more client side oriented and these attacks originate from different areas with diverse attack patterns. In conclusion the project has a strong ground for developing a tool that is based on the client side and can actually detect the attacks on the client system. This relieves the overload from the servers and allows the developers time to focus on the development of the systems and leave the security to the clients. The recommendation on the future development tools that affectively detect and deter XSS attacks should consider the overload associated with having the tools on the server side as opposed to client side. The XSS tool developed in this work is over 85% effective, from the test done, and future researchers can and should use this as a base to implement more effective tools.

Table of Contents

List of Figures

List of Table

Definitions

XSS (Cross site scripting)- is a type of computer security vulnerability which enables attackers to inject client-side scripts into web pages viewed by other users.

Web applications- is a client–server computer program in which the client runs in a web browser

SQL (Structured Query language)- is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system

Design Science- science is an outcome based information technology research methodology, which offers specific guidelines for evaluation and iteration within research projects.

Abbreviations

XSS	Cross-Site Scripting
WWW	World Wide Web
HTTP	Hyper Text Transfer Protocol
PHP	Hypertext Preprocessor
ASP	Application Service Provider
JSP	Java Servlet Pages
HTML	Hypertext Markup Language
CGI-PERL	Common Gateway Interface WITH Practical Extraction and Report Language
SQL	Structured Query Language
DNS	Domain Name Service
SOME	Same Origin Method Execution

CHAPTER ONE: INTRODUCTION

Cross-Site Scripting has been known to be the most common and serious attack against web based services. Having vulnerability detection tools that runs on the browser side is the easiest way to detect this. Available tools for this XSS attacks detection are quite poor. In this chapter I discuss the introduction, the problem, and about scope and limitation of this work.

1.1 Introduction

The internet comprising of the transport layer to the Application layer i.e the web has become so big and widespread that millions of applications and web services have been developed. We have had developers of these applications come into play with some who have no knowledge in security of the applications. This has lead to the developers depending on the security offered by the browsers. Growth on the internet and more so on the World wide web is so tremendous that if no solution is put across then attacks will be everywhere. Due to rapid growth in internet, the web sites have become more professional and dynamic. At current day and age dynamism in websites has taken over the internet. Social networks, transport services, smart cities and homes, banking among others use web services for their day to day operations.

Hamada et al (2012) says “clients usually go to Google to search information, Amazon or E-Bay to buy books and many other goods and also they go to facebook to communicate with friends. Therefore, there is no doubt that Internet is gradually becoming an integral part of our daily life.” Since these sites have traffic ranging in millions of hits per hour the security against XSS is more required and well implemented on the browser itself as opposed to the web service provider. Hamada et al (2012) further states “that they should offer more security to the users of the above sites in a more reliable way. If the sites become vulnerable then due to the traffic to these site attacks are imminent and reputation for such sites are dented. So having a secure and beneficial network environment is necessary.”

As technology becomes more and more complicated and connected, attacks too are evolving and the web service providers will require other developers to handle the attack security part. This will relieve the web service developers to deal with their core job of developing without thinking of the security component hence high productivity and reduced cost. Injection of cross site scripts has been done by the hackers and has resulted in losses from data to critical information. Damaging reputation of a site once an attack occurs is quite huge and at times can be irreversible. If you take an example of Facebook where traffic is so high with comments and counter comments if one injection was to occur, the possibility of the same being spread is so high. When such attacks occur it’s the site that loses its business but not the browser company.

“One reason for the widespread of XSS vulnerabilities is that many developers aren't trained well enough about the security of websites. Security is often considered as a burden and as an extra effort that costs time and money, which can only be added at the end of a software project, if time and money still allow it. Regular security tests need to be part of an effective software development process and automated tools such as Web Vulnerability Scanners play an important role in providing a testing framework. Unfortunately, these tools aren't capable of detecting all kinds of XSS vulnerabilities, mainly because their attack strategy is ineffective” (Hydara et al 2015.)

1.2 Problem Statement

The internet comprising of the transport layer to the Application layer i.e the web has become so big and widespread that millions of applications and web services have been developed. We have had developers of these applications come into play with some who have no knowledge in security of the applications. This has lead to the developers depending on the security offered by the browsers. Growth on the internet and more so on the World wide web is so tremendous that if no solution is put across then attacks will be everywhere. As the Internet is growing, the web sites become more professional and dynamic. Running the malicious codes is mostly on the client with the participant not having any knowledge of the operation. It takes over and compromises the customer information without his/her knowledge. The problem associated with this compromise is so detrimental that can be irreversible. These attacks have been so common on the client side and at times are even initiated by the participant themselves. As a result, the problem should be considered at the client side in default. The performance and reliability of the current tools have come into question. They do not cover the whole scope of XSS attacks and hence need to develop more light, fast and accurate tool. In this work a tool capable of dealing with this problem is developed.

1.3 Objectives

1.3.1 Main objective:

The objective is to develop a reliable high performance and secure tool that can detect malicious XSS code at browser side.

1.3.2 Specific objective

a) To analyze, classify and discuss the current situation of XSS problem.
b) To identify, build and test the components of the proposed tool.
c) To determine the capability, accuracy and performance.

1.4 Research Question

This project tries to answer some frequently thought about questions, but not resolved effectively, about Cross site Scripting (XSS) attacks. These questions are:-

a) Is it possible to stop XSS attacks in a more secure and easy way?
b) What do the current tools offer?
c) What are the shortfalls of these tools and previous researches?
d) Can we develop a secure and light tool that can be incorporated within the web service scripts to reduce overhead load associated with current tools?

1.4 Scope and limitations

1.4.1 Scope of the project

a) The project analyzes the current situation on scripting attacks.

b) The work covers the current problems associated with XSS attack at the browser side.

c) The solution is geared towards ease of use of tools to deter XSS attacks.

1.4.2 Limitation

a) The model is only incorporated on the web service through a call within the service.

b) The model is only applicable around the input from the participant.

CHAPTER TWO: LITERATURE REVIEW

2.1 Introduction

This section is broken down into background of XSS, risks associated with XSS, related tools, protection of the perimeter wall, exploitation of XSS vulnerability, the need to avoid XSS attacks, summary of the review and Conceptual design.

2.2 Background

Security of web applications is a major concern in many organizations. Organizations come up with widespread ways to protect themselves. They try to protect their sensitive data and systems through many tools like firewall, Intrusion Detection System (IDS) e.t.c from attacks on their web based services. Most don’t achieve they intend even after putting up such tools. Some go to the extent of deploying humans just to monitor the traffic within their organizations. But still we have cases of attacks even on the most presumed secure organization. The main attacks are on SQL queries through SQL injection by XSS designed by hackers. This literature review digs deep into the security threats/vulnerabilities that are exploitable or are already exploited by these XSS attacks. Gadhiya et al 2014, states that,“as new technology arrives, it comes with lot of new features and possibly attacks. In the today’s trend of social web application, the SQL injection, cross site scripting (XSS) attack and cross site forgery attack are major challenges for web application.”

Federal Information Processing Standards Publication (FIBS PU 200) defines a threat as a possible danger that might exploit a vulnerability to breach security and thus cause harm. Threats are basically exploits that allows one to obtain valuable thing at absolutely no cost with the intent of destroying reputation.

The most important security notion within web communication is the idea of the same-origin policy. Same origin policy plays a lot in the systems vulnerability by assuming that what follows in a secure connection is always trusted. Even though the idea of same origin policy is much welcomed, it has brought with it the problem of hackers taking advantage. The principal intent for this mechanism is to make it possible for largely unrestrained scripting and other interactions between pages served as a part of the same site (understood as having a particular DNS host name, or part thereof), whilst almost completely preventing any interference between unrelated sites.

Same origin policy in reality and practically does not exist, but a set of algorithms with artificial identity exist.

K Selvamanii et al (March 2010). “International Journal of Computer Science and Information Security Vol 7, No 3, describes the rapid growth of Internet as resulting in feature rich, dynamic web applications hence result in harmful impact of security flaws in applications.” The most popular form of security threats in the current development of web applications is XSS. This is due to the ease of having a malicious script run alongside the real script without being detected. When you have a trusted site and you access it the XSS flaw will occur across the intermediate trusted site and the un-trusted site. Client side solution acts as a web proxy to mitigate Cross Site Scripting Flaws which manually generate rules to mitigate Cross Site Scripting attempts. It protects against having information’s leak from the authorized user to the hacker. Cross Site Scripting Flaws are easy to execute, but difficult to detect and prevent. The current solutions tent to reduce the level of performance of client’s system resulting in a poor web surfing experience. This project will provide a client side solution that is much light and easy to use.

Hayak (2015) states that “To satisfy our need for a faster and more efficient user experience while using web applications, new protocols and techniques are formulated. They supply the web application developers with a powerful variety of options.” Developers are given variety of options in the new ways of dealing with security of web applications. As more applications are becoming more powerful, greater responsibility is required. It implies that the more growth in diversification of web application, more power is acquired and with wrong implementation the a disastrous outcome is expected. "Same Origin Method Execution" (SOME) is a result of such wrong implementation. SOME is a new web application attack where a site assumes that once a connection is established the connection is assumed to be from the same origin and that is kept until the communication is dropped. Importantly, the Same origin method execution entails a risk level threat only surpassed by Cross site Scripting.

As a result almost every web service suffers from continues injection flaws. Xssed.com has tracked the reported issues which are of these kind. They did gather over 80,000 entries in slightly over three years and some of the persistent attacks were rather devastating. According to Shalini et al 2011, “the solutions on server side result in considerable degradation of web application and are often unreliable, whereas the client side solutions result in a poor web browsing experience, there is need of an efficient client side solution which does not degrade the performance.”

According to Hydara et al (2015), “Research on XSS is still very active with publications across many conference proceedings and journals. The research has been further put high up on these conferences with the target being the scripting area. Other areas that investments have been focused on are attack prevention and vulnerability detection. Majority of solutions proposed and looked into are the dynamic analysis. Reflected XSS was dealt with the most Even thoughit still remains a big problem XSS attacks are still not under control even after many solutions being developed. More research is needed in the area of vulnerability removal from the source code of the applications before deployment.”

Upto date XSS has remained the most common vulnerability despite it being discovered back in 2000 by Computer Emergency Response advisory team. XSS has grown to be the most common attack in web applications and ranked first in OWASP Top Ten report 2007, second in 2010 and third in 2013. It occurs when one receives a malicious code then sent to other persons with or without knowledge. Systems that have comments and are able to give users an input field are most susceptible to attacks of this nature. Social media sites, chats, Forums, , are all good examples of applications most susceptible to XSS.

Hamada (2012) states that “XSS is a common vulnerability which can let hackers inject the code into the output application of web page which will be sent to a visitor’s web browser and then, the code which was injected will execute automatically or steal the sensitive information from the visitors input. This code injection which is similar to SQL Injection in Web Application Security, Figure 1.1 below shows how atypical XSS attack can be executed.”

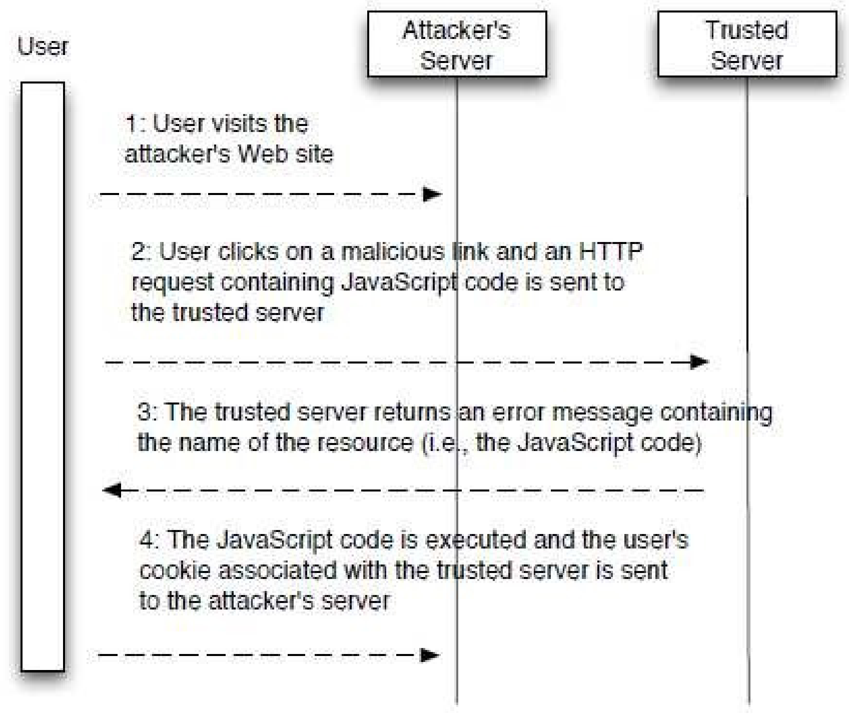
User
Attacker's Server
Trusted Server
1: User visits the attacker's Web site
2: User clicks on a malicious link and an HTTP request containing JavaScript code is sent to the trusted server
3: The trusted server returns an error message containing the name of the resource (i.e., the JavaScript code)
4: The JavaScript code is executed and the user's cookie associated with the trusted server is sent to the attacker's server

Figure 1 Scenario in XSS

2.3 Cross site scripting examples

Most sites are based on scripts and this makes them easy to attack. With millions of web services developed everyday attackers are also busy checking vulnerability associated with such application development. Generally cross site scripting vulnerability has been so profitable and attackers are so informed on how to exploit this. Going on with getting the vulnerability the hackers can get control of the whole site.

The risks associated with XSS attacks if not dealt with are so wide and leaves your website vulnerable to:

a) Stealing of login details like username and passwords.
b) Taking sensitive information from clients and their website behavior.
c) Stealing of company data.
d) Keystroke logging of clients that visit your site
e) Bringing down your whole site.
f) Causing denial of service
g) When they take control of your site they can ask for ransom.
h) Redirecting your visitors to wrong sites.

When XSS attackers exploit the vulnerability and attack a site they use those websites to:

a) Probe the rest of the intranet for other vulnerabilities
b) Spread the XSS attack cross the intranet
c) Monitor the intranet behavior
d) Launch Denial of Service attacks
e) Launch Data illegal acquisition

2.4 Related tools

Many tools have been developed to deal with vulnerability. A study by Shodhgana et al (2013), states that "many authors analyzed the limitations of the vulnerability scanners, by the study of penetration testing and static code analysis. An important fact when considering the limitations of the vulnerability scanning tools is that the testing for security is difficult. The testing involves complicated ways that requires some technical knowledge. The study further state that, finding vulnerability of sites is easy but determining how secure an application is could be challenging. Both penetration testing and static code analysis tools have intrinsic limitations. Penetration testing relies on effective code execution; however, in practice, vulnerability identification only examines the Web application's output Penetration testing is not done on the source code but just on how a site can be penetrated whereas static analysis involve having access to the source code. Failure to use the right tool to detect the above will leave the applications with undetected vulnerabilities."

Vulnerabilities in the application layer protocol are the main access point for hackers and this makes it very hard to detect. Several tools and techniques have been developed to analyze the vulnerabilities in web-based applications. Having this vulnerability scanned on daily basis is one of the ways to make sure your website is protected. Once the vulnerability is detected a proactive measure should be taken to ensure that the exploits are deterred. Having a reactive rather than proactive measure would allow for quick identification of vulnerabilities, which leave many websites in good health. Zero day attacks are quite common and deterring them is of high interest. There is no patch available for this vulnerability. Even though a Zero-Day attack may occur, it is possible to identify where the compromised websites are extracting the attack information from, or where the malicious website visitors are being redirected to.

2.5 Protection of the perimeter wall

Perimeter wall is the area around the local network that allows for external access to the internet. Hyper Text Transfer Protocol (HTTP) is a stateless protocol widely used in the internet world wide web. The stateless protocol simplifies the server conception, because there is no need to dynamically allocate storage to deal with every current state. If a client disconnects in midtransaction, no part of the system needs to be responsible for cleaning the present state of the server. However, this forces web developers to use alternative methods to authenticate HTTP requests and to maintain users' states.

2.6 Exploitation of a cross-site scripting vulnerability

According to Patil and Agrawal 2015, "wherever web based applications are distributed applications consisting of components that execute either on a web server or on a user's client, scripting vulnerabilities arise when content controlled by an adversary (i.e. Un-trusted data) flows into critical operations of the program (i.e. critical sinks) without sufficient security confirms. Patil determines that attackers take advantage of distributed applications to allow for scripting attacks to flow into critical parts of any application. This causes an attacker to gain higher privileges than intended by the web based application, characteristically contributing entrusted data the same chances as the web application's code."
Most of the attacks are first achieved by doing random checks on the vulnerability of the applications to XSS attacks. Unfortunately, most web applications, both Free/Open Source Software and commercial software, are susceptible. Attackers simply perform a Google search for terms that are often found in the software. After discovering the vulnerable web site the attacker looks through the HTML to find where the exploit code can be put. After this has been determined, the attacker then begins to code the exploit in any of these three ways:

1. **Reflected attacks:** The client/end-user is duped into clicking a malicious link or submitting a changed or manipulated form. The injected code is sent to a vulnerable web server that directs the cross-site attack back to the user's browser. The browser then executes the malicious code, assuming it comes from a trusted server.

2. **Stored (persistent) attacks:** This is where a malicious code is injected into the target server and sits there waiting for trigger. When interacting with the target server, an end-user inadvertently retrieves and executes the malicious code from the server.

3. **DOM-based attacks:** Remote execution is activated to allow the attacker to execute malicious code on the target computer. The attacker injects the written code into the target site.

Getting the pay

After the attacker gains access to the vulnerable site, he/she now starts to be rewarded with the pay for the attack. In cases where the hacker's intent was to steal login credentials then they are now collected. Actually the attacker is now in control and will do whatever they wish to do. Google and other site engines usually flag vulnerable sites after such attacks even if the sites have been fixed. This is quite expensive to the organization, considering the reputation cost it has on the organization.

If the attacker succeeds , he/she will often do it on other related websites to increase the reward.

2.7 Why cross-site scripting attacks should be avoided

Cross-site scripting costs organization by destroying their reputation and also on lose of sensitive data. The trust that takes ages to be built is destroyed by just one attack. The visitors will be discouraged to return to such sites even if it has been cleared and given good bill of health. Most of the search engines flag such sites even if the owners have fixed everything. Recovering from such kind of embarrassment is almost impossible and time will be required for the trust to be restoredThe level of threats that attacks like XSS, SQL injections, denial of service among others have against the sites, requires the web site owners and developers to be on alert to deal with any of these.

2.8 Review summary

With the current tools you can detect and deter some XSS attacks since they inspect the traffic and checks if the site has XSS vulnerabilities. But still these tools have some issues that need to be solved:

a) Additional run time overhead and lack of identification of vulnerabilities due to false positives.

b) Pricing costs which are not economical to small organizations or individuals. Even though dotDefender is placed as the cheapest tool it goes for around $3,995
c) Most tools require almost the entire configuration to be done manually and changed each time the site is altered.
d) There are no tools for improving the performance and speed of your website.

To address the above said limitations, I propose to come up with a system that will try to counter these limitations by:

a) Having the code run within the services and on the browser side hence reduce the overhead of false positives by only checking the injection within its own service
b) A cheap tool that is incorporated within the code of the web service
c) A tool that is customizable by allowing users to configure each service on its own

2.9 Conceptual Design

Xssed.com tries to put facts about XSS attacks. They explain that these attacks are generally invisible to the victims. These leads to many attacks going undetected and even the client might be a victim without realizing. Web/application servers and web application platforms are easily attacked by XSS. Without real protection they are susceptible to XSS. Websites that prefer to use Secure Socket Layer are not left out of these attacks. The only difference between protected sites, i.e the SSL secure and the unencrypted sites is the encryption of the connection. Every month over 25 XSS holes are published and explained by the Xssed.com

To try and counter thesethe project brings use of three components: crawling, attack and analysis component.

The crawling component looks for pages within the web application. It acts as the scanner and if its poor then it will miss major vulnerability. This component is the most crucial part since it must get all the pages within the web application.The attack component scans and extracts all links that are within the web application and then all the page forms which have the URL parameters and injects some patterns of attack. The patterns have parameters that can be either part of the HTTP POST request or URL query string. The two are not hard to exploit and can be so easily attacked. The analysis component determines the servers response and uses attack-specific keywords and pattern to determine how successful this was.

Basically the concept is to detect, analyze and deter XSS attacks.

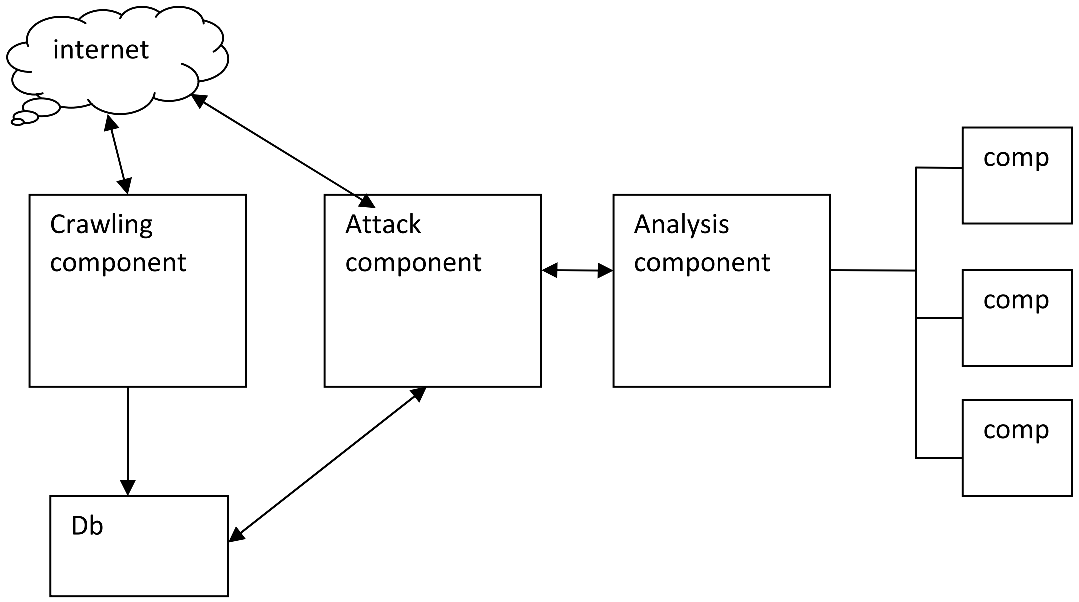

Figure 2 Conceptual design

CHAPTER THREE: METHODOLOGY

3.1 Overview

This chapter covers research strategy, guidelines, sample and sampling procedure, validity and reliability of mechanisms, data collection procedures and data analysis procedures.

As a penetration technique XSS entails code injection; where user input is mistakenly interpreted as malicious program code. In order to prevent this type of code injection, secure input handling is needed, hence the need to come up with strategies on how to handle the above problem.

Most systems attempt to prevent XSS attacks against web applications on the web server or attempt to remove vulnerabilities from the web application directly. While it is good to protect users from an attack when interacting with a specific web application, the users are unprotected when visiting other web sites. In this chapter I will describe the proposed model which works as current web vulnerability scanner, then describes technical details of XSS.

3.2 Research Design

The study involved a careful and complete observation of the unit, which is a web page technology and emphasized on depth rather than the breath of study of XSS.
Design science research is the research methodology used in this project. Design science is a research methodology that yields a final outcome based on information technology and offers guidelines for evaluation and iteration within research projects. With the project intent in having a functional algorithm and an artifact, these design method best suits this project. It focuses on the development and performance of (designed) artifacts with the explicit intention of improving the functional performance of the artifact. Design science research is typically applied to categories of artifacts including human/computer interfaces, design methodologies and languages.

3.3 Design Science Guidelines

3.3.1 Design Artifact

This work main objective is to come up with a python based tool that deters XSS attacks at the browers side. The tool is able to analyze the input by a user to determine whether the Input is received as was sent or there has been manipulation.

Below is the code that for the checking of the input in each text inserted.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class _Default : System.Web.UI.Page
{
   protected void Page_Load(object sender, EventArgs e)
   {
      string id = Request.QueryString["id"] as string;

      if (id == null)
      {
         lblId.Text = "NA";
      }
      else
      {
         lblId.Text = id;
```

```
        }
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        lblMessage.Text = "Hello " + TextBox1.Text;
    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        lblMessage2.Text = "Hello " + TextBox2.Text;
    }
    protected void Button3_Click(object sender, EventArgs e)
    {
        string rawInput = TextBox3.Text;
        string encodedinput = Server.HtmlEncode(rawInput);

        lblMessage3.Text = "Hello " + encodedinput;
    }
}
```

3.3.2 Problem Relevance

As stated in the problem statement in chapter 1, as web application becomes the new way of doing business, web security is becoming crucial to programmers. If these are not sorted then we will be left at the hands of attackers.

3.3.3 Design Evaluation

The tool is put to thorough evaluation by doing experimental tests, analytics and observation. The tool is subjected to many input text/scripts in a real web application platform and results noted against the other tools currently available. The main method that is used in this project is by experimental evaluation where the simulation is performed on a sample data received from previous attacks reported by the xssed.com site.

3.3.4 Research Rigor

The XSS tool developed here in is secure, fast and high performance based on a research conducted and developed by Shodhganga et al 2013 . Using that research, combined with research on SQL injection the XSS tool is developed.

3.4 Methodology

The project brings use of three components: crawling, attack and analysis component.

3.4.1 Crawling Component

The crawling component collects all pages of a Web application, and then stores each application with its corresponding page links within the application.

It picks on the given URL within the pages and determines all the links within each page and put then on a result list. Scanners engines are poor in detecting vulnerability but the crawling component arguably get the best since it collects all the links within a page.It basically collects all the input to the system and store them for the next component to analyze.

3.4.2 Attack Component:

The attack component extracts all internal links by going through the website, then injects various attack patterns. It parses this attack patterns into the URL parameters which were scanned by the crawling component. Parameters can be part of the URL query string or part of the request body in HTTP POST requests. Both are equally exploitable.

3.4.3 Analysis Component:

The analysis component parses and interprets the server's responses. If keywords or certain pattern is detected then an attack was successful. The analysis component uses some kind of attack vector, a piece of HTML or JavaScript code that is put into a parameter in-order to be reflected to user by being embedded into a HTTP response, to achieve its goal. A trusted website or part of the attack vector could carry the malicious code.

3.5 Sampling Procedures

The sampling process involves the acquisition of known and reported attacks from the websites that have archives of these attacks. The sampling involved using of random data reported year by year. This is because the attacks evolve over period and hence this random collection achieved the best procedure.

3.6 Data Collection and Analysis

As stated above in the sampling procedure the data collected were as follows:-

1.
```
<SCRIPT type="text/javascript">
var adr = '../evil.php?cakemonster=' + escape(document.cookie);
</SCRIPT>
```

2.
```
<%...
 Statement stmt = conn.createStatement();
 ResultSet rs = stmt.executeQuery("select * from emp where id="+eid);
 if (rs != null) {
  rs.next();
  String name = rs.getString("name");
%>

Employee Name: <%= name %>
```

3.
```
<% String eid = request.getParameter("eid"); %>
...
Employee ID: <%= eid %>
```

4.
```
<META HTTP-EQUIV="refresh"
CONTENT="0;url=data:text/html;base64,PHNjcmlwdD5hbGVydCgndGVzdDMnKTwv
c2NyaXB0Pg">
```

5. <img src="http://url.to.file.which/not.exist" onerror=alert(document.cookie);>

6. <b onmouseover=alert('Wufff!')>click me!</b>

7. *<SCRIPT>var+img=new+Image();img.src="http://hacker/"%20+%20document.cookie;</SCRIPT>*

CHAPTER FOUR: RESULTS, RECOMMENDATIONS AND CONCLUSION

4.1 Introduction

This chapter entails the results, recommendations and conclusion based on the whole project life.

4.2 Results

The first objective of this research was to analyze, classify and discuss situation of XSS problems. I was able, as shown in chapters 1 and 2, that XSS is a reality and has to be dealt with through development of effective tools. The project has been able to analyze the problems that come along with XSS attacks and these problems are grouped as:

a) Stealing of login details like username and passwords.
b) Taking sensitive information from clients and their website behavior.
c) Stealing of company data.
d) Keystroke logging of clients that visit your site
e) Bringing down your whole site.
f) Causing denial of service
g) When they take control of your site they can ask for ransom.
h) Redirecting your visitors to wrong sites.

These attacks are truly risky in the current day of e-commerce where almost all transactions are internet based.

The second objective was to identify, build and test the components of the XSS Detection tool. The tool has been identified, built and tested within the time plan. The tool is called XSStool and is run on the client side. A working tool that implements a python to deter XSS attacks at the client side was developed. The tool is able to analyze the input by a user to determine whether the Input is received as was sent or there has been manipulation. The XSS detection tool has three major components:

The crawling component collects all pages of a Web application, and then stores each application with its corresponding page links within the application. It picks on the given URL within the pages and determine all the links within each page and put then on a result list. Scanners engines are poor in detecting vulnerability but the crawling component arguably get the best since it collects all the links within a page.

The attack component will scan website, extracts all internal links then scans all crawled pages forms field which is used in URL parameters then injects various attack patterns into these parameters. The analysis component parses and interprets the server's responses. It will use attack-specific criteria and keywords to determine if an attack was successful.

For simplicity and easy installation, data is stored in a text file rather than in a database. Finally the third objective was to determine the capability, accoracy and performance.

The analysis of the data collected in previous chapter involved passing this above known and reported attacks through the tool with the following results which were further compared to other tools:-

Table 1 Results of data tests

ATTACK	XSS TOOL	ACUNETIX	
<SCRIPT type="text/javascript"> var adr = '../evil.php?cakemonster=' + escape(document.cookie); </SCRIPT>	DETECTED	DETECTED	
<%... Statement stmt = conn.createStatement(); ResultSet rs = stmt.executeQuery("select * from emp where id="+eid); if (rs != null) { rs.next(); String name = rs.getString("name"); %> Employee Name: <%= name %>	DETECTED	DETECTED	
<META HTTP-EQUIV="refresh" CONTENT="0;url= data:text/html;base64,PHNjcmlwdD5hbG VydCgndGVzdD MnKTwvc2NyaXB0Pg"> <% String eid = request.getParameter("eid"); %> ... Employee ID: <%= eid %>	NOT DETECTED	DETECTED	

<img src="http://url.to.file.which/not.exist" onerror=alert(document.cookie);>	DETECTED	NOT DETECTED	
<b onmouseover=alert('Wufff!')>click me!</b>	DETECTED	NOT DETECTED	
<SCRIPT>var+img=new+Image();img.src= "http://hacker/"%20+%20 document.cookie;</SCRIPT>	DETECTED	DETECTED	

From the above it was noted that the tool had an accuracy of over 85%. This tool has the ability to deter this kind of attacks.

4.2 Limitations

1. The study had challenges in that the attacks keep changing due to the dynamic field of scripting. As per the time of doing the test most of the known attacks were being handled by the server side tools.
2. Time constraint. Due to the wide scope of XSS attacks the time required to finalize the project was short.

4.3 Conclusion

Looking at the size of the tool as compared to others is quiet simple and easy to set up. The tool basically runs on the client side and hence is so easy to handle. The results indicated over 85% accuracy in detection. This was quite impressive since the tool is more client side oriented and these attacks originate from different areas with diverse attack patterns. In conclusion the project has a strong ground for developing a tool that is based on the client side and can actually detect the attacks on the client system. This relieves the overload from the servers and allows the developers time to focus on the development of the systems and leave the security to the clients.

4.3 Recommendation

The recommendation on the future development tools that affectively detect and deter XSS attacks should consider the overload associated with having the tools on the server side as opposed to client side. The XSS tool developed in this work is over 85% effective, from the test done, and future researchers can and should use this as a base to implement more effective tools.

Reference

1. Berinato, Scott 2007, *Software Vulnerability Disclosure: The Chilling Effect. CSO (CXO Media).* p. 7. Viewed February 21, 2016 http://www.webappsecblog.com/CsrfAndSameOriginXss.html

2. Hevner, A. R.; March, S. T.; Park, J. & Ram, S 2004. *Design Science in Information Systems Research,* MIS Quarterly, 2004, 28, 75-106. viewed July 12, 2016 http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.103.1725&rep=rep1&type=pdf

3. Isatou, Hydara, Novia, Admodisastro, Hazura Zulzalil, Abu, Bakar, Md, Sultan 2015, *Current state of research on Cross-site scripting,* viewed 20 July 2016 http://www.sciencedirect.com/science/article/pii/S0950584914001700

4. Shodhganga 2015, *Prevention of code-injection attacks,* viewed February 25, 2016

   http:// shodhganga.inflibnet.ac.in:8080/jspui/bitstream/10603/24523/

5. Vaishnavi, V. and Kuechler, W. (2004/14), *Design Research in Information Systems*, viewed 12 July 2016 http://desrist.org/design-research-in-information-systems

6. WASC Threat classification Version 2.0 viewed March 7 2016 http://projects.webappsec.org/f/WASC-TC-v2_0.pdf

7. 'Web security attacks and Injection-A Survey International Journal of Advancements in Research & Technology' February -2015, vol. 4, Issue 2.