THREAT MODELLING AND ANALYSIS OF WEB APPLICATION ATTACKS

by

Tolulope Bukola Awojana

November, 2018

Director of Thesis: Dr. Te-Shun Chou

Major Department: Technology Systems

Abstract

There has been a rapid growth in the use of the Internet over the years with billions of businesses using it as a means of communication. The World Wide Web has served as a major tool for disseminating information and this has resulted to the development of an architecture used in information sharing between remotely connected clients. A web application is a computer program that operates on web technologies and browsers to carry out assignments over the Internet. In designing a secured web application, it is essential to assess and model the viable threats. Threat modelling is a process used to improve application security by pointing out threats and vulnerabilities, outlining mitigation measures to prevent or eliminate the effect of threats in a system. With the constant increase in the number of attacks on web applications, it has become essential to constantly improve on the existing threat models to increase the security posture of web applications for proactiveness and strategic goals in operational and application security. In this thesis, three different threat models - STRIDE, Kill Chain and Attack Tree were simulated and analyzed for SQL injection and Cross-Site Scripting attacks using the Microsoft SDL threat modelling tool, Trike modelling tool and SeaMonster modelling tool respectively. This study will aid future research in developing a new and more efficient threat model based on existing ones. It will also help organizations determine which of the models used in this study is

best suited for the business' security framework. The objective of this study is to analyze the three commonly used models, examining the strengths and weaknesses discovered during the simulation and compare performances.

THREAT MODELLING AND ANALYSIS OF WEB APPLICATION ATTACKS

A Thesis

Presented to the Faculty of the Department of Technology Systems

East Carolina University

In Partial Fulfillment of the Requirements for the Degree

Master of Science in Network Technology

by

Tolulope Bukola Awojana

November 2018

THREAT MODELLING AND ANALYSIS OF WEB APPLICATION ATTACKS

By

Tolulope Bukola Awojana

APPROVED BY:

DIRECTOR OF THESIS:

_____

Te-Shun Chou, Ph.D.

COMMITTEE MEMBER:

_____

John Pickard, Ph.D.

COMMITTEE MEMBER:

_____

Biwu Yang, Ph.D.

CHAIR OF THE DEPARTMENT OF TECHNOLOGY SYSTEMS:

_____

Tijjani Mohammed, Ph.D.

DEAN OF THE GRADUATE SCHOOL:

_____

Paul J. Gemperline, Ph.D.

ACKNOWLEDGEMENT

I would like to first thank the Almighty God for seeing me through as this would have been impossible without Him. I also thank my advisor, Dr. Te-Shun Chou for his unending support and guidance, Dr. John Pickard and Dr. Biwu Yang for being very supportive and being a part of the committee to assess this work. I also thank Dr. Tijjani Mohammed for his support and guidance all through this program. My immense appreciation goes to my parents and Mr. Omorobe, this study would have been impossible without them providing and training me up to this phase. I would also like to appreciate Mr. and Mrs. Dare for their consistency, support, love and guidance. I also thank Nicholas Hempenius for his immense help and support through the course of this study as well. Finally, I would like to thank my love, Olasupo for his unflinching love and understanding throughout this entire program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

XSS:	Cross Site Scripting
HTML:	Hypertext Markup Language
IP:	Internet Protocol
DNS:	Domain Name System
ISP:	Internet Service Provider
TCP:	Transmission Control Protocol
PHP:	Scripting Language
HTTP:	Hypertext Transfer Protocol
SQLi:	Structured Query Language Injection
OWASP:	Open Web Application Security Project
DFD:	Data Flow Diagram
NETSPA:	Network Security Planning Architecture
NAT:	Network Address Translation
CSRF:	Cross Site Request Forgery
DoS:	Denial of Service
TAL:	Threat Agent Library
UML:	Unified Modeling Language
ADVISE:	Adversary View Security Evaluation
STRIDE:	Spoofing, Tampering, Repudiation, Information Disclosure and Elevation of Privilege
LMS:	Learning Management System
OBS:	Online Banking System

TMT: Threat Modelling Tool

TARA: Threat Analysis and Risk Assessment

PC: Personal Computer

CVSS: Common Vulnerability Scoring System

TVM: Total Vulnerability Measure

AGP: Attack Graph-Based Probabilistic

MPL: Mean Path Length

BN: Bayesian Network

MTTB: Mean Time to Breach

MTTR: Mean Time to Recovery

MTTF: Mean Time to Failure

ROC: Receiver Operating Characteristic

API: Application Program Interface

FPR: False Positive Rate

CUA: Continuous User Authentication

SAS: Security Analysis System

SIG: Soft Goal Interdependency

HLPN: High Level Petri Nets

DSM: Design Structured Matrix

URL: Uniform Resource Locator

PASTA: Process for Attack Simulation and Threat Analysis

VAST: Visual Agile and Simple Threat Modelling

DREAD: Damage, Reproducibility, Exploitability, Affected Users and Discoverability

DBMS: Database Management System

JS: JavaScript

EFSM: Extended Finite State Machine

ROC: Receiver Operating Characteristics

OVAL: Open Vulnerability and Assessment Language

WEKA: Waikato Environment for Knowledge Analysis

USB: Universal Serial Bus

TLS: Transport Layer Security

SSL: Secure Sockets Layer

SCAP: Security Content Automation Protocol

SOAP: Simple Object Access Protocol

CHAPTER 1: INTRODUCTION

With the constant increase in data breaches and attacks on businesses and large-scale organizations, threat modelling has become a critical part of any organization. Threat modelling involves recognizing the major assets of a system, as well as examining, recording and prioritizing the vulnerabilities detected. It also assists in locating the entry and exit point of the system (Priya & Arya, 2016). The entry point signifies the channel through which an attacker gains access to a system, while an exit point is the medium used by the attacker to leave the system after executing the attack. Threat modelling plays a vital role in developing secure web applications; it defines the threat subjected on to a system and the harm that could stem from the vulnerabilities.

Web applications communicate with the back-end servers to collect data and present to the user as a dynamically generated output using Hypertext Markup Language (HTML) web pages (Su & Wassermann, 2006). The major process involved in sending information from the client to the server entails the following - the user types the Uniform Resource Locator (URL) on the web page, the browser checks the cache for a Domain Name System (DNS) record to search for the Internet Protocol (IP) address linked to the URL. The caches checked include the browser cache, operating system cache, router cache and Internet Service Provider (ISP) cache. If the IP address is not available in the cache, ISP's DNS server starts a DNS query process to search for the IP address of the DNS server hosting the domain name of the URL through a forward lookup to the name servers. After the IP address has been found, the client browser initiates a Transmission Control Protocol (TCP) connection with the server. The browser sends an HTTP request to the web server through a GET or POST request. Post establishment of a TCP connection, the web server can handle the request and return a response which is often written in

compiled or scripting language. After the web handler has identified the major information, the server returns a Hypertext Transfer Protocol (HTTP) response for the browser to display the HTML web page. SQL injection (SQLi), if used as an attack, injects malicious codes into a web application to modify, update and delete the information from the database. The Cross-Site Scripting (XSS) attack executes malicious scripts in a legitimate web application to deface the page of the website or change the background color of the web application.

However, if the user's input is not properly handled, it could result in security problems as the queries are structured in a dynamic ad hoc way through low-level string manipulation. Improper handling of the user input causes the web applications to be prone to malicious attacks like SQLi attacks, XSS attacks, and Buffer Overflow attacks. There are three ways in which SQLi attacks can be classified - in-band SQLi, inferential SQLi and out-of-band SQLi. In-band SQLi uses the same transmission medium for launching the attack and gathering results (Muscat, 2017). Inferential SQLi operates based on logical reasoning and does not involve the transfer of data through the web application. Out-of-band SQLi attack uses the alternative channels for inference time-based techniques to obtain data from the server especially during fluctuating server response. An example of an SQLi attack is the breach on Heartland Payment Systems that exposed 134 million credit cards through the installation of a spyware on Heartland's data systems (Armerding, 2018).

Open Web Application Security Project (OWASP) also listed XSS and SQLi attacks among the top 10 attacks prevalent in web application vulnerabilities (Dionach, 2016). There are two major forms of XSS attacks - stored and reflected attack. Stored XSS attack, also known as persistent XSS attack is the more damaging of the two. It occurs when a malicious script is injected directly into a vulnerable web application. Reflected XSS attack involves the reflection

of a malicious script off a web application, onto a user's browser. The script is embedded into a link which is activated when the link is clicked (Rouse, 2017). Threat modelling of these application attacks is vital in strengthening an organization's cybersecurity posture. Research has shown that many organizations do not understand the technical know-how used in implementing threat models even though there are several threat models currently in existence (Vijayan, 2018). Most of these businesses are either too application centric, placing too much emphasis on the application or focusing on the vulnerabilities rather than the threats and some confuse threat mapping for threat modelling. All these stems from the uncertainty of what approach to use in modelling the threats in an organization. Hence, a comparison of these commonly used threat models needs to be simulated. There are different types of threat models in use including the Spoofing, Tampering, Repudiation, Information Disclosure and Elevation of Privilege (STRIDE) model, Diamond model, Kill Chain model, Process for Attack Simulation and Threat Analysis (PASTA) model, Damage, Reproducibility, Exploitability, Affected Users and Discoverability (DREAD) model and the Attack Tree model. In this study, three common models - STRIDE model, Kill Chain model, and Attack Tree model with features for practical simulations were used to model SQLi and XSS attacks using the following simulation tools - Microsoft Security Development Lifecycle (SDL) threat modelling tool, Trike threat modelling tool, and SeaMonster threat modelling tool.

1.1. Statement of the Problem

The problem of this study was to evaluate three threat models for SQLi and XSS attacks on web applications. Currently, there is no known scholarly study with a practical approach on the analysis of the different threat models.

1.2. Research Questions

The following questions were asked in this study:

(i) Which of the models contain the appropriate framework for modelling the attacks on web applications?

(ii) Which methodology uses the best approach based on the test of each model on SQLi and XSS attacks?

(iii) Can these threat models result in the mitigation and elimination of SQLi and XSS attacks?

1.3. Research Objectives

The objective of the study was to evaluate three different models on web application attacks to determine the strengths and weaknesses of each model and identify the most suitable threat model for web applications based on the three models. It also suggests ways to improve on the existing threat models to boost the web application security posture.

1.4. Significance of the Study

The results of this study will assist organizations in determining the most suitable threat model for their web applications through the analysis outlined in this study. This study contributes to the field of cybersecurity as it measured the efficacy of security initiatives through the vulnerabilities displayed all through the release cycles. This study will also assist in examining the state of cybersecurity in a web application. Finally, this is the first known scholarly study that compared three different threat models for web application attacks. The results of this study can be used by security administrators, developers and network security engineers in building the security framework of an organization.

1.5. Limitations

(i) The study placed emphasis on web applications. Although the use of web applications is ubiquitously global, other applications may also be subject to cyberattacks.

(ii) The research was carried out with a limited sample and configuration of devices. Due to the resource restrictions, it might yield a different outcome if the research was conducted with more devices.

(iii) The models were evaluated based on the tools available at the time. These tools often require constant updates and modifications.

1.6. Terminology

(i) **Structured Query Language Injection Attack (SQLi)**: This form of attack uses malicious SQL codes to manipulate the URLs based on SQL parameters at the backend thereby accessing information that was intended to be displayed. The information accessed could include sensitive data of the victim, user lists or private customer details (Imperva Incapsula, 2017).

(ii) **Cross-Site Scripting Attack (XSS)**: This is an injection attack carried out on web applications that accept input but do not properly separate data and executable codes before the input is delivered back to a user's browser (Rouse, 2017).

(iii) **Threat Modelling**: This is a process used to optimize network security by identifying the goals and vulnerabilities and outlining countermeasures to eliminate and mitigate the impacts of the threats on the system (OWASP, n.d.).

(iv) **Simulation:** This is the process of running a model. The objective of a simulation is to forecast the future behavior of a system and ascertain what should be done to impact the behavior (GoldSim, n.d.).

(v) **Uniform Resource Locator (URL):** This is a subset of uniform resource identifiers (URIs) (Digital Guide, 2018). It comprises of a protocol identifier and an IP address or a Fully Qualified Domain Name (FQDN). It is used in world wide web to identify resources on the internet.

(vi) **Hypertext Transfer Protocol (HTTP):** This is the fundamental protocol of the web and it controls the activities performed by web servers and web clients (browsers) in response to specific commands (Digital Guide, 2018).

(vii) **Data Flow Diagrams (DFDs):** This diagram models the flow of information for a process or system (Lucidchart, n.d.)**.**

CHAPTER 2: LITERATURE REVIEW

Threat modelling is an important aspect of operational and application security that involves the cooperation between security and other teams. It applies a disciplinary approach to solve problems (Shostack, 2018). The process of threat modelling improves software and network security by identifying and ranking the viable threats and vulnerabilities facing the software in an organization (Rubens, 2016). Threat analysis of a web application can result in a plethora of identified threats. Some of these threats include SQLi, XSS, Malware and DoS attacks. The aspect of threat modelling and analysis is a broad subject and requires in-depth research and evaluation.

2.1. Contributions and Findings

A major part of this chapter discussed the substantial contributions of various researchers to this field of study. The standard classification categorized the different studies according to the various features - model application, which involved the application of an existing model to an area of study, model development which involved the development of a model applied to a body of knowledge, and model analysis which entailed the examination of the existing model in relation to its application.

2.2. Model Application

Research by Ingols, Chu, Lippman, Webster, and Boyer (2009) utilized the application of threat modelling through the use of Attack Graphs to model modern network attacks and countermeasures. One of the major goals of the study was to envision the risks of threats posed to networks. The Network Security Planning Architecture (NETSPA) tool was developed to model oppositions and countermeasure effects by creating a network model which uses firewall

rules and network vulnerability scans. The model was used to calculate network reachability and Attack Graphs signifying potential attack paths for the opposition to exploit known vulnerabilities in server software. The research further evaluated the possibility of theorizing a zero-day vulnerability in each application on a network individually for proper understanding and the application of mitigation techniques such as building an Attack Graph for a new vulnerability to evaluate the effort put in by the attacker. With this in place, the defenders can place emphasis on the applications that provide more benefits to the attacker. Adequate modelling of zero-day vulnerabilities involve a list of servers and client software on each computer using the Security Content Automation Protocol (SCAP)-based common process enumeration. The effect and impact of vulnerabilities were the areas of Intrusion Prevention Systems that required modelling. The NETSPA tool was used to model these areas by adjusting the reachability model. The NETSPA network model assumed a single host comprised of one or more interfaces with a unique IP address. The interfaces could have more than one open port with vulnerability instances and flaws that could be exploitable by the attacker. Several rules were set on the firewall models to represent the set as binary decision diagrams (BDDs), FIREMAN system was also used to handle the filtering rules. The BuDDy's exist function handled the Network Address Translation (NAT) rules. In processing the NAT rules, different rules were put in place. First, the traffic was isolated so the rule matches, then reachability translated from one source address to the other before accepting the traffic. A track of all the chains reached were kept and reverse reachability carried out on the chains with potential exit nodes. To assess the scalability of the system, several tests were carried out on different synthetic networks using a 2.4GHz Pentium 4 computer running on Linux Operating System (OS). The test was further broken down into areas conducting it on full and partially synthetic networks

with or without firewalls and it resulted in a log-linear graph scale allowing the firewall rules as expected in practice. Assessing it on a real network behind a Juniper Netscreen firewall, an Open Vulnerability and Assessment Language (OVAL) scanner was deployed on Windows-based hosts to run a vulnerability scan on every user access. A random pick was done on the remote-to-root vulnerabilities from the results of the scan. Evaluating these random picks with NETSPA, it was discovered that major chunk of the server-side vulnerabilities and more than half of the hosts could be directly exploited.

Lin, Zavarsky, Ruhl, and Lindskog (2009) conducted a research on threat modelling and tree-based analysis for Cross Site Request Forgery (CSRF) attacks. Applying the classification of CSRF - reflected and stored, the threat model was carried out using a real world CSRF attack in a Chinese Social Network called the kaixin001 network. A reflected CSRF attack was launched with a link placed on a malicious web page so a victim could click on it. The goal of the CSRF attack was classified into three categories - the first objective involved the attacker accumulating bases for other attacks like SQLi and DoS attacks; the second objective was for entertainment purposes and the last objective was for financial reasons and commercial profit. The study also described how to achieve the goals and motivations of the attacker earlier highlighted by demonstrating the vulnerabilities that could be exploited. In most cases, the attacker explored the victim's web page to ascertain the functionality that would be of best use. An authorized user account was required on the target website before identifying the vulnerabilities used to test the efficacy of the malevolent actions. After a successful test, a malicious link was created on the web page for a user to launch the CSRF attack. A Data Flow Diagram (DFD) was used to describe the login process, authentication operations and the processes used by the attacker to explore the target website. One of the major weaknesses identified in the web application was the

improper security limitations which could be easily exploited by the attackers to upload malicious scripts such as placing CSRF worms on the victim's web server. Another weakness identified was the lack of restriction to upload attachments in the Web 2.0 applications. Another DFD was designed to illustrate the flow between the components during reflected CSRF attacks after which an Attack Tree model was designed for these attacks to specify a formal and systematic way of describing the security systems based on the several attacks. An Attack Tree was also created to compel a user to logout through stored CSRF attack. The steps to achieve this feat were recorded and used to create the model. Another tree was created to compel a user to logout through reflected CSRF attack with all the steps recorded. Some mitigation techniques were highlighted to reduce the vulnerabilities identified and session expiration dates set to defend against rolling reiterating session ID. For sensitive transactions against CSRF attacks, security tokens or two factor authentication methods were devised, and file extension type and size were formatted to limit upload attachments and input/output filters applied to prevent input validation errors.

Desmet, Jacobs, Piessens, and Joosen (2005) conducted a research on threat modelling for web services-based web applications. The web services communication model used in this study was the Simple Object Access Protocol (SOAP) messaging protocol for a document-based information flow using Microsoft threats and countermeasures as a guide. The essential resources within a web services-based web application were identified - application specific assets, web service specific technology artifacts, private information on the user's machine, availability of the several machines, connections and services in the architectural depiction. Different scenarios were illustrated with an attack entry point diagram of the feasible locations for a web service customer and web service provider. Assumptions were made to identify threats

for the strong cybersecurity posture of the company's network and servers. It was also assumed that attacks were often targeted at the server. Under the STRIDE threat classifications, different scenarios were used to describe the components involved and it was identified that the highest risk exists with the client. A synopsis of the countermeasure technologies was given with an appropriate guidance on the selection of the technologies used. The issues and questions related to selecting a type of technology were outlined for the designer. The following were identified as countermeasures for the STRIDE model - authentication countered spoofing, data protection countered tampering and information disclosure threats for data in transit. Authorization countered tampering and information disclosure threats on data inhabiting on servers, it also countered elevation of privilege and denial of service while input validation countered all of the STRIDE threats. Other countermeasure technologies mentioned include non-repudiation, sandboxing, secure coding, intrusion/fraud detection and other availability related countermeasures. In synopsis, the research was able to identify the various threats and countermeasures in using web service technologies for web applications with appropriate guidelines on how to mitigate the accompanying risks.

Nostro, Ceccareilli, Bondavalli, and Brancati (2013) conducted a research on the methodology of insider threats and attacks assessment and mitigation. The authors investigated the motive of an insider, potentiality, and critical importance of potential violations with countermeasures using a crisis management system Secure as a case study. An insider threat was defined as the misuse of given privileges in a way that constitutes threat in an environment. The intent of the research was to provide a clear-cut methodology to manage systems and related risk assessment process relevant to insider threats. Six constant phases were described to develop the methodology - the first phase was the system under analysis which involved a semi-formal

notation like DFDs, finite state machine and activity diagrams. The second phase comprised of ways to profile viable insiders identifying their features and criticality to the system using the Threat Agent Library (TAL) to achieve this feat. Some characteristics identified for the insider threats include intent, access, outcome, limits, resource, skill level, objective and visibility. The third phase involved the identification and description of potential threats in a vulnerable system; the damage was identified in this process with the resulting consequences. The fourth phase pointed out the path exploitable by the insider to determine the threat and achieve the objective. The various techniques used to achieve this include Attack Trees, Attack Graphs, and Privilege Graphs. The next phase comprised of ways used to select the countermeasures to mitigate the threats identified and this was achieved with a set of libraries that contained countermeasures for dogged attacks and other methods from the state of the art. The countermeasures were categorized into preventive, deterrent and detection. The last phase of the methodology iterated and updated the system by gathering feedbacks on the progression and changes in the system to update the insider threats library and the attack paths. All of these phases, with the exception of the last phase, were applied to the Secure project, a system subject to strict security and privacy needs. A Unified Modelling Language (UML) use case diagram was used to illustrate the scenarios in the different stages involved. A number of threats were identified with different levels of criticality related to the activities by the insiders. The attack path for Secure operations involved performance degradation of the system using the Adversary View Security Evaluation (ADVISE) model to create an attack execution graph. Another important goal represented in the ADVISE diagram for the system administrator was the theft of sensitive data and the delay of Secure Rescue operations. In relation to the attack goals, preventive, deterrent and detection countermeasures were recommended for the system administrators and the system expert. The

authors were able to recreate new considerations on the vulnerabilities in a system associated with insider threats.

Eng (2017) combined three main approaches for threat modelling - asset-centric, attacker-centric and software-centric to form integrated modelling. Questionnaires were used to obtain results which formed threat scenarios on Attack Trees. The integrated threat modelling approach was then applied in a hypothetical Software as a Service (SaaS) cloud solutions based on a public cloud model and results were investigated. The main problem identified in this study was the ways in which several threat modelling techniques could be combined highlighting their advantages and disadvantages. Asset-centric approach comprised of threat scenarios that could affect the organization's assets and attacker-centric approach used potential threat agents as a basis for searching for threat scenarios. Software-centric approach also referred to as system-centric modelled the software using the STRIDE model as a good prerequisite model of the system. Also, the use of DFDs were instrumental in achieving this feat. Three questionnaires were presented to respondents using a modelling approach. The information from the three questionnaires were collated and presented on the Attack Tree form. The Hypothetical Human Resource Management System (HRMS) was used as a foundation for testing and assessing the methodology for the data collected in the study. UML use case diagrams were used to illustrate the HRMS modules and the functions supported by the system. A total of 12 trees were produced by the three approaches but reduced to 6 trees upon completion of the merge process. Although there were no threats related to the cloud, the author proposed a threat model for cloud system as there were both similarities and differences between the several approaches to threat modelling which in turn suggested a comprehensive threat model with multiple approaches to achieve the desired objective.

Palanivel and Selvadurai (2014) applied the threat modelling approach in risk-driven security testing using risk analysis. To achieve risk-based testing, STRIDE model was combined with risk analysis. An Extended Finite State Machine (EFSM) diagram was designed to differentiate the states and the existing transitions in the system model with a workflow diagram for the state representation module. The risks identified for the different threats were pointed out using risk parameters - risk possibility, risk threshold and risk impact. The following modules were identified in the system - state representation module, threat modelling module, risk analysis module and test case selection module. The STRIDE threat model was applied to the elements of DFDs specifically on the elements data flow, data store, process and interactor. For a practical experiment on the assessment of risk analysis and threat modelling, the following systems were considered - Library Management System (LMS), Online Banking System (OBS), Movie Ticket Reservation System (MTRS), and Online Shopping System (OSS). It was observed from the analysis that the quantitative risk value calculated with the recommended system was three times better than the existing system. Hence, the application of STRIDE threat modelling in risks performed better than the current system.

Ma and Schmittner (2016) applied threat modelling in an automotive territory with focus on security analysis. The main research problem identified was the concern for security and safety of modern vehicles. Threat modelling was categorized into different phases of the development life cycle - concept, product development, production and operation. The major objective of the research was to model a system with architectural drawings and specifics to the element in DFD, pin-point threats that originated from the data flows using the STRIDE model - restructure the system with a higher level of mitigation to address each threat and authenticate the threat modelling diagram in comparison with the actual system addressing each of the threats

identified for accuracy of the results from modeling the threats. Using Threat Modeling Tool (TMT), stencils were created for each automotive component with further details to describe each component. A top level DFD diagram was used to illustrate the automotive unit. The authors used the experience of conducting Threat Analysis and Risk Assessment (TARA) of automotive cockpit unit well-found with wireless communication modules for different remote access functionalities involving maintenance and Over-The-Air (OTA) software update. It was established that threat modelling was an effective and beneficial analysis method for automotive security.

De Cock, Wouters, Schellekens, Singelee, and Preneel (2005) investigated threat modelling for security tokens in web applications. The authors assessed the threats that occurred using smart cards in web applications. The main objective focused on electronic Identity (eID) cards and smart cards. Different attack entry points were identified and described. Some of the entry points include the smart card itself and the reader, smart card driver software, the application, the user and web servers. The techniques employed to mitigate security threats include the following - technical solutions for the token, smart card reader, user's personal computer (PC), web server and some procedural mitigation techniques. A summary of the threats related to the use of smart cards in web applications was described in this study. Several threats were listed - the first threat described how the token would no longer be in possession of the legitimate holder while the second threat resulted into a damaged security token. The third threat entailed secret data extraction from the token and the fourth threat entailed avoiding the access control mechanisms of the security token. Remote private/secret operation was also identified as a threat where the attacker's main goal was to perform operations with cryptographic keys. Meddling with data in token was also identified as a threat. In synopsis, the authors identified the

threats on security tokens and the countermeasures for each of them. User awareness, education and training were recommended as the essential practices for the workplace and its environs.

2.3. Model Development

This category involved the introduction of a framework built from existing models and previous research. Jaganathan, Cherurveettil, and Sivashanmugam (2014) proposed a mathematical prediction model on the effect of an attack based on the distinct factors that influence cybersecurity. The model was designed to be proactive so that technical analysts could examine the data and predict possible results. The multiple regression method was selected to predict the impact of these cyberattacks. The factors that were identified to influence an attack in the study include level of security protection on the target system, use of traffic in the identified networks, existing vulnerabilities in the target system, and current value of assets in the network. The Common Vulnerability Scoring System (CVSS) was also considered as a metric to measure the impact to develop a prediction model. To achieve this objective, data points from the CVSS calculator were documented for every attack encountered including the network traffic. One of the results showed that there was a direct relationship between CVSS score, vulnerability, and network traffic. The what-if analysis was performed based on the chosen threshold values. Different scenarios were used and provided as inputs to the model. Through the prediction of CVSS scores, higher emphasis would be placed on vulnerabilities and high risks prevented.

Abraham and Nair (2015) proposed a randomly determined security framework for cybersecurity analytics with the use of Attack Graphs considering the factors connected with vulnerabilities that changes over time. The CVSS was used as a vulnerability scoring framework to achieve this objective. The model proposed in this study should assist in pointing out the

critical systems that should be hardened based on the possibility of intrusion by an attacker and the risk of being compromised. Based on this research, network security was classified into different categories. The first category is core metrics, which does not require any structure to enumerate the security of the framework. Examples of this category include the Total Vulnerability Measure (TVM) and Langweg Metric (LW). The next is structural metrics, which requires the fundamental structure of the Attack Graph to aggregate the security features of individual systems so as to measure network security. Examples include the Shortest Path (SP), Number of Paths (NP) and Mean Path Lengths (MPL). Probability-based metrics combines probabilities with single entities to measure the overall security state of the network. Examples of this include Attack Graph Based Probabilistic (AGP) and Bayesian Network (BN) based metrics. Time-based metric helps to measure the speed at which a network could be compromised. Examples of the metric under this category include Mean Time to Breach (MTTB), Mean Time to Recovery (MTTR) and Mean Time to Failure (MTTF). The author developed a framework that comprised of all the processes involved in building the security metric. The inter-relationship was captured using Attack Graphs and assumptions were made that time-parameters played a key role in capturing the developments of the attack process. Markov model was used as a modelling technique to carry out system performance analysis and dependability analysis. A high-level overview was introduced in the study on cyber security analytics architecture. The architecture was further split into four layers. Layer 1 was the Attack Graph model and layer 2 encompassed the CVSS framework. Layer 3 was a stochastic model which applied pertinent stochastic processes over the Attack Graph to explain the attacks with respect to the relationships amongst the vulnerabilities in a system. Layer 4 was structured around the vulnerability life cycle model that identified the trends and the evolving strategy of

network over time. The models were further represented with the following characteristics whilst absorbing the Markov chain - (a) an Attack Graph with a minimum of one absorbing state (b) the possibility of moving from the universal state to an absorbing state. The base exploitability score was computed to measure the intricacy in exploiting the vulnerability. Under the non-homogenous model, the temporal weight score of the vulnerabilities was also calculated based on the outcome of Frei's model. A quantitative analysis was carried out on the cyber security analytics model using the Node Rank Analysis (NRA) which defined the critical nodes in the Attack Graph most frequently visited by the attacker, Expected Path Length (EPL) metric which calculated the projected number of steps taken by the attacker from the initial state to the finished state of compromising the security system while Probabilistic Path (PP) metric which calculated the probability of getting an attacker to attain the absorbing states of the graph. In the Attack Graph generation, different scenarios were combined for the attacks to reach the objective state. Different open source Attack Graph tools like TVA, Attack Graph Toolkit, MulVal and NetSPA were analyzed and the MulVAL tool was chosen to generate a logical Attack Graph. The Attack Graph metrics were distributed over a period of 150 days and the general trend indicated that it required less effort from an attacker to compromise the security goal. A simulation of the Absorbing Markov Chain was conducted based on the generation of the Attack Graph from the network. The author's framework would assist the security engineer in making a practical and objective security assessment of the network.

Kozik, Choraś, Renk, and Hołubowicz (2016) proposed an algorithm for web application cyberattack detection in modelling the behavior of web applications with the information generated from HTTP requests from a client to a web server. Several tools were identified to search for vulnerabilities that could be exploited by an attacker. Examples of these tools are the

PhpMiner II, AMNESIA and STRANGER. Some of the tools used in cyber-attacks detection also include Snort, PHP-IDS and SCALP. The research method used machine learning hypothesis in presenting two distinctive phases on the diagram - the learning and the testing phase. In the learning phase, the labelled data was used to form the model parameters of the normal application behavior and in the testing phase the key was formed with the same procedure used earlier. For every HTTP request that comprised of the parameters, it was encoded to produce the characteristic vector with comprehensive information of when a given request is on the whitelist or not. Two separate algorithms were written to establish and encode text in the dictionary used for the HTTP parameters. The dataset used in this experiment comprised of thousands of HTTP protocol requests structured in a similar way to Apache Access Log. These datasets included generated traffic targeted to an e-Commerce web application. The data obtained was randomly spilt into 10 different parts with 10% used for assessment and the remaining 90% used for training purposes. The Receiver Operating Characteristics (ROC) curve for the various classifiers was then generated using the Waikato Environment for Knowledge Analysis (WEKA) tool. Based on the results generated in the research, it was concluded that there was an increasing number of attacks targeted on web-based applications.

Axelrad, Sticha, Brdiczka, and Shen (2013) proposed a BN model for forecasting insider threats and conducted a survey to measure the effect of the model. The BN model integrated psychological variables that signified a level of interest in malicious insiders. The development of the BN was split into different phases - the first phase identified and selected predictors, 83 variables related to insider threat were developed and each score estimated according to the rank and power of each variable. The BN then integrated these variables from the rank ordered list. The second phase contained an outline of the BN model indicating the categories of the variables

chosen for insertion. The categories include vigorous environmental stressors with the addition of personal life stressors and job stressors, stationary personal features with the addition of personality and capability, vigorous personal features including perceived stress and manner, insider actions and level of interest. The third phase estimated conditional probabilities using several sources for parameters drawn to explain the relationships between variables in the model. The techniques used to assess these parameters include the level of interest where inferences were made about the level of interest of individual with a degree of known and unknown characteristics, vigorous environmental stressors where the stress variables were connected in the following ways - (a) The general environmental stress directly connected to the level of interest (b) Both personal and job stressors that often times resulted in a decrease in job satisfaction (c) Stressors that might heighten the dynamic effect variable which in turn increases the level of interest. Under inactive personality, five personality factors were identified - neuroticism, conscientiousness, extraversion and candidness to experience. Capability was identified as a factor that influenced the level of interest and combination with hostile intent which increases the level of the threat. Counterproductive behavior was also identified as a function of the level of interest, job satisfaction and personality. A BN was designed based on the features mentioned. To validate the model, a psychological survey was conducted using the Amazon's Mechanical Turk but excluded some of the features highlighted in the BN model. The variables in the survey and model were assessed with established reliability and validity. Different methods were implanted to identify and exclude participants with unreliable answers. A structural equation model was developed with the highest possible estimation of variables and the results were integrated into the BN model. In testing the BN, the level of interest was the main hypothesis

variable. It was discovered that the revised model projected the simulated data less than the original model.

Caltagirone, Pendergast, and Betz (2013) proposed a Diamond model of intrusion analysis where an opposition deployed a capability over some structure against a victim. As the events were identified, the analysts or machines created the model's vertices linking each with edges to describe the relationship between the characters. The event in the Diamond model includes the following core features - adversary, capability, infrastructure and victim. The meta-features include the timestamp, phase, result, direction, methodology and resources. The authors used different axioms to illustrate the event lifecycle of the Diamond model. An extended Diamond model was highlighted to illustrate the relationship between the adversary-victim and the capability infrastructure, the authors used Venn diagram to describe the adversary-victim relationships. Centered approach is a distinct feature of the Diamond model that can be used to identify a new malicious activity and other connected features. Various types of this approach outlined include the victim-centered approach (where the data analyzed related to a viable victim uncovers other related elements), capability centered approach (which described the characteristics of a capability to identify other related elements in adversary operations), infrastructure centered approach (that placed emphasis on the malicious infrastructure of the adversary), adversary centered approach (where the adversary is being monitored closely to determine their infrastructure and capabilities), social-political centered approach and technology-centered approach. An activity Attack Graph was used in this research to identify the paths to be taken by an adversary while the activity threads defined the paths already taken by the adversary. Six unique steps were identified to be involved in the process in the activity groups - the first step is the analytic problem which can be solved by classification, the second

step is the feature selection where the event characters and adversary processes are selected, the third step is the creation which create the activity groups from the set of events and threads, the fourth set is the growth of the new events into the model, the fifth step is the analysis stage where the analytic groups are analyzed to attend to the analytic problem defined and the last step which also equates the sixth step redefines the activity group sequentially. The model specifically addressed the dependencies between the parts that make up the adversary.

Milton (2015) designed a user behavioral model in an intruder detector for continuous user authentication in web-based systems. Two classification techniques were used to carry out this experiment - binary and multi-class classifications. Different custom Application Program Interfaces (API) used to develop the infrastructure include the weblog data handler, a front-end API used to fill data and split into training and test sets as well as the test data manager, a java tool for keeping and retrieving keyword data. A non-relational document database called MongoDB was used as the back-end data storage system. The BerkeleyLM N-gram Language Model Library was used to build the n-gram models from the data. Different scenarios were used to illustrate the mode of operation of intruder detector. The real-world user data collection was to examine the efficiency of n-gram models for Continuous User Authentication (CUA). The practical CUA for web-based organizations produced the following results. In the role-based identification, the characters with the best f-measure achieved 82% and approximately 10% False Positive Rate (FPR) with bi-gram models. In the user-based identification, each user had a noteworthy effect on the user identification with the best measure of classification effectiveness (f-measure) at 88% and approximately 10% FPR giving an overall lower FPR in comparison with role-based models. In the identification of outliers, the best results stemmed from binary classification with an accuracy of 90%. The author created a novel keyword abstraction method

to pre-process huge volume of web logs and developed a consistent user authentication structure based on n-grams to categorize user sessions and separate usage profiles.

Red (2016) proposed an alternative Cyber Kill Chain model with expansion for embedded system architecture. Different themes were used to form this alternative model. The first theme explained how exploitable embedded systems modules can be. Embedded systems were reported to be more exploitable than the conventional computing equivalents. It also exemplified how the flaws in an embedded system allow adversarial action on the goal in a lesser number of steps. The second theme involved weaponization, delivery, installation or command and control (C2) phases against embedded systems. Embedded systems attacks were reported to show distinct patterns and the system exploitation excluded the requirement to follow-up installations outlined in Cyber Kill Chain models. The third theme comprised of limitations that differentiated embedded system threats and as a result of these limitations, several desired effects were accomplished with the whole extent of the application of the intrusion model. The fourth theme involved threat models that used the system engineering approach. The major constraints affecting embedded systems identified in the research include power, cost and form factor for negligible and dedicated functionality sets. The system engineering approach considered technical complexities used to locate the ideal path to achieve the adversarial threat goal against an embedded system which would also reduce the number of steps required to model attacks in a Kill Chain leading to a more efficient embedded system design. The UML was used in this research. The introduction of the systems engineering approach led to a solution with a viability to interdict an adversarial Kill Chain. The study took into consideration the main characteristics of the architecture-centric, asset-centric and attacker-centric approaches in embedded systems and applied each of these features in the Cyber Kill Chain.

2.4. Model Analysis

This category involved the assessment and analysis of the existing models for possible improvements and modifications. Al-Mohannnadi et al. (2016) gave an overview of the different cyber-attack modelling analysis techniques currently in existence namely - Attack Graph, Kill Chain and Diamond model. The authors explained the basic elements in a Diamond model which include the adversary, infrastructure, capability and victim. The seven steps involved in the Kill Chain model were also highlighted. The first step described the process of how the attacker collects information through public means prior to the attack. In the second step, a malicious payload was created by the attacker before it was sent to the target. The malicious payload was sent to the victim in the third step before the exploitation in the fourth step. In the fifth step, the malware was installed on the targets' computer and through this installation a channel was created for access to the target victims' data for the attacker. The final step achieved the set-out goal. A case study was used to analyze the three modelling techniques to examine the behavior of each model to a cyberattack. Two scenarios were put into consideration in this study where the first victim was an employee of an organization and a computer user. Phishing attack was used in sending an email with a Portable Executable (PE) file with an extension (.exe) executed by the employee, until this was eventually identified by the security team in the company. Another analysis was also carried out with the Diamond model and different questions were developed for the four components that make up the model - victim, capability, infrastructure and adversary. Analysis was also carried out on the Kill Chain model to understand the attack in detail through the seven steps involved in the model. The final analysis was conducted with the Attack Graph model using the various paths used by the attacker to gain control over the victim's network. To further illustrate the Attack Graph model, a flow diagram was used to describe the

process. The uniqueness of each of these models were identified in different ways and the common characteristics highlighted.

Liu (2016) conducted a research on the analysis and detection of cyberattacks in smart home cyber-physical energy systems. The study was conducted to examine the vulnerability of the smart home cyber-physical systems. The effect of pricing cyberattacks, and energy theft was assessed, and detection techniques were recommended. Long term detection techniques were developed for pricing cyberattacks and energy theft using the Markov decision process. The results from the simulation in the study indicated that the detection technique proposed could lower the ultimate energy load and electricity bills produced by the cyberattacks without an increased labor costs. In addition, a cross entropy state was proposed to resolve the solution of the decision process.

Kotenko and Stepashkin (2006) suggested a new method of security evaluation based on the all-inclusive simulation of malefactor's actions and structure of Attack Graphs. The authors created a model of Security Analysis System (SAS) based on the inputs of several security factors and construction of Attack Graphs. Some functions identified the fundamentals required for SAS which include replicating malefactor's activity, constructing viable assault actions graph, assessing malefactors' actions from several network points and directing them to execute different security threats, exposing vulnerabilities and security weak points, computing various security factors and analyzing the general security level. The attack scenarios were classified into three levels - integrated level, script level and action level. To evaluate the level of security, the authors used the qualitative express evaluation based on qualitative methodologies of risk analysis constructing the different factors based on the Attack Graph. The security metric used in the study includes metrics based on network configuration, metrics of hosts, metrics of attack

actions, metrics of attack routes, metrics of threats and common Attack Graph. Devices were set up for the test computer network to denote the Attack Graph. A step-by-step process was outlined to illustrate the procedures involved in building an Attack Graph.

Shar, Tan, and Briand (2013) proposed a prediction based on the classification and clustering of SQLi or XSS attacks to forecast vulnerabilities. Some characteristics were developed based on data dependence and static analysis classifications. Some dynamic features were also identified - numeric, URL, HTML tag, limited length and the event handler. Based on the information in the study, a vulnerability prediction framework was developed for data preprocessing, constructing supervised and unsupervised vulnerability predictors. Testing and training were conducted for the 10-fold cross validation setup. The tool called the PhpMiner was modified and evaluated with the data collected. The experiments were carried out on the real-world PHP based web applications. In the different scenarios, it was observed that the hybrid features can be used to forecast vulnerabilities.

Zaeem, Manoharan, Yang, and Barber (2016) examined the behavior of identity threat through text mining of identity behaviors through the collection of news stories and reports on related topics. The patterns and resources used by the thieves were statistically analyzed through several text mining methods. The authors' objective was to create a fount of knowledge to understand the techniques used by the identity thieves and swindlers. The research successfully applied natural language processing to extract data about habits and techniques of identity thieves. It was also the first of its kind to use news stories from over 3,500 articles to analyze risks, losses and movements in identity theft further comparing them with over 250 manually studied identity theft stories. The sources of data used for the research include agency data,

surveys, case studies from victims and news articles. This research was carried out based on the related work to natural language processing on the Identity Threat Assessment and Prediction (ITAP) Project at the Center for Identity at the University of Texas at Austin. The various methods used in text mining applied in this analysis include Text Preprocessing (TP) which removed the factors dependent on language for better clarity, Noun Phrases (NP) which assisted in the collection of informative characteristics and meaningful data than a single word, named entity recognition outlined the detailed information of persons. A pipe-lined system model was designed to extract the identity news stories from the internet and generate analytics with a better understanding of the process as output. The first step involved in this process collected the article links from the internet, the second step searched for identity theft stories by manually constructed key words used for ease of process and the third step mined theft reports from the Identity Theft Resource Center. With all the information extracted, an identity theft record was created and further analyzed. The identity theft record was developed by defining the characteristics of Personal Identifiable Information (PII), impacted target, risk calculation, PII category, time of occurrence, location statistics and loss calculated. Validating the efficacy of the text mining techniques with manually investigated theft stories totally aligned with minor differences.

Oladimeji, Sapakkul, and Chung (2006) proposed an objective-oriented approach to security threat modelling and analysis with the use of visual model elements to apprehend threat related concepts. The authors used an online banking system to illustrate the threat modelling and analysis process based on the Non-Functional Requirements (NFR) framework. The processes used were recorded in a Soft Goal Interdependency Graph (SIG). The SIG properly documented the processes involved in logging in to a secured online banking system. To analyze

the adopted framework, a notion of Negative-soft (N-soft) goal and an inverse contribution was introduced to the system. An activity diagram was used to depict the threat modelling and analysis process. It includes four high-level steps which covered the definition of security to the system, reducing and examining threats with their accompanying risks and analyzing how countermeasures result to the achievements of security goals. All the documented steps formed the threat model in the entire development life cycle. A metamodel was created to define the connections amongst the conceptual model elements. The following sources of threat data were used in threat elicitation - system goals, threat catalogs, STRIDE classification schemes, system boundary assessment and insider threat. The DREAD process was used to define and hierarchize risks. In analyzing the threat model, the following steps were put to action - threat decomposition, threat ranking and severity assessment. After evaluating the threat SIG, it was discovered that all the expert judgements indicated by the claims soft goals were advocated to be valid resulting to a satisficing high-level security soft goal.

Manzoor, Zhang, and Suri (2018) conducted a comprehensive threat analysis approach across a multilayer cloud operational stack that captured the operational behavior of practical cloud implementations using OpenStack. This approach was developed to address the scalability issues on the increasing number of vulnerabilities in the cloud. A layered cloud model was developed using High Level Petri Nets (HLPN) to profile the behavioral attributes of cloud services and flow of data. A Design Structured Matrix (DSM) approach was also developed to examine security threats by inspecting attack surfaces from vulnerabilities all through the layers of the Cloud model.

CHAPTER 3: EXPERIMENTAL METHODOLOGY

Three different models used for the SQLi and XSS attack were outlined and described. It starts with the description of each model and the tools used in modelling the web application attacks. The threat models used in this experiment include the STRIDE Model, Attack Graph and the Kill Chain model. Although web applications are used by many individuals, they are often developed without security in mind and contain serious security vulnerabilities. For example, prior to integrating a database to a web application, a database user must be identified before any communication can be established with the backend Database Management System (DBMS). An unauthorized user should not be allowed access without the use of valid credentials. However, malicious users can still gain access through a manufactured SQLi and XSS attack. Figure 1 shows how a web application can be accessed through a web browser it goes through a firewall and establishes a connection to the webserver using a scripting language either in JavaScript or PHP to communicate with the database server, SQL database was used as the database server. The fundamental structure of a compromised web application for SQLi and XSS attacks is illustrated in Figure 2.

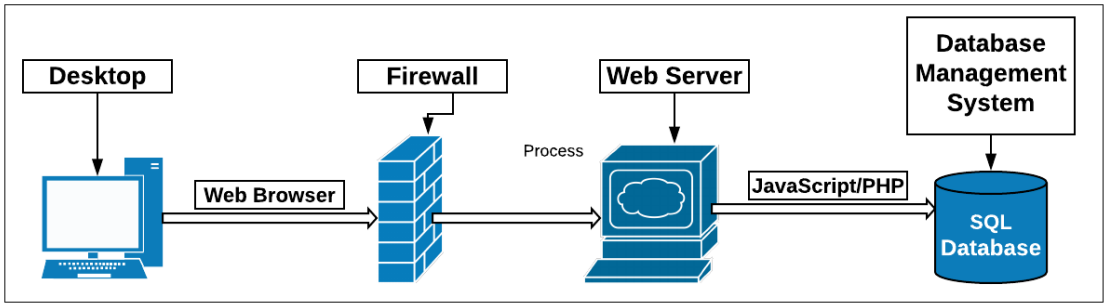


Figure 1. Basic web application architecture. This figure illustrates the fundamental structure of a web application.

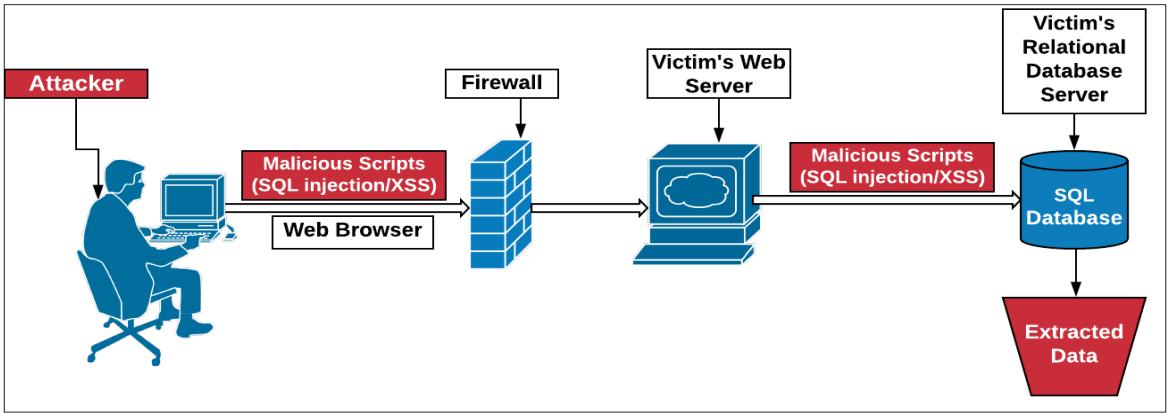


Figure 2. Compromised web application architecture. This figure illustrates a basic web application architecture after it has been compromised by SQLi or XSS attack.

Figure 2 describes the underlying infrastructure behind an SQLi attack or a XSS Attack on a web application. The attacker accessed the website through a browser and injected malicious scripts through the web page. The SQLi was performed through the username and password column of the web page, the malicious scripts go past the firewall to the victim's webserver and a successful entry would reproduce an exploitable URL. This exploitable URL would grant the attacker access to the database to extract, modify, update or delete information from the database. The three threat models highlighted above would be modelled using the Microsoft SDL threat modelling tool, Trike modelling tool and the SeaMonster modelling tool. Each of the models were modelled based on the following scenarios.

3.1. Description of the Virtual Environment

This study was carried out in a virtual environment for practical simulation purposes and accurate results. No mitigation measures were put in place, the firewall was not enabled, and the input statements were not sanitized. For both scenarios, the attacker exploited the environment using the penetration testing tools in Kali Linux OS. The target victim being exploited was the

Linux, Apache, MySQL and PHP (LAMP) Server on CentOS operating system. The penetration testing tools used in scenario one and two include sqlmap, DirBuster, iptables and nmap.

3.2. Steps Involved in Scenario One: SQLi Attack

In this scenario, the malicious user entered a malicious code in the login page of a web application. After series of attempts there was a successful login into the web application. In addition, the URL was copied to the vulnerability scanning tool on the Kali Linux OS to determine the vulnerabilities in the application. Once these vulnerabilities were found, a tool in Linux OS known as sqlmap scanned the malicious URL to extract vital information from the database. The information extracted which included PII was used to update the existing information on the database using a vulnerable web page detected through a tool on Kali Linux called DirBuster. A web page was created for this exercise and no mitigation measures were carried out. The following steps were drafted from the scenarios to create the fundamental structure for the three models.

Step 1: The attacker scanned the network with a vulnerability scanning device called nmap in a Kali Linux OS and discovered a target victim on CentOS with open ports that appeared vulnerable and exploitable. The scan results are shown in Figure 3.

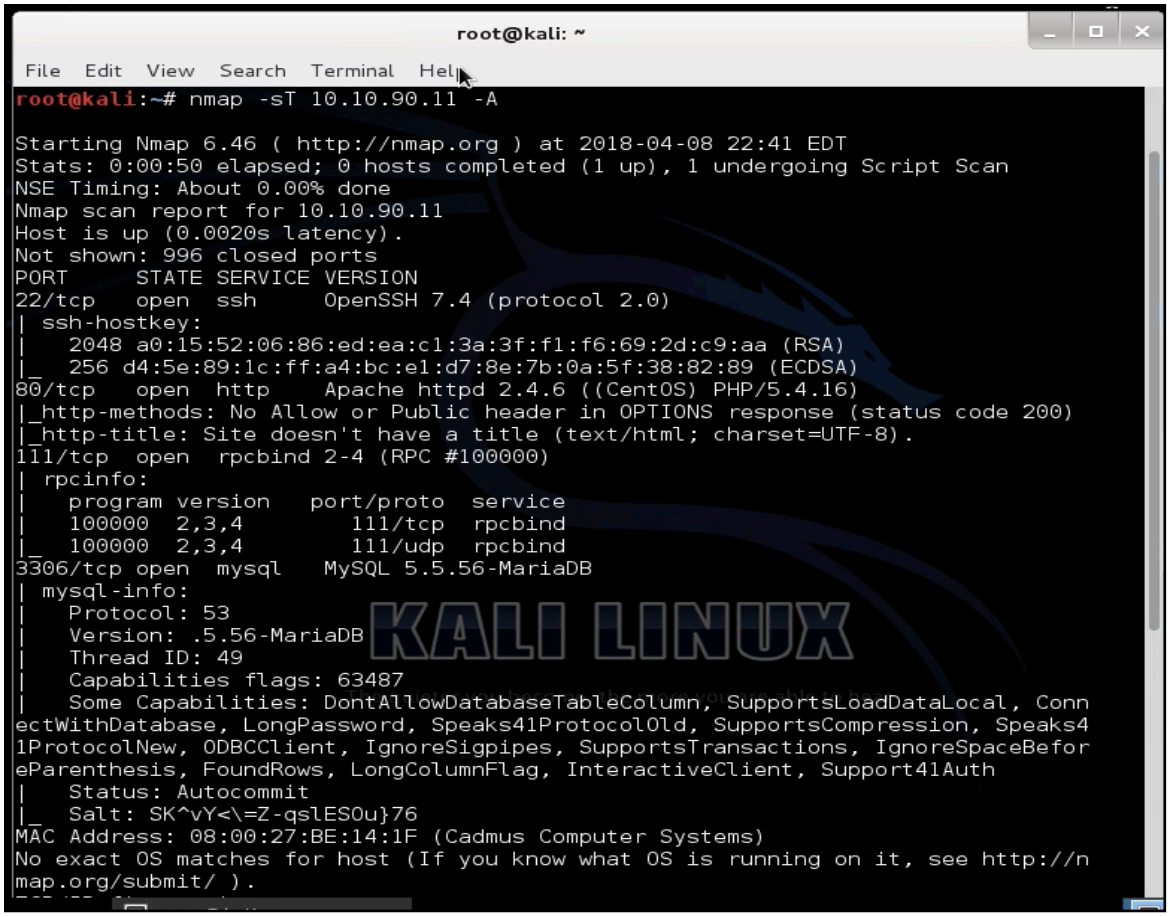


Figure 3. Vulnerability scan for SQLi attack using nmap for scenario one. This figure shows how nmap scanned the target IP address for open ports.

Step 2: After discovering the IP address and open ports, the malicious user attempted to gain access by entering malicious scripts on the login page of the web address. The malicious input shown in Figure 4 was a crafted SQLi, `' OR '1'='1` placed into the password field.

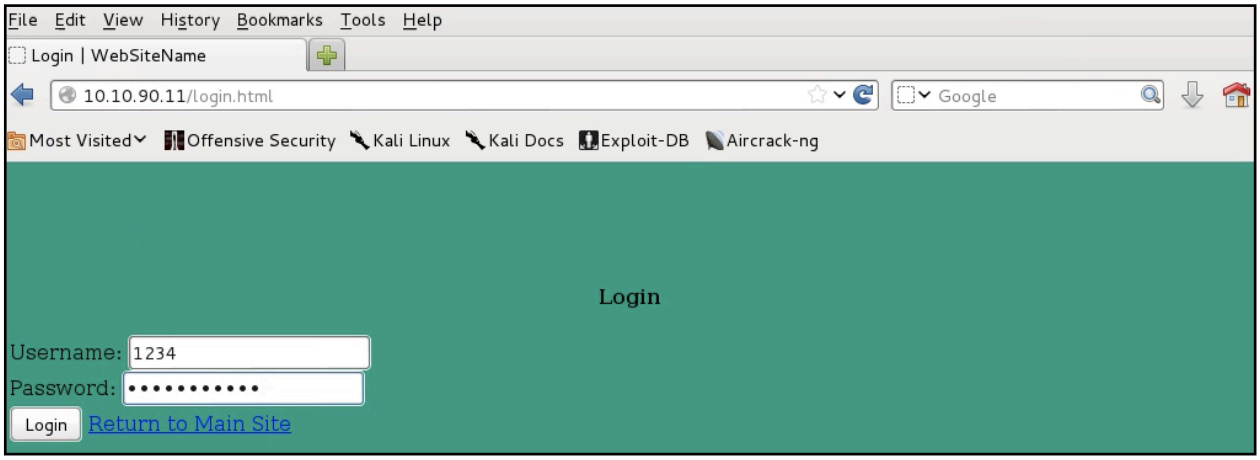


Figure 4. Inputting crafted SQLi on the login page of the target IP address. This figure shows how crafted malicious SQLi was entered on the login page of the discovered IP address.

Step 3: The attacker bypassed login after a series of attempts and copied the URL to a vulnerability scanning tool shown in Figure 5 for nefarious purposes.

Figure 5. URL capture for SQLi on the web application for scenario one. This figure shows how the extracted URL was captured before it was used for malicious purposes. The URL captured was copied to the sqlmap tool to extract data.

Step 4: The vulnerability tool sqlmap exploited the URL and used it to extract information from the database by determining the exploitable addresses and providing the information stored in the database. The extraction process is shown in Figure 6, Figure 7 and Figure 8.

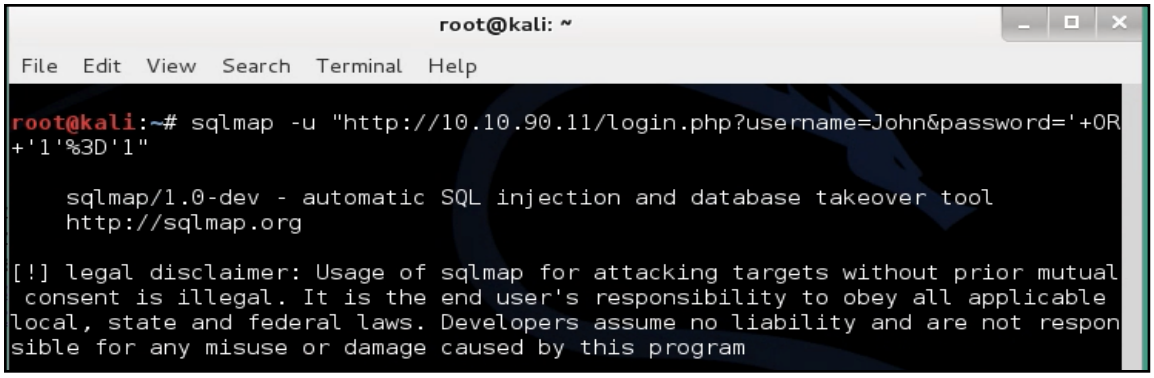


Figure 6. Sqlmap data extraction process from the web application for scenario one. This figure shows the use of the URL copied to extract data using the sqlmap tool. The URL was paired with the sqlmap in Kali Linux to begin the information extraction process.

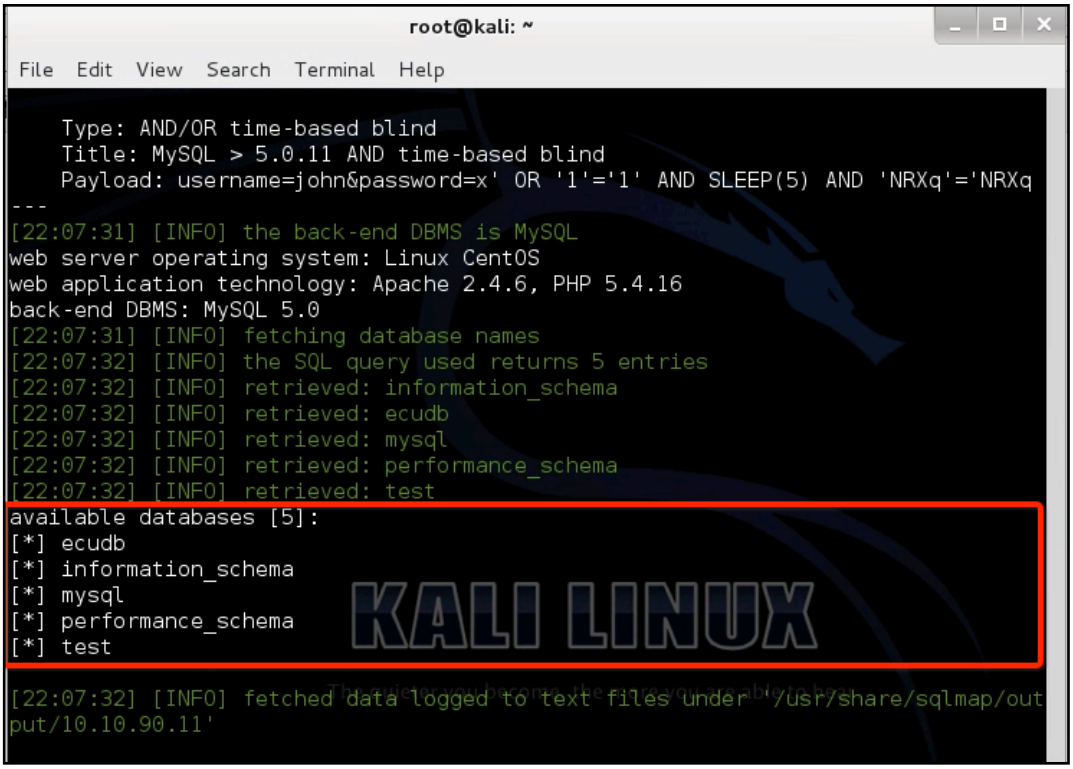


Figure 7. Sqlmap data extracted from the web application for scenario one. This figure shows some of the data extracted from the SQL database using the sqlmap tool.

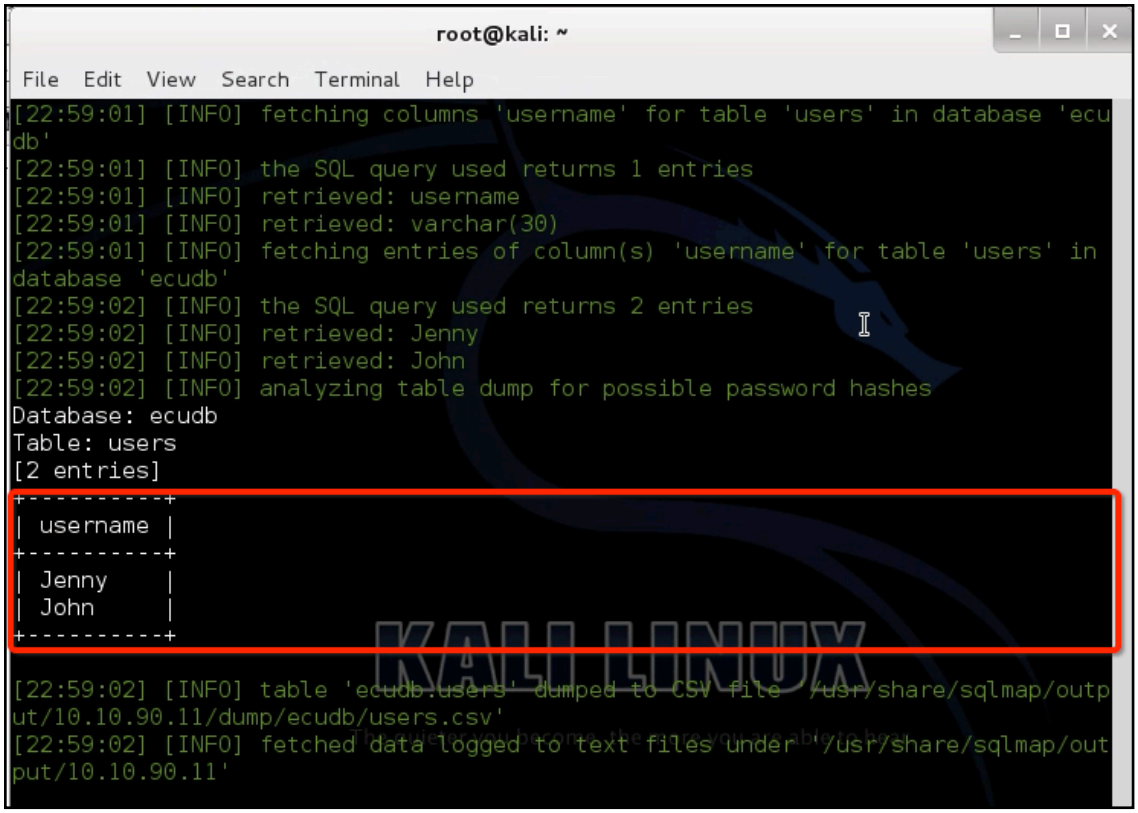


Figure 8. Sqlmap complete data extracted from the web application for scenario one. This figure shows the required data has been extracted from the SQL database using the sqlmap tool.

Step 5: The vulnerability tool called DirBuster identified the vulnerable part of the web page that was used to update the existing user account information. The scan results are shown in Figure 9.

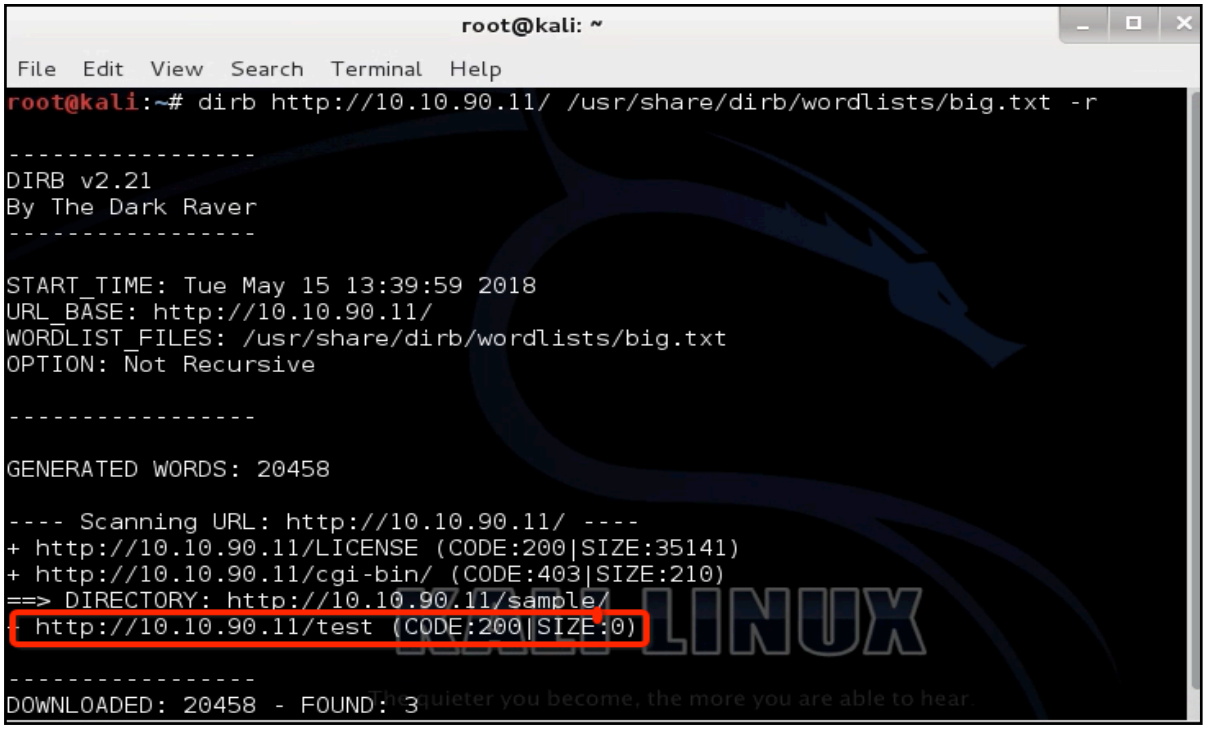


Figure 9. DirBuster tool scan for a vulnerable web page for scenario one. This figure shows the results of the scan for a vulnerable web page using the DirBuster tool.

Step 6: Information extracted using the sqlmap tool assisted in attacking the exploitable website, update user account information and prevented the main account holder from gaining access to his account. The crafted malicious SQLi statement was entered in the username field shown in Figure 10 to change the password of the user **John**. The user **John** was unable to gain access to his account with the old password after a new password was updated by the attacker.

```
1', 'john', '123456') ON DUPLICATE KEY UPDATE password = '2156';
-- -
```

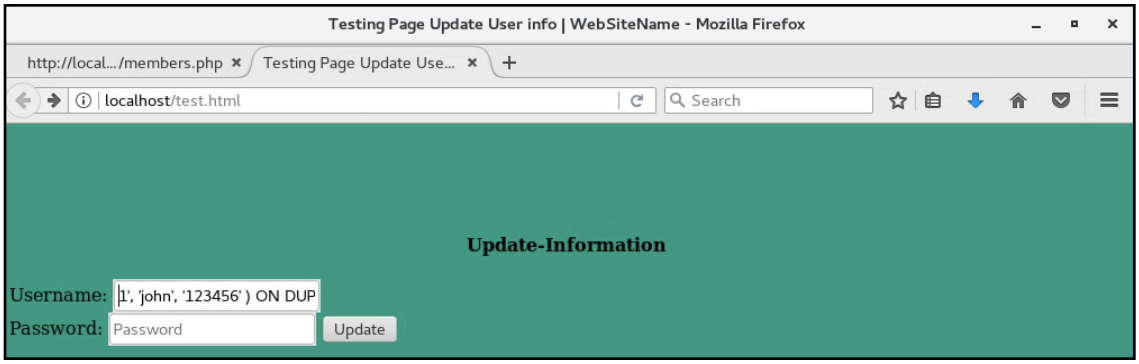


Figure 10. Malicious scripts inputted on the web page for scenario one. This figure shows how the crafted SQLi statements was entered on the username field vulnerable web page to update John's account information.

3.3. Steps Involved in Scenario Two: XSS Attack

In this scenario, an attacker entered a malicious code written with the HTML script tag in the comment section of a web page to deface the website with a change in background color and text.

Step 1: The attacker scanned the network with a vulnerability scanning device called nmap in a Kali Linux OS and discovered a target victim on CentOS with open ports that appeared vulnerable and exploitable. The scan results are shown in Figure 11.

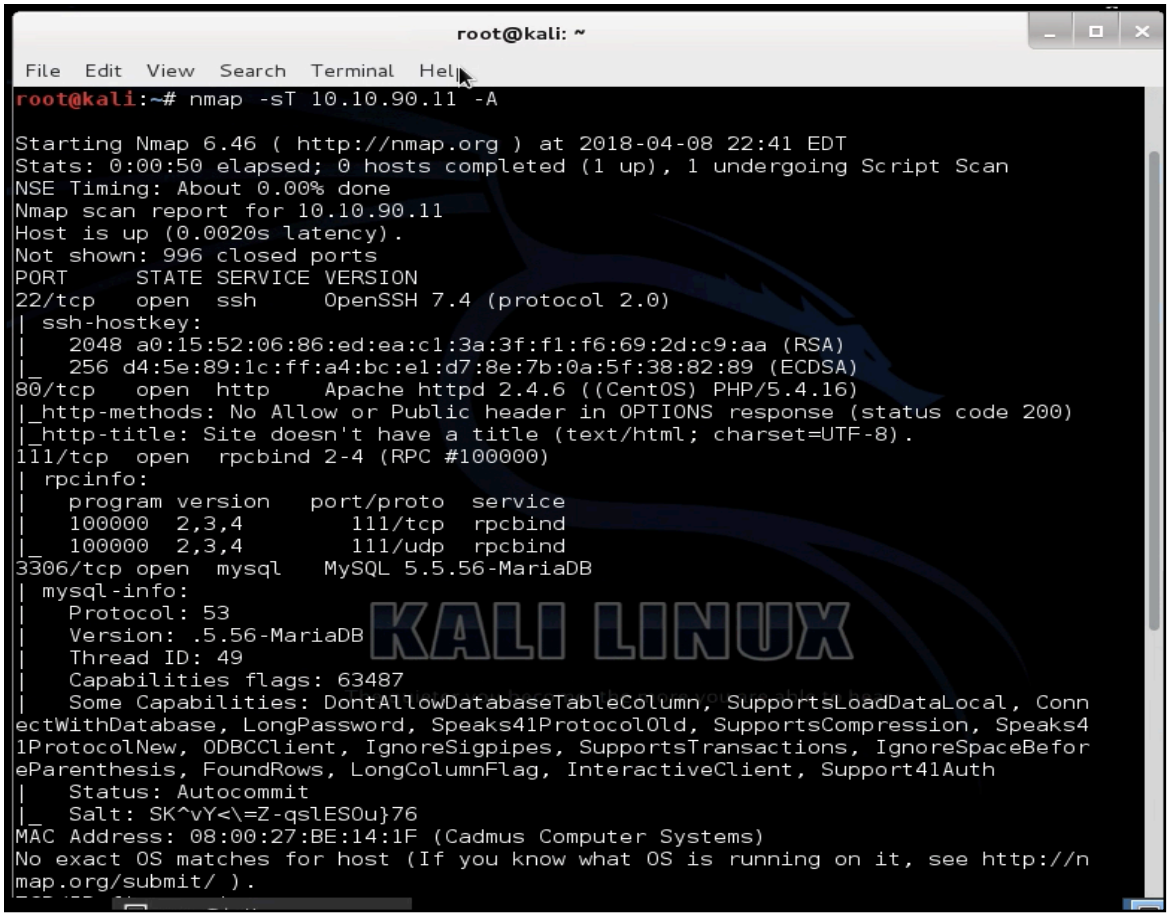


Figure 11. Vulnerability scan for XSS attack using nmap for scenario two. This figure shows how nmap scanned the target IP address for open ports.

Step 2: After discovering the IP address and open ports, the malicious user created a new account on the web page shown in Figure 12.

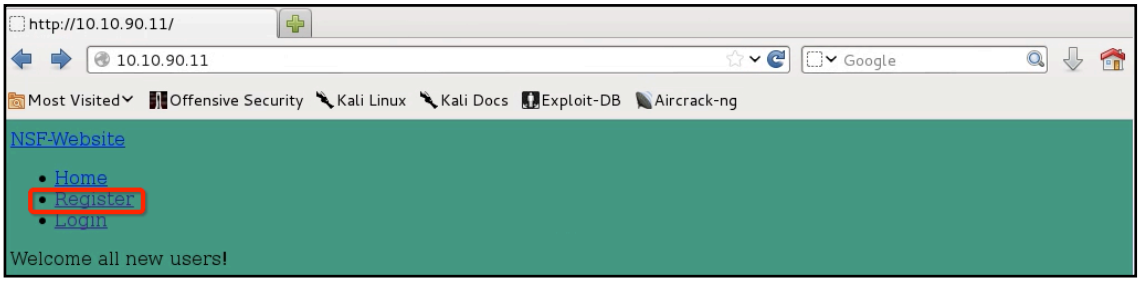


Figure 12. Registering a new account for XSS attack. This figure shows how a malicious user registered a new user account for malicious purposes.

Step 3: With the new account created, the attacker entered malicious scripts on the comments section of the web page shown in Figure 13. The malicious script used in this study is outlined below.

```
<body style="background-color:red;">

<h1> This site has been hacked!!! </h1>

<script>alert("You have been hacked") </script>
```

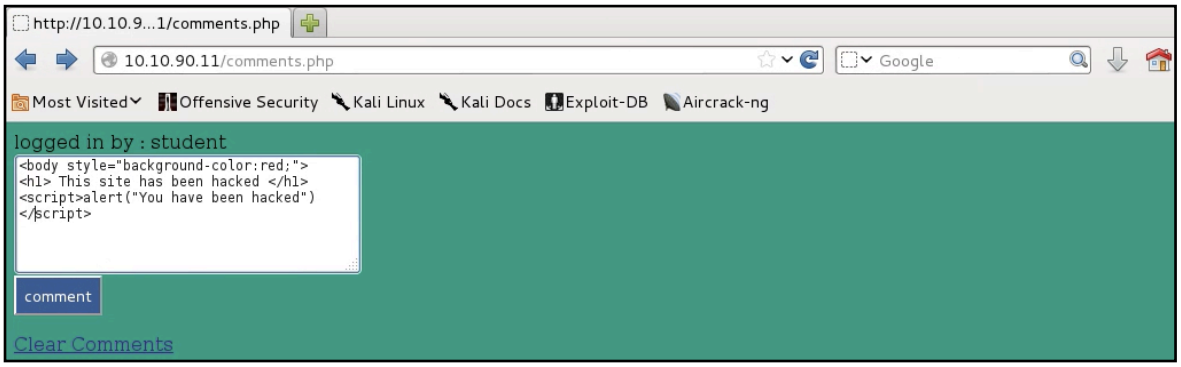


Figure 13. Malicious user input XSS scripts on comments page. This figure shows the XSS scripts entered by the attacker on the comments section of the web page to deface the background color and text.

Step 4: Upon successful posting, the web page was defaced. The color of the web page background was changed, and text was displayed in Figure 14 indicating that the website had been hacked.

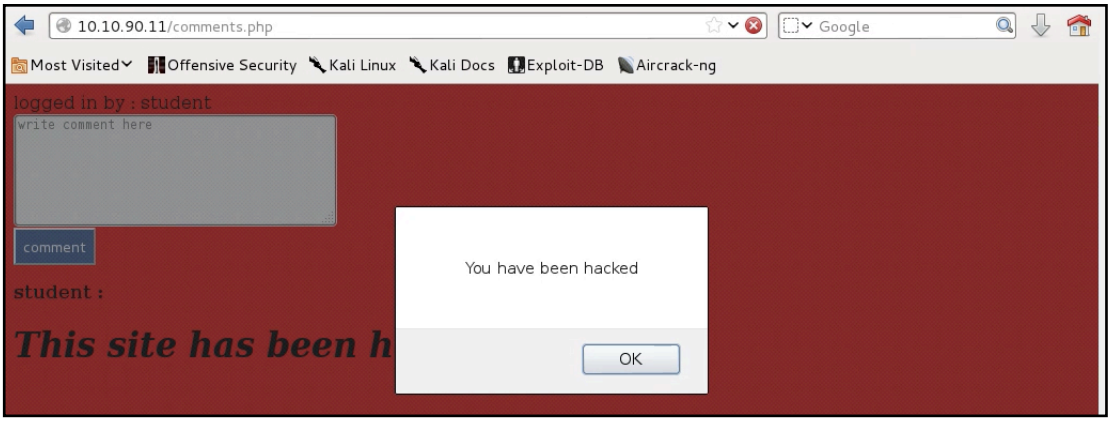


Figure 14. Defaced web page by the malicious user for scenario two. This figure shows the web page defaced through the input of the XSS scripts.

3.4. STRIDE Threat Model

STRIDE was invented by Loren Kohnfelder and Praerit Garg (Shostack, 2014). The major aim was to assist individuals in developing software to identify the various forms of attacks experienced by software. Spoofing involves masquerading as someone other than yourself. The attacker can either spoof a process on the same machine, a file, a person or a role. Tampering can be used to modify a file on a disk, network or memory (Shostack, 2014). Attackers can modify and alter files where they have write permission. Repudiation involves not claiming responsibility for an occurrence. It can be done in a deceptive or honest manner. Actions, processes or logs can be repudiated. Information Disclosure threats entails divulging information to unauthorized parties. This data can be from a process, data store or a data flow. DoS consumes resources needed to provide service. It can also be against a process, data store or a data flow. This model is the direct opposite of the expected properties of a system including authenticity, integrity, non-repudiation, confidentiality, availability and authorization. Table 1 explains the STRIDE model in detail.

Table 1

Stride threats and victims. This table adopted from Shostack (2014) describes the STRIDE threats, accompanying features, definition, victims and examples. It also shows the different actions that results into the violation of the STRIDE model and level of interactions involved.

Threat	**Violated**	**Definition**	**Victims**	**Examples**
Spoofing	Authentication	Masquerading to be something or someone other than yourself	Processes, external entities, people	Falsely claiming to be wellsfargo.com or Barack Obama
Tampering	Integrity	Altering something on disk or on the network.	Data stores, data flows, processes	Modifying the contents of a database, adding or removing packets over a network
Repudiation	Non-Repudiation	Denial of truth or validity of something. It usually raises the questions for system designers on the kind of evidence presented.	Process	It could either be a process or a system. "I didn't press the green button, I didn't hit the halt button."
Information Disclosure	Confidentiality	Disclosing information to unauthorized parties.	Processes, data stores, data flows	Access to files, e-mails, and databases.
Denial of Service	Availability	Absorbing resources needed to provide service	Processes, data stores, data flows	It uses up of all its memory through several network connections thereby making it unavailable to users.
Elevation of Privilege	Authorization	Granting access for unauthorized users to perform various activities	Process	Allowing remote access without any privilege to run code.

As seen in Table 1, falsely claiming to be wellsfargo.com violated the authentication feature and affected the spoofing category in the STRIDE model while unauthorized access to a

user to run a code violated the authorization feature and directly affected the elevation of privilege in the model.

3.4.1. STRIDE Threat Model Against A Standard Web Application

Several assumptions were made in this category. The following notions were made from accessing the threats through the STRIDE category.

- **Spoofing**: It was assumed that a web client could try different methods to log in with random or stolen credentials in the same manner as an SQL database client. Spoofed clients could bypass security checks if there was an assumption that the SQL client made the security checks and a false front-end connection to the web client could result in divulging credentials.
- **Tampering**: The program or data files being sent could be tampered with by an unknown individual. Sometimes, the SQL database or web client could also be tampered with.
- **Repudiation**: In this case, the web clients could deny the validity of something. Although, these threats could be prevented with the use of logs and log analysis.
- **Information Disclosure**: This had to do with disclosure issues. It involved exposing information stored in the database to the wrong client. The exposed information could either be data (credit card information of customers) or metadata (journalist's call records). An example of this is the illegal access of call records in Australia (Kelly, 2017). The OS and Access Control Lists (ACL) should protect the database files that contained storage and partitions. The log files encompassed confidential information that needed to be protected.

- **Denial of Service**: In this case, the front-end application could be overwhelmed with series of crafted requests, also crowding the network connections with a voluminous amount of data. The logs or database could be filled up in most cases.
- **Elevation of Privilege**: Here, several queries were run by unauthorized web clients. In cases where the web client enforced security, any authorized access would be granted. If there was a connection from the database cluster to a corporate directory service, then there is no restriction on the login access to the database servers. Easy access was granted to anyone in the corporate directory to make changes to the systems.

Bertino, Bruschi, Franzoni, Nai-Fovino, and Valtolina (2005) identified steps to be taken in modelling threats for a web application. These steps involved the identification of assets that required protection which comprised of the data stored in the database and data management systems. Table 2 describes the mapping of STRIDE model with various attacks.

Table 2

Mapping STRIDE model with other attacks. This table shows the areas where the different attacks are relevant in the STRIDE model categories. The symbol 'X' signifies the evident areas for each of the attacks. In comparison with other attacks, SQLi and XSS attacks were applicable in spoofing, tampering, information disclosure and elevation of privilege categories of the STRIDE model.

Attacks	S	T	R	I	D	E
SQL Injection	X	X		X		X
Cross Site Scripting	X	X		X		X
Denial of Service					X	
Network Eavesdropping	X			X		X
Malicious Data Mining				X		
Unauthorized Access	X	X		X		X

3.4.2. Microsoft SDL Threat Modelling Tool Simulation

Microsoft SDL Threat Modelling Tool was developed not only for security experts but also for developers and software architects for instructions on creating and examining threat models (Microsoft, n.d.). Its main objectives include to enable any developer to speak about the security design of their systems, examine the scheme for viable security issues through different techniques and to proffer mitigations for the security issues. The first step includes drawing the diagram of the design of the software system on the SDL threat modelling tool. The various units embedded in the architecture include processes, data stores and communication channels. The tool used built in rules from Microsoft's STRIDE threat model, appropriately applying it to a user's architecture. Modelling the basic web application architecture produced the output in Figure 15 on the threat modelling tool.

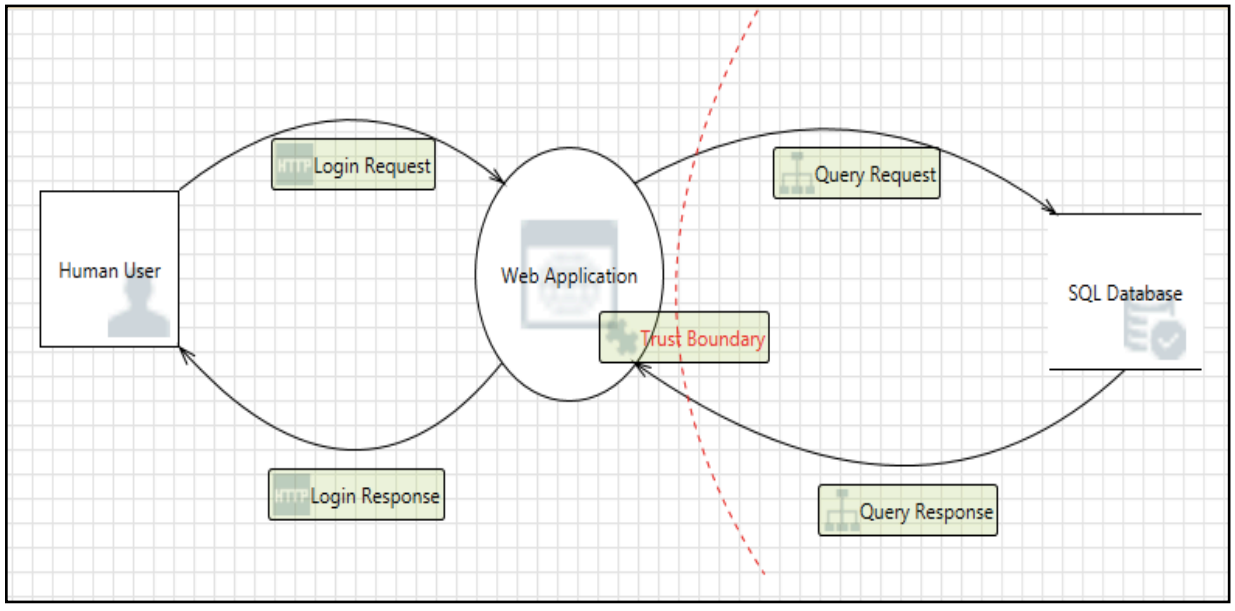


Figure 15. Web application threat modelling using the SDL threat tool. This figure illustrates the threat modelling of a basic web architecture for SQLi and XSS attack.

Figure 15 shows the high-level breakdown of the STRIDE model applied to each interaction of a web application for SQLi and XSS attack using a data DFD on the tool based on

both scenarios earlier highlighted. Both scenarios were infused together to produce the threat model because the SDL threat modelling tool models an application based on the fundamental architecture of a web application. Further analysis generated a list of threats from the web application architecture which included the SQLi and XSS attacks. Each of the threats identified required inputs for the mitigation measures for the different scenarios.

3.5. Kill Chain Model

The Kill Chain model was initially used in the military. It involves modelling and analyzing the steps and actions of a cyber attacker (Yadav & Rao, 2015). It is particularly essential for security analysts to be involved in the development of countermeasures. The word "compressing the kill chain" was coined by General John Jumper in October 1996 to validate the techniques required to compress the time it takes to search for and eliminate the enemies on the battleground (Security Boulevard, 2018, para 1). The Cyber Kill Chain structure was developed by Lockheed Martin as part of the Intelligence Driven Defense model for the identification and prevention of cyber intrusion activities. The model pinpoints what the adversaries must complete to achieve their aim.

3.5.1. Kill Chain Model Mode of Operation

The Advanced Persistent Threat (APT) Documentation of Lockheed Martin highlighted the seven stages involved in the Cyber Kill Chain model (Lockheed Martin, n.d.). Figure 16 describes the stages involved.

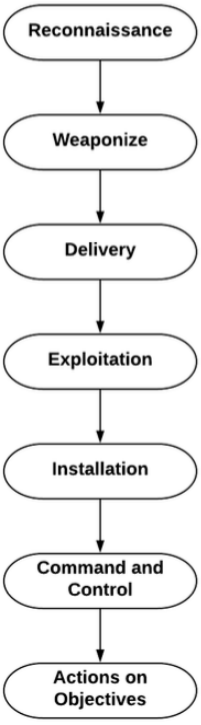


Figure 16. Stages involved in a cyber kill chain model. This figure shows the sequential relationship between the seven stages involved in a kill chain model.

(i) **Step 1 - Reconnaissance:** This is the initial planning phase. It entails collecting information about the potential target. The victim or target in this case could be an individual or an organization. The information collected from this step can be used in the later stages of the Cyber Kill Chain to scheme and deliver the payload (Yadav & Rao, 2015). Reconnaissance comprises two parts; passive reconnaissance which entails collecting data without the knowledge of the victim and active reconnaissance which prompts an alert to the target.

(ii) **Step 2 - Weaponize:** This step involves the preparation and planning to stage the attack. It entails utilizing the information gathered from the first step. It includes the use of malware payload, developing the phishing email campaign, and exploiting kits and botnets (Wrozek, n.d.). Two components were identified to be the major components of

this step; the Remote Access Tool (RAT), a software that can be executed on the computer, mobile or embedded device of the victim and the exploit which aids the execution of the RAT.

(iii) **Step 3 Delivery:** This step entails interacting with the target network in a way that the weaponized payload is transferred (Red, 2016). It is the most essential part of the model as it determines the efficiency and effectiveness of the cyberattack. Several mechanisms used to accomplish this task include email attachments, phishing attacks, drive by download, Universal Serial Bus (USB) removal media and DNS cache poisoning.

(iv) **Step 4 Exploitation:** Upon the completion of step three, the exploit stage is initiated. At this stage, the attacker gains access to the victim through exploited hardware or software vulnerabilities with known patches and in some cases attack or victim-triggered exploits. Scripted code can also be injected into the victim's environment (Mindsight, n.d.).

(v) **Step 5 Installation:** In this step, the attacker has gained grounds in the victim's environment. It includes the installation of backdoor, web shell or webserver. Additional services are included to generate point of persistence like AutoRun keys (Security Boulevard, 2018).

(vi) **Step 6 Command and Control (C2):** This step is a two-way communication medium used for remote control on compromised machines. The hacker has taken full access of the system and can manipulate the channel over web, electronic mail (e-mail) and DNS. These movements are usually done internally (Wrozek, n.d.). There are three major C2 structures - centralized, decentralized and social network-based structures.

(vii) **Step 7 Action and Objectives:** This is the final step in the chain. It is the concluding part where the attacker achieves their major aim, and this usually varies on the type of attack.

In most cases, the goal of the attacker is to steal data, modify, delete or destroy the victim's system or data.

3.5.2. Trike Modelling Tool Simulation

Trike modelling tool was developed for inexperienced security developers to use and discover issues. It gives a brief description of what needs to be analyzed and the point to which it should be analyzed. It uses the Kill Chain model to carry out this analysis. Eddington, Larcom, and Saitta (2005) described this tool as "a conceptual framework for security auditing from a risk management perspective through the generation of threat models in a reliable and repeatable manner" (para. 1). Its high level of automation distinguishes it from the other existing threat models. It includes systems and the intended behavior of the system and uses the requirement model as the basis of its methodology. This tool was modeled based on the Create (C), Read (R), Update (U), Delete (D), Execute (X), and Configured (C). The attributes are described in Table 3.

Table 3

Description of features used in trike modelling. This table describes the attributes and options used in the trike modelling tool.

Name	Attribute	Description
Create	**C**	Create a new object. For example, add a row to a database table, call Object new, or allocate a memory buffer.
Read	**R**	View or use [part of] the contents of an object. For example, select values from a database, open and read a file, or receive network traffic.
Update	**U**	Change [part of] the contents of an object. For example, update values in a database, set a variable, or write to a file.
Delete	**D**	Remove or destroy an object. For example, delete a row from a database table, remove a file, or free memory.
Execute	**X**	Execute an object. For example, run a program, invoke a function in a library, or interpret code in an interpreted language.
Configure	**F**	Configure an object. For example, set file permissions, change header bits in a TCP packet, or set configuration registers on a hardware device.

Modelling the basic web application architecture using the Kill Chain model produced Table 4 on the threat modelling tool. The system was designed with the infusion of both scenarios as the tool created a model based on the architecture of a web application. In the first scenario, an attacker hacked into a webserver with login credentials to update the existing password of one of the users; mitigation techniques were also advised for the SQLi attack. For the second scenario an attacker updated the comment section of the web page with XSS scripts to change the color of the background of the website after which appropriate input in the model produced the output in Table 4 declaring the assets at risk with the appropriate color codes.

Table 4

Trike modelling use case for SQLi and XSS attacks. This table shows the output model from the steps involved in the scenario one and two declaring the assets and resources at risk.

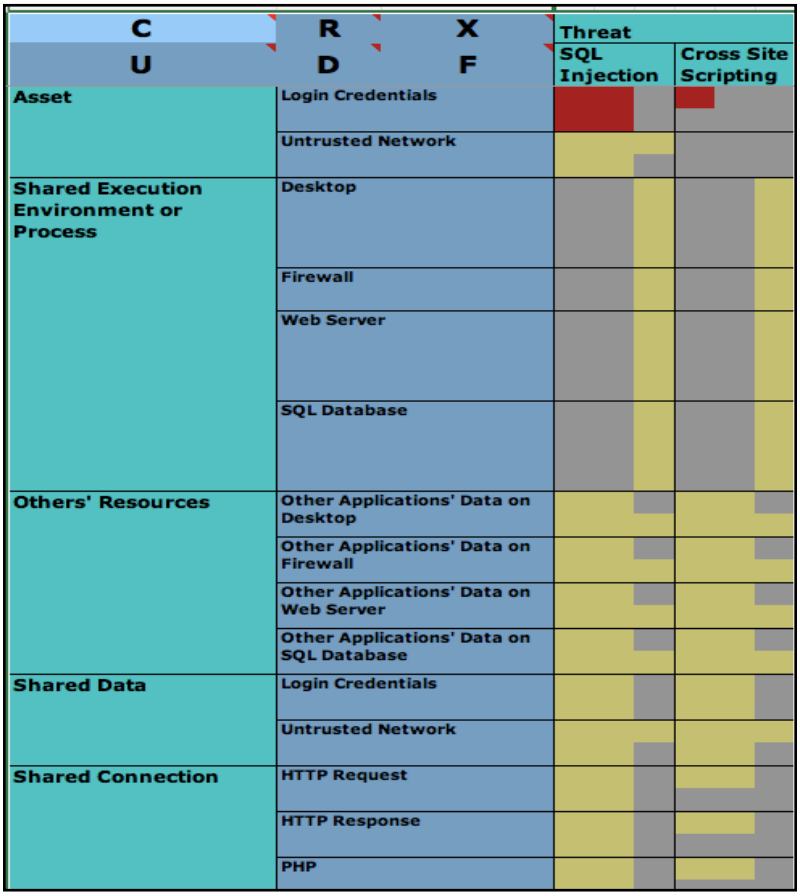


Table 4 outlined the assets, shared execution environment or process, other's resources, shared data and shared connection as the resources at risks for SQLi and XSS attacks. The color codes used are described in Table 5.

Table 5

Color codes used in the trike model. This table describes the use of the color codes in the trike model for SQLi and XSS attacks.

	This action does not apply to this asset, based on the asset's type in the Data Model tab.
	(Never) The system should never let this actor take this action on this asset.
	(Conditionally) The system should let this actor take this action on this asset when certain conditions (typically documented in the cell comment) are met.
	(Always) The system should always let this actor take this action on this asset.

This model analyzed each of the components involved in the system architecture and mapped out a model based on these fundamental structures. Mitigations techniques would be determined by the stakeholders from a risk management approach.

3.6. Attack Tree Model

Scheier (1999) defined Attack Trees as a methodical way of illustrating the security of systems depending on different forms of attacks. Often, Attack Trees are used to look for threats with the use of building blocks. The objective was to iterate over each node in the tree and analyze the impacts of the threats on the system (Stostack, 2014). To aid in this analysis, the following steps are involved in creating an Attack Tree.

(i) Decide on a representation

(ii) Create a root node

(iii) Create sub-nodes

(iv) Consider completeness

(v) Prune the tree

(vi) Check the presentation

3.6.1. Attack Tree Mode of Operation

In deciding the representation, there are 'AND' and 'OR' trees. The 'AND' tree depends on all the nodes for it to be true while for the 'OR' tree, the node is true if any of its sub-nodes are true. The fundamental stage in starting an Attack Tree is the root node which varies from the components used in the analysis to the major goal of the attacker. After the creation of a root node, the next step involves creating a sub-node with adequate communication between the 'AND' or 'OR' on the nodes. Upon creation of the sub-nodes, the completeness of the attack is validated. The next step involves checking for repetitive and preventive actions on each nodes and sub-nodes. The final step ensures a complete work is done on the nodes for a crisp and clear presentation. Trees can be represented in two ways - free form model which is usually easily readable by humans and a structured representation of several types and metadata to assist in the analysis.

Wang, Whitley, and Parish (2010) classified Attack Tree modelling into two different types - Conventional Attack Trees and Augmented Attack Trees. Conventional Attack Trees use the connection types 'OR' and 'AND' to connect numerous child nodes to the same parent node while Augmented Attack Trees are an extension of the Conventional Attack Trees which combines each branch with a series of malicious operations used in the attack. To model an attack, an outline is drawn to achieve the goal in different nodes. SeaMonster threat modelling tool was used to model the nodes and sub-nodes for SQLi and XSS attacks.

3.6.2. SeaMonster Threat Modelling Tool Simulation

SeaMonster is a graphical security modelling tool built on the eclipse framework with the description of various perspectives related to security vulnerabilities. It originated from the work

carried out by a student of Norwegian University of Science and Technology (NTNU) in association with the Applied Research Foundation SINTEC (Meland, 2008). The purpose of the tool was to assist developers and security experts to model security in the different stages of software development. The different perspectives involved in this development go hand in hand, and are listed as follows:

- The major cause of the vulnerability
- Level of threat and attack because of the vulnerability
- Mitigation measures on the effects of the vulnerability

SeaMonster tool was built based on the Attack Tree model with the above viewpoints. Using the first scenario of the SQLi attack where a malicious script was injected into a web application to update and modify the existing information on the database, a Vulnerability Cause Graph (VCG) was created describing the various causes of SQLi vulnerability. VCG are visual representation of vulnerability models that provides an overview of the relationship between the vulnerabilities and their causes for security experts and developers (Meland, 2008). The VCG for SQLi is shown in Figure 17.

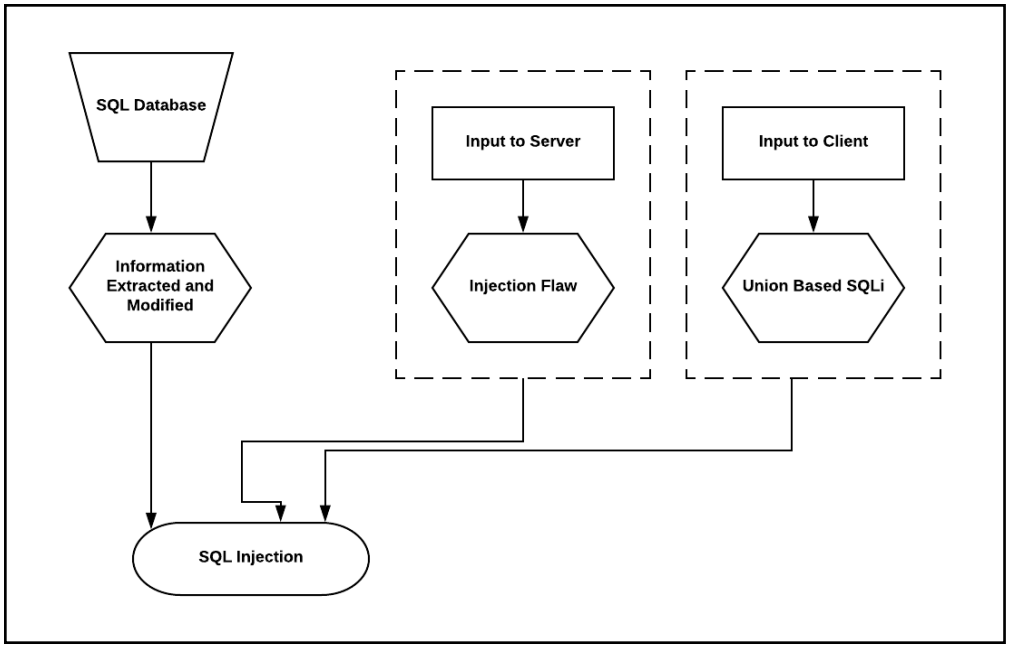


Figure 17. Vulnerability cause graph for SQLi vulnerability. This figure shows the relationship between SQLi vulnerabilities and their causes.

Figure 17 shows the causes and vulnerabilities of a directed acrylic graph for SQLi vulnerabilities with different nodes - simple, compound, conjunction and exit nodes. The first case indicated how information could be extracted from the database through root access to the web server. The second option illustrated a conjunction node around a simple node and a compound node indicating how both nodes are needed to be present. The last option indicated how a user was lured into clicking on a malicious link through which access was granted into the SQL database. The VCG for XSS is shown in Figure 18.

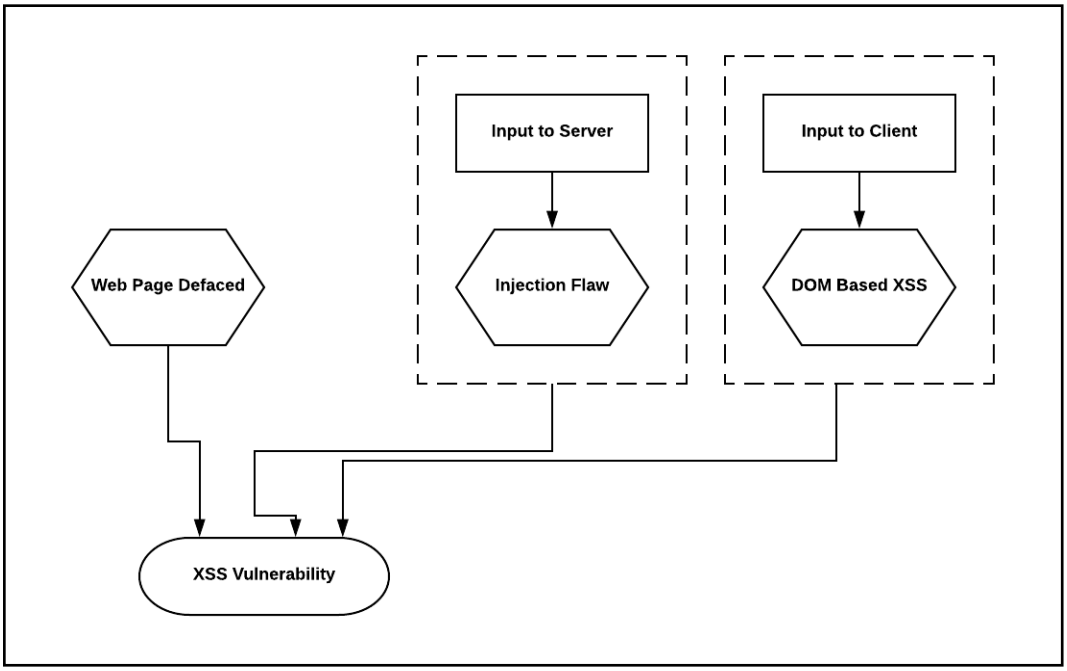


Figure 18. Vulnerability cause graph for XSS vulnerability. This figure shows the relationship between XSS vulnerabilities and their causes.

Figure 18 shows the causes and vulnerabilities of a directed acrylic graph for XSS vulnerability with different nodes - simple, compound, conjunction, and exit nodes. The first case indicated how the web page was defaced through root access to the web server. The second option illustrated a conjunction node around a simple node and a compound node indicating that both nodes needed to be present. The last option indicated how a user was lured into clicking on a malicious link through which access was granted into the webserver. The other perspective of the tool highlighted causes of the injection vulnerabilities of both attacks shown in Figure 19.

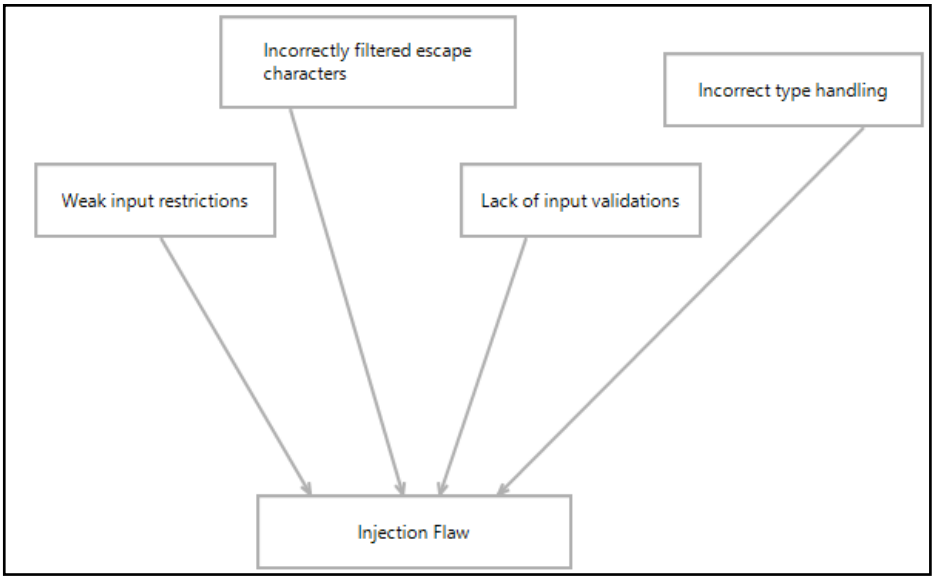


Figure 19. Injection flaw vulnerability. This figure describes the various causes that can result to an injection flaw applicable to both SQLi and XSS attacks.

Weak input restrictions, incorrectly filtered escape statements, lack of input validations, and incorrect type handling are all possible causes that could result into an injection flaw in a web application. Figure 20 and Figure 21 shows XSS and SQLi attacks represented in a tree structure with the steps required to execute the attacks in the nodes and sub-nodes.

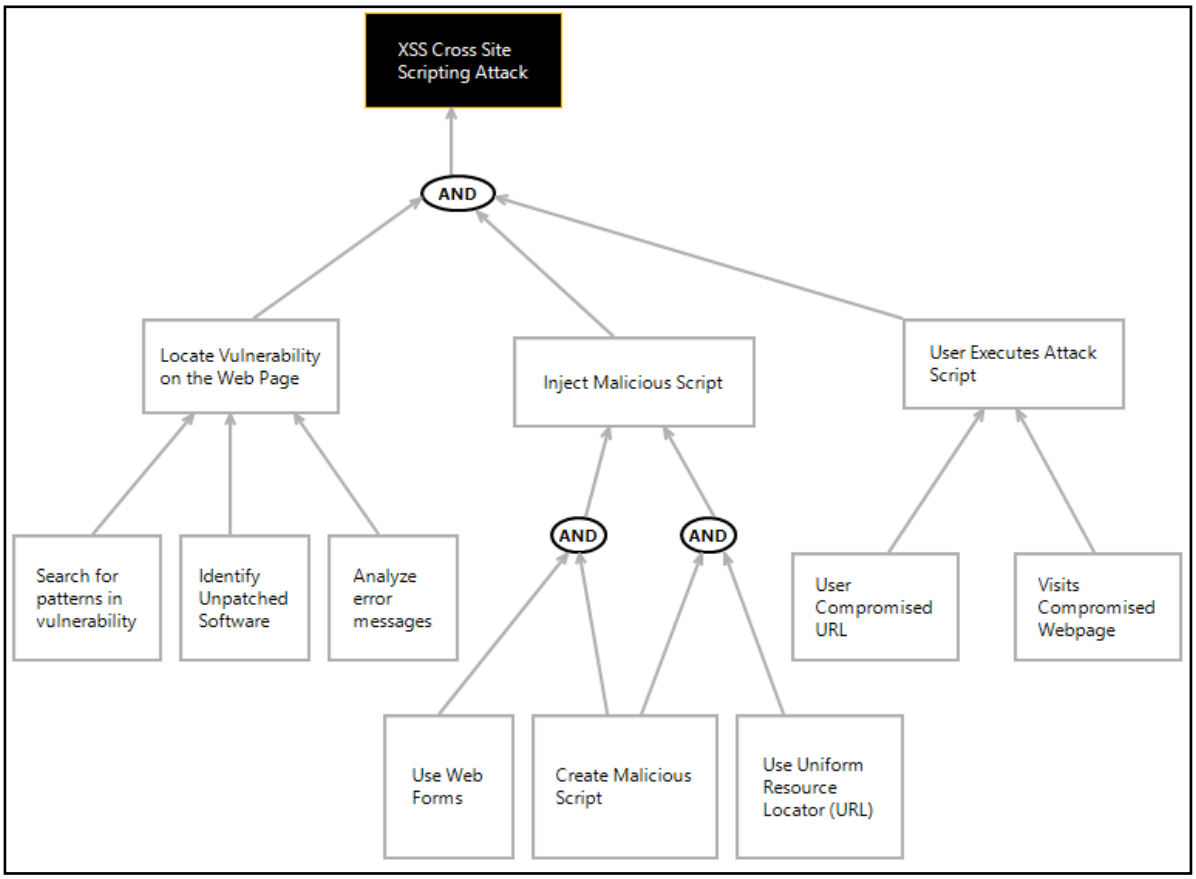


Figure 20. Attack tree model for XSS attack. This figure represents the tree structure for the nodes and sub-nodes for XSS attack.

In Figure 20, the three nodes with fields "*locate vulnerability on the web page*", "*inject malicious scripts*" and "*user executes attack scripts*" worked together to execute XSS attack. Each of the nodes and sub-nodes represented the steps required to execute XSS attack.

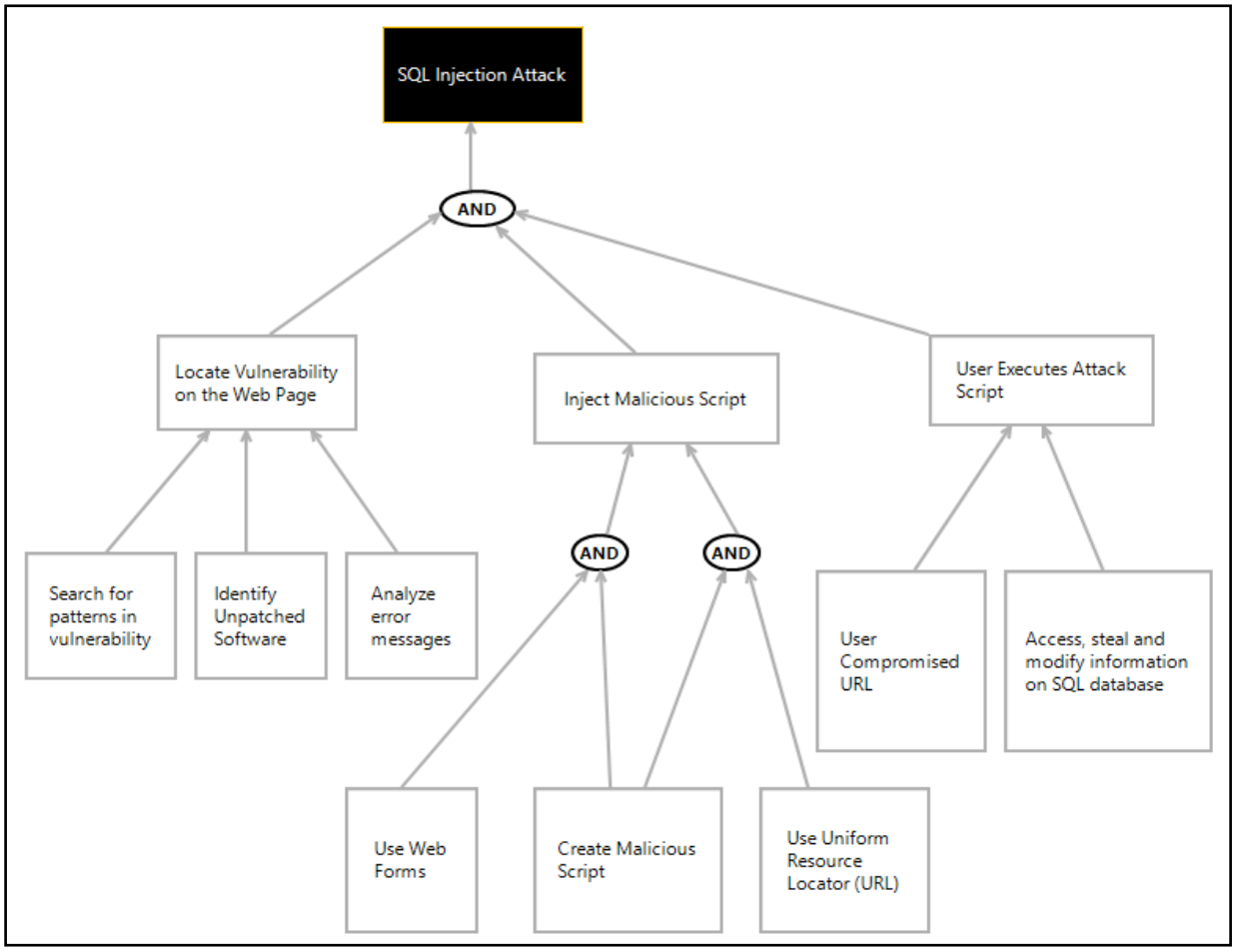


Figure 21. Attack tree model for SQLi attack. This figure represents the tree structure for the nodes and sub-nodes for SQLi attack.

The three nodes with fields "*locate vulnerability on the web page*", "*inject malicious scripts*" and "*user executes attack scripts*" worked in conjunction with each other to execute SQLi attack. Each of the nodes and sub-nodes represented the steps required to execute SQLi attack. The last objective of the tool used a UML case diagram shown in Figure 22 and Figure 23 to address the security issues in SQLi and XSS attacks using four major elements - systems, actors using the system and the functionality of the system with an overview of the countermeasures. The top node contained the vulnerability and the leaf nodes comprised of the activities used in preventing the vulnerability.

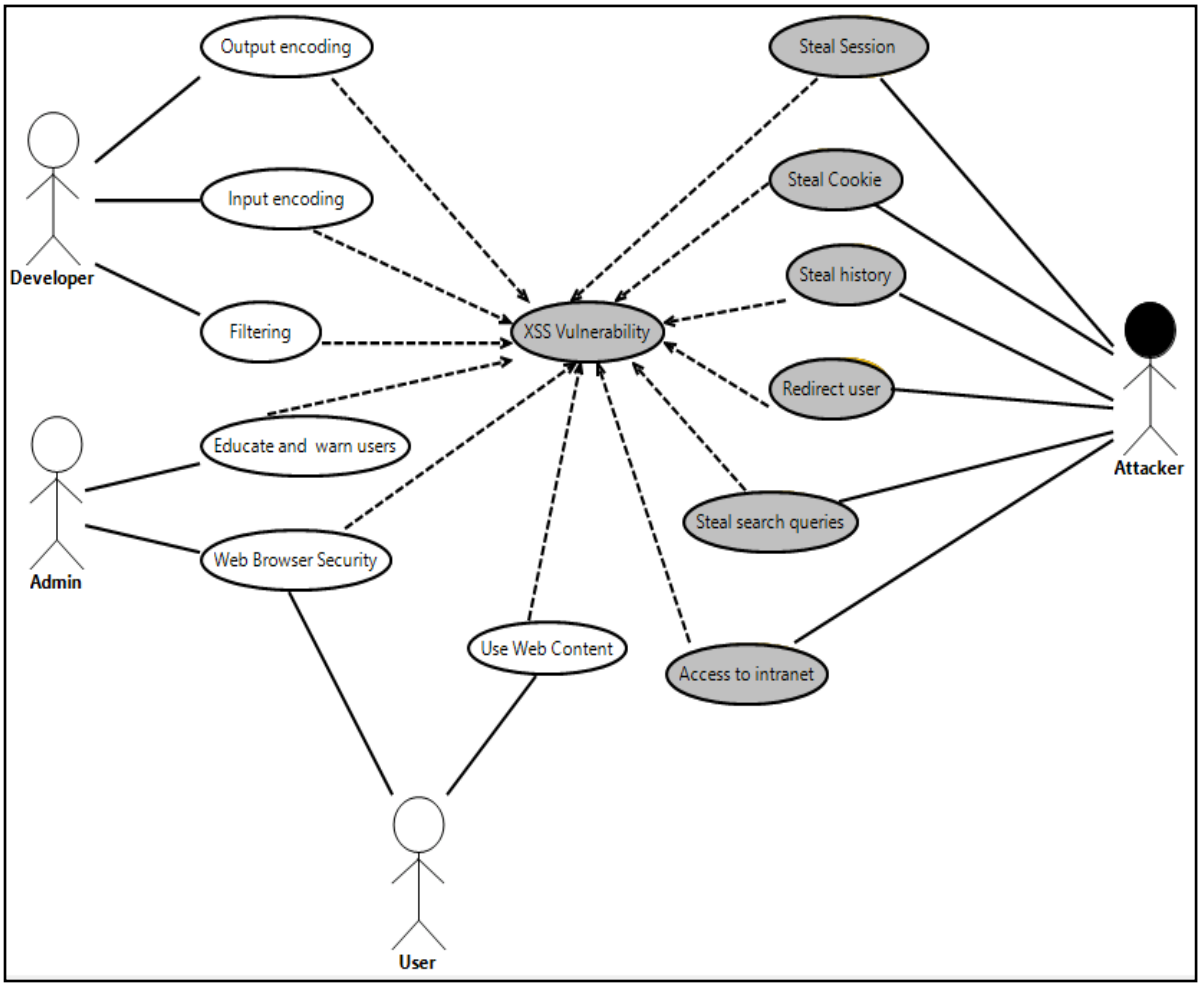


Figure 22. Use case model extended with misuse case notation for XSS vulnerability. This figure shows the mitigation measures and actions to be carried out by depicting the threats and countermeasures of an XSS vulnerability based on the attack tree model developed.

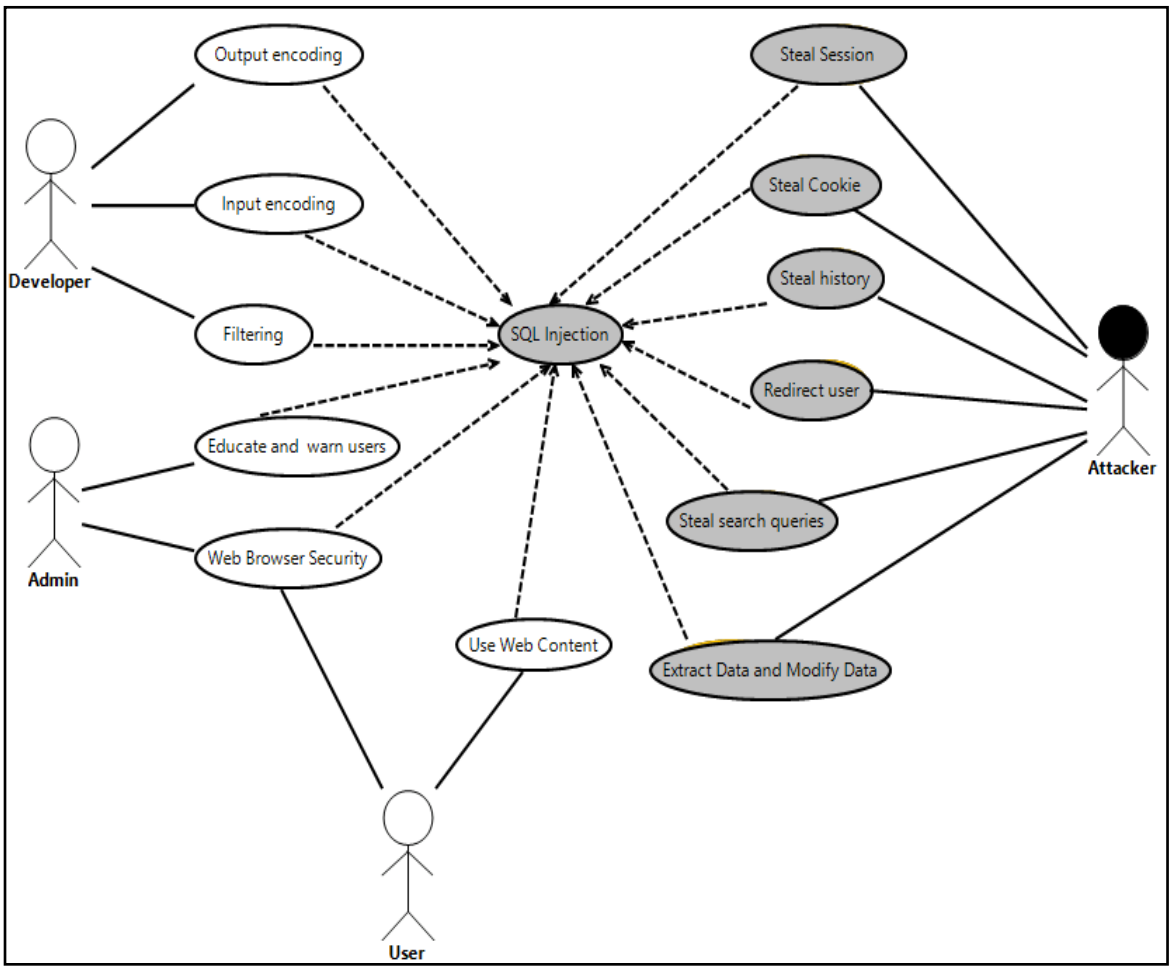


Figure 23. Use case model extended with misuse case notation for SQLi vulnerability. This figure shows the mitigation measures and actions to be carried out by depicting the threats and countermeasures of a SQLi vulnerability based on the attack tree model developed.

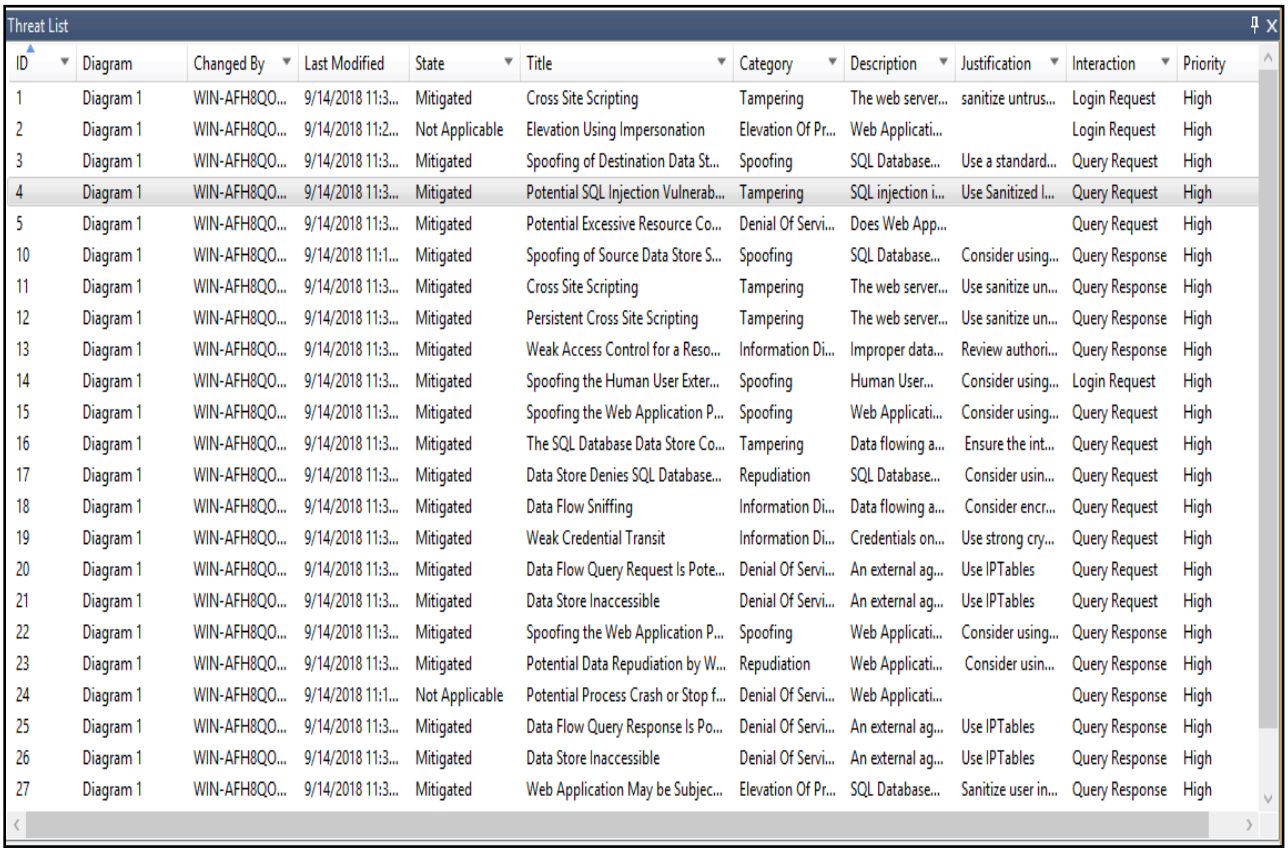
CHAPTER 4: RESULTS AND ANALYSIS

This chapter analyzed the results of the simulations of the STRIDE model, Kill Chain model and the Attack Tree model based on scenario one and two.

4.1. Analysis of the STRIDE Model for Scenario One & Two

The STRIDE model which is, an acronym for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege, was built to model the flow of data through systems and subsystems. It took into consideration the checklist for the STRIDE categories and designed its methodology based on this approach. Hence, in modelling the web application attacks the model considered the underlying web application architecture for both scenarios since both attacks are executed on a web application. Designing the model based on the web architecture produced a model described in the previous chapter and resulted in the analysis in figure 12. Using the SDL threat modelling tool to design this model, the analysis showed the checklist-based approach where each of the six categories in the acronyms for STRIDE model were adequately utilized.

The analysis in the input model shown in Figure 24 indicated that the model moved through each step used in creating the web application architecture highlighting the possible threats applicable to each of the steps with room for mitigation techniques to prevent the occurrence of the attacks.



Threat List

ID	Diagram	Changed By	Last Modified	State	Title	Category	Description	Justification	Interaction	Priority
1	Diagram 1	WIN-AFH8QO...	9/14/2018 11:3...	Mitigated	Cross Site Scripting	Tampering	The web server...	sanitize untrus...	Login Request	High
2	Diagram 1	WIN-AFH8QO...	9/14/2018 11:2...	Not Applicable	Elevation Using Impersonation	Elevation Of Pr...	Web Applicati...		Login Request	High
3	Diagram 1	WIN-AFH8QO...	9/14/2018 11:3...	Mitigated	Spoofing of Destination Data St...	Spoofing	SQL Database...	Use a standard...	Query Request	High
4	Diagram 1	WIN-AFH8QO...	9/14/2018 11:3...	Mitigated	Potential SQL Injection Vulnerab...	Tampering	SQL injection i...	Use Sanitized I...	Query Request	High
5	Diagram 1	WIN-AFH8QO...	9/14/2018 11:3...	Mitigated	Potential Excessive Resource Co...	Denial Of Servi...	Does Web App...		Query Request	High
10	Diagram 1	WIN-AFH8QO...	9/14/2018 11:1...	Mitigated	Spoofing of Source Data Store S...	Spoofing	SQL Database...	Consider using...	Query Response	High
11	Diagram 1	WIN-AFH8QO...	9/14/2018 11:3...	Mitigated	Cross Site Scripting	Tampering	The web server...	Use sanitize un...	Query Response	High
12	Diagram 1	WIN-AFH8QO...	9/14/2018 11:3...	Mitigated	Persistent Cross Site Scripting	Tampering	The web server...	Use sanitize un...	Query Response	High
13	Diagram 1	WIN-AFH8QO...	9/14/2018 11:3...	Mitigated	Weak Access Control for a Reso...	Information Di...	Improper data...	Review authori...	Query Response	High
14	Diagram 1	WIN-AFH8QO...	9/14/2018 11:3...	Mitigated	Spoofing the Human User Exter...	Spoofing	Human User...	Consider using...	Login Request	High
15	Diagram 1	WIN-AFH8QO...	9/14/2018 11:3...	Mitigated	Spoofing the Web Application P...	Spoofing	Web Applicati...	Consider using...	Query Request	High
16	Diagram 1	WIN-AFH8QO...	9/14/2018 11:3...	Mitigated	The SQL Database Data Store Co...	Tampering	Data flowing a...	Ensure the int...	Query Request	High
17	Diagram 1	WIN-AFH8QO...	9/14/2018 11:3...	Mitigated	Data Store Denies SQL Database...	Repudiation	SQL Database...	Consider usin...	Query Request	High
18	Diagram 1	WIN-AFH8QO...	9/14/2018 11:3...	Mitigated	Data Flow Sniffing	Information Di...	Data flowing a...	Consider encr...	Query Request	High
19	Diagram 1	WIN-AFH8QO...	9/14/2018 11:3...	Mitigated	Weak Credential Transit	Information Di...	Credentials on...	Use strong cry...	Query Request	High
20	Diagram 1	WIN-AFH8QO...	9/14/2018 11:3...	Mitigated	Data Flow Query Request Is Pote...	Denial Of Servi...	An external ag...	Use IPTables	Query Request	High
21	Diagram 1	WIN-AFH8QO...	9/14/2018 11:3...	Mitigated	Data Store Inaccessible	Denial Of Servi...	An external ag...	Use IPTables	Query Request	High
22	Diagram 1	WIN-AFH8QO...	9/14/2018 11:3...	Mitigated	Spoofing the Web Application P...	Spoofing	Web Applicati...	Consider using...	Query Response	High
23	Diagram 1	WIN-AFH8QO...	9/14/2018 11:3...	Mitigated	Potential Data Repudiation by W...	Repudiation	Web Applicati...	Consider usin...	Query Response	High
24	Diagram 1	WIN-AFH8QO...	9/14/2018 11:1...	Not Applicable	Potential Process Crash or Stop f...	Denial Of Servi...	Web Applicati...		Query Response	High
25	Diagram 1	WIN-AFH8QO...	9/14/2018 11:3...	Mitigated	Data Flow Query Response Is Po...	Denial Of Servi...	An external ag...	Use IPTables	Query Response	High
26	Diagram 1	WIN-AFH8QO...	9/14/2018 11:3...	Mitigated	Data Store Inaccessible	Denial Of Servi...	An external ag...	Use IPTables	Query Response	High
27	Diagram 1	WIN-AFH8QO...	9/14/2018 11:3...	Mitigated	Web Application May be Subjec...	Elevation Of Pr...	SQL Database...	Sanitize user in...	Query Response	High

Figure 24. Analysis of the threat model generated using the SDL threat modelling tool. This figure shows the output analysis for the web application architecture based on STRIDE model for SQLi and XSS attacks.

In Figure 24, the first part of the process which involved the *Login Request Interaction* on ID 1 and 2 outlined "*Cross Site Scripting Attack and Elevation Using Impersonation*" as the potential attacks applicable to that step. In addition, the third and fourth ID "*Spoofing of Destination Data Structure and Potential SQL Injection Vulnerabilities"* were identified as the potential attacks applicable to the *Query Request Interaction*. It is the responsibility of the system administrator or the security analyst to determine the priority of the potential attacks and apply mitigation measures at the necessary steps in the architecture to avoid danger. In Figure 24, the mitigation measure for *ID 1* with *Title Cross Site Scripting* on the *Login Request Interaction* is to *sanitize untrusted input parameters* which would be done by the security

analyst. Several mitigation techniques were applied all through the web application to avoid any form of attack. The model was used as a state of practice after navigating through each of the steps and mitigation measures applied to the points necessary. This model is an ideal approach for organizations that do not have a lot of prowess in security because the checklist-based approach limits users and reduces the possibility of false positives. Hence, the average number of incorrect threats was low while the average number of overlooked threats was high.

4.2. Analysis of the Kill Chain Model for Scenario One & Two

The seven steps listed in the Kill Chain model was used in this study. The steps include reconnaissance, weaponization, payload delivery, exploitation, installation, command and control, actions on objectives. There was a comparison of these steps with the procedures involved in both SQLi and XSS attack. Hence, both attacks were considered as one since they followed the same process. Using the Trike modelling tool, the Create, Read, Update, Delete, Execute and Configure correlated to the seven steps involved in the Kill Chain model. The Create option (C) on the Trike modelling tool helped to find and create the victim for a malware attack while the Read (R) option completed the creation of the malware and built the weapon. The Update (U) options tracked the malware created by sending the weaponized bundle to the target web application. The Delete (D) option completed the exploitation of the system through the malware code and the Execute (E) option delivered the payload to the systems, commanded and communicated the APT code to the target victim. Finally, the Configure (F) option completed the sole objective of the attacker through the required configurations. The Kill Chain model broke down each stage of an attack enabling the security analyst or administrator to identify and halt the process.

Using the steps to analyze the model with the trike modelling tool combined both scenarios for the web application attacks to produce the following steps:

Step 1: The attacker scanned the network with a vulnerability scanning device to discover the victim alongside the open ports that were vulnerable and exploitable.

Step 2: After discovering the IP address and open ports, the malicious user attempted to gain access by entering malicious scripts in the accessible area of the web application often through the login page of the web page or through the comments section.

Step 3: Upon successful login attempt or post, the attacker used the information to modify, update or delete the existing information on the database or completely deface the web page to make it unavailable to the end users.

The Trike modelling tool used the options involved in the Kill Chain model and considered the different resources that make up the web application. The assets include the login credentials and the untrusted network, the shared execution processes or environment comprised of the desktop, firewall, web server and SQL database, all other resources, data, and shared connections. The tool used color codes to provide information on the necessary areas for the security analyst to act upon. It used the risk-based approach which specified the risks the system entailed to all the stakeholders and ensured that it was acceptable. It also communicated the effects of the risks to the stakeholder for mitigation plans. The red code indicated the locations where the malicious user gained access using malicious codes as it was the point through which a connection was established with the backend database. The yellow code indicated that an action could be taken by the user provided mitigation measures were appropriately carried out. The output depicted on the Trike model were based on steps from the attributes - Create (C),

Read(R), Execute (E), Update (U), Delete (D) and Configure (F). The highest resource at risk was the asset for login credentials because once the login platform is compromised, the color code for the other resources would turn red and become susceptible to SQLi and XSS attacks. This model allowed stakeholders identify the areas that needed to be strengthened. The level of criticality for the resources are shown in Table 6.

Table 6

Level of criticality for the resources in the trike modelling tool for SQLi and XSS attacks. This table shows resources at risk in a web application for SQLi and XSS attacks based on the level of criticality. The symbol 'X' is used to signify the critical and less critical resources.

Resources at Risk	Components	Most Critical	Less Critical
Assets	Login Credentials	X	
	Untrusted Network		X
Shared Execution Environment or Process	Desktop		X
	Firewall		X
	Web Server		X
	SQL Database		X
Others' Resources	Other Application's Data on Desktop		X
	Other Application's Data on firewall		X
	Other Application's Data on webserver		X
	Other Application's Data on SQL database		X
Shared Data	Login Credentials		X
	Untrusted Network		X
Shared Connection	HTTP Request		X
	HTTP Response		X
	PHP		X

From Table 6, the login credentials were the most critical asset at risk while the untrusted network was less critical. The resources for shared execution environment or process which comprised of the desktop, firewall, web server and SQL database were identified as less critical. For others' resources, the data associated with the shared execution process were all identified as less critical. The shared data in the shared environment were also identified as less critical. The

shared connection which comprised of HTTP request, HTTP response and PHP scripting were also identified as less critical for the shared service. Hence for a secured web application, the login credentials must be appropriately defended before mitigating the other shared platforms.

4.3. Analysis of the Attack Tree Model

The Attack Tree model mapped out the several ways an attacker was able to intrude the network. Security analysts and system administrators use this model to analyze the weaknesses in a system and make decisions on which security measure is the most effective to deploy. The SeaMonster Modelling tool modelled the Attack Tree for each scenario individually based on the following objectives - the major cause of the vulnerability, level of threat and attack as a result of the vulnerability as well as mitigation measures on the effects of the vulnerability.

4.3.1. Analysis of the Attack Tree Model for Scenario One

Based on the steps highlighted in scenario one, a VCG was designed which described the different causes that can lead to an SQLi vulnerability. From the model, it can be deduced that there could be three possible causes. The first cause was through gaining root access to the webserver by inputting malicious scripts according to the steps in scenario one to update the information on the database. The second cause considered the simple and compound node i.e. the conditions that resulted to a vulnerability and the facilitation of the reuse. Although, the conditions usually stemmed from the client or server, however, based on this study, it came through the client where a malicious code was injected into the login area located in the public area of the website. However, in other conditions that originate from the server the user is tricked to click on a maliciously crafted link that redirects the user to another webserver which would request for login credentials for exploitation purposes.

After designing the VCG, the SeaMonster tool designed an Attack Tree based on the steps in scenario one. It broke down each of the activities required to execute the attack with the use of the 'AND' gate. There were three nodes in total. The first node scanned for the vulnerabilities and it was synonymous to the first step in scenario one, the node also comprised of three sub-nodes. The three sub-nodes comprised of major activities involved in locating the vulnerabilities which involved the vulnerability search in patterns, identifying unpatched software and analyzing error messages. The second node which injected malicious scripts to the web application was also synonymous to the step two and three in scenario one, the second node comprised of three sub-nodes. The three sub-nodes comprised of the use of web forms, the creation of the malicious scripts and the use of the URL. In the malicious scripts injection node, the sub-node for the use of web forms and that of the creation of malicious scripts must work in conjunction with one another. Likewise, sub-nodes for the creation of the malicious script also must work in conjunction with the use of URL. Finally, the third node which involved how the user executes the attack scripts was synonymous with step four, five and six and it comprised of two sub-nodes - user compromised URL, access, steal and modify information on SQL database. The three nodes worked in conjunction with each other to execute the SQLi attack. All these steps were executed sequentially. With the model in place, security analyst and administrators can determine the best mitigation technique required in the various levels in the model in the event of an attack.

Upon the completion of the Attack Tree, a use case model was created for developers, security administrators and end users to highlight the countermeasures from the steps taken by the attacker. For each of the steps highlighted in scenario one, countermeasures were outlined. In the use case model, the major actions taken by the attacker include extracting and modifying

data, stealing search queries, redirecting users to another malicious URL, stealing browser history, cookie and session. While on the other end for the developers, security administrators and developers, the mitigation measures to be taken include encoding the output and input statements, filtering and sanitizing statements, educating the end users on best practices and securing the web browser through Transport Layer Security/ Secure Sockets Layer (TLS/SSL). In synopsis, the Attack Tree model analyzed in detail the steps used in executing the SQLi attack and used these steps to mitigate the attack.

4.6.2. Analysis of the Attack Tree Model for Scenario Two

Based on the steps highlighted in scenario 2, a VCG was designed to describe the various causes that could result to an XSS vulnerability. From the graph, it was deduced that there were three possible causes. The first cause was through gaining root access to the webserver by inputting malicious scripts according to the steps in scenario two to replace the web page content to deface the web page. The second cause considered the simple and compound node i.e. the conditions that resulted into a vulnerability. The various conditions stemmed from the client or server, if the vulnerability came through the client, malicious scripts would be injected into the comments section area of the web page. However, with other conditions that originated from the server, the user is tricked to click on a maliciously crafted link that redirects the user to another webserver which would request for login credentials for exploitation purposes. The VCG provided a good overview of the relationship between the vulnerabilities and its causes.

Upon the completion of the VCG, the SeaMonster tool represented the steps in scenario two in a tree structure using the Attack Tree model. It decomposed each of the activities required to execute the attack with the use of the ‘AND’ gate. There were three nodes in total. The first node which scanned for the vulnerabilities was synonymous to the first step in scenario one, the

node also comprised of three sub-nodes. The three sub-nodes comprised of major activities involved in locating the vulnerabilities which involved the vulnerability search in patterns, identifying unpatched software and analyzing error messages. The second node which injected malicious scripts to the web application was also synonymous to the step two and three in scenario one, the second node comprised of three sub-nodes. The three sub-nodes comprised of the use of web forms, the creation of the malicious scripts and the use of the URL. In the malicious scripts injection node, the sub-node for the use of web forms and that of the creation of malicious scripts worked in conjunction with one another. Likewise, sub-nodes for the creation of the malicious script also worked in conjunction with the use of URL. Finally, the third node which involved how the user executes the attack scripts was synonymous with step four. The node also comprised of two sub-nodes - user compromised URL, compromised web page. The three nodes worked in conjunction with each other to execute the XSS attack. With the model in place, security analyst and administrators would be able to determine the best mitigation technique required in the various levels in the model.

After the design of the Attack Tree for the XSS attack, a use case model was developed for the developers, security administrators and end users highlighting the countermeasures for the actions from the steps taken by the attacker. For each of the steps highlighted in the scenario, countermeasures were outlined. In the use case model, the major actions for the attacker include illegal access to the intranet, stealing search queries, redirecting users to another malicious URL, stealing browser history, cookie and session. On the other end, for the developers, security administrators and developers the mitigation measures include encoding the output and input statements, filtering and sanitizing statements, educating the end users on best practices and securing the web browser through TLS/SSL. In synopsis, the Attack Tree model analyzed in

detail the steps used in executing the XSS attack and used these steps to mitigate the attack as seen in the use case model.

CHAPTER 5: DISCUSSION

The analysis of STRIDE model, Kill Chain model and Attack Tree on SQLi and XSS attack using the Microsoft SDL threat modelling tool, Trike modelling tool and SeaMonster modelling tool were useful as each of the models presented different approaches in different ways. Hence, the following evaluations were drawn from the study in terms of the strengths and the weaknesses of each of the models.

5.1. Evaluation of the STRIDE Threat Model Using the Microsoft SDL Tool

The evaluation of the STRIDE threat model based on the scenario one and two which is the SQLi attack and the XSS attack, prove to be a very effective approach as it served as a proactive measure to what could possibly go wrong on the web application architecture. It highlighted the different possible attacks prone to the web application and provided an avenue for security and system administrators to map out measures to prevent the attacks. The STRIDE model could serve as a starting point for the inexpert members of a threat modelling team. With the knowledge of the different categories and the potential attacks highlighted in the model, it would instill the knowledge to the inexperienced threat modelers of the dangers of exposing specific technical information that could be used by the attacker in gaining insight as to where vulnerabilities are present in their web applications enabling them to see the web applications through the eyes of an attacker. This model also served as a checklist that could be used by both the experienced and the inexperienced members of the security team using the list of threats provided and it is easier to apply the mitigation measures. The STRIDE model helped to track assumptions and threats before it consolidated into an Attack Tree. A major advantage of this model is that it decomposed the system into logical components. Each model was analyzed using

the DFD to visualize the functionalities within and outside the system. For the web application, each of the components that make up the architecture were broken down into the STRIDE per interaction to indicate the threats between the system components.

However, there were some shortcomings noticed in the design of the STRIDE threat model for both scenarios. The STRIDE was more generic to the entire application as it was not specific to the main attacks in question. Because it cuts across all the different forms of attacks prone to web applications, certain vulnerabilities that would require in-depth analysis might be overlooked. In addition, the STRIDE model does not require the steps required to execute an attack. All the attacks displayed in the output have been predetermined and in the event of a new and updated form of attack prone to the application, it would not be captured in this model. The amount of effort required to check the six categories highlighted in the model could be laborious. Some of these threats outlined could go undetected during a STRIDE model analysis due to the number of threats to be analyzed. Time spent on designing this model was also a major concern. For the checklist of the six categories of threats, a lot of time was spent on adequately analyzing each of the threats and their respective mitigation techniques. After intense analysis of the STRIDE model, it was observed that the model required an expertise knowledge in mitigating the threats identified since the countermeasures are manually inputted. The security administrator should be skilled in applying the mitigation techniques to strengthen the security of the web application. In addition, the model required manual input of the priority of the threats and handling of risks.

5.2. Evaluation of the Kill Chain Model Using the Trike Modelling Tool

Analyzing the Kill Chain Model using the Trike modelling tool for both scenarios using the steps highlighted in the methodology created a risk management model with a high

automation level. It highlighted assets in the web application that needed to be protected at a reasonable level and with the use of the color codes, it would help the stakeholders in an organization understand the risks and map out measures on how to mitigate the risks. The Trike model automated the steps required in both scenarios to produce the assets at risk. With the Kill Chain model in place in web applications, it is easy to pinpoint the indicators that would in turn activate alerts to the security administrators and analysts on the assets at risks. It also assisted in preparing a determined defense through a proper understanding of an adversary. Through proper analysis of the steps and risks declared in the model, it was easy to think with the strategic motive of being ahead of the attackers and crafting out defensive measures.

In this study however, the use of the techniques Kill Chain model was not appropriately utilized by the Trike modelling tool. The tool only highlighted the assets at risks and the level of risks for each point of entry, but it did not highlight every step required in the Kill Chain methodology. It was able to successfully identify the assets and resources that needed to be appropriately defended by the security team. Considering the major steps in the Kill Chain model highlighted in this research, it would be more insightful to upgrade the model because the current model does not place emphasis on what would be done after an attacker successfully gained access into a web application. Hence, another feature could be added to the recovery phase after the successful execution of an attack. Due to the resilience of the cyber attackers, it is not considerably enough to only map out the steps in model required in executing an attack. It would be better to prepare a good defense, keeping in mind the stages of pre-occurrence and post-occurrence. The Kill Chain model is not a very effective approach for insiders and social engineering attack. For example, in cases where a malicious user sends a crafted link to redirect users to his own website and the users gets compromised through this process, it would be a

daunting task capturing all these events in this model. The model is malware-focused and emphasizes on the irrelevance of the reconnaissance, weaponization and delivery from an operational perspective. It can be viewed as a documentation of steps taken by the attacker at the back-end where security professionals have little or no influence over it.

5.3. Evaluation of the Attack Tree Model Using the SeaMonster Modelling Tool

The Attack Tree model created a detailed analysis with the SeaMonster modelling tool as it cut across all the necessary areas in the security systems. It modelled the vulnerabilities of the system through the VCG before designing the Attack Tree that also modelled every step in the process of executing the SQLi attack (scenario one) and XSS attack (scenario two). The Attack Tree model designed and reproduced a use case model that comprised of mitigation techniques for security administrators, analysts and developers for every step outlined in both scenarios. The model was visual and flexible with direct communication with the technical audience. It provided an in-depth and holistic protection of the relationships between the nodes and sub-nodes.

In this study, the Attack Tree model was very effective with meaningful recommendations for security measures. With its in-depth analysis, it resulted into a quick response time for real time attack mitigation which is a very crucial aspect in the field of security. This model is quite easy to develop as the major part of the design would be carried out by the security analyst. Hence it would be easy to identify errors in the event of its occurrence. Also, there was a smooth transitioning from the VCG to the Attack Tree model to the use case model. Retracing the occurrence of an attack in a system would be relatively easy to implement with this model. The security administrator would be able to address the security problems through the vulnerabilities highlighted and steps outlined in both scenarios. Based on the

analysis, Attack Tree model could be seen as a multiple path model as it carved out multiple ways an attack could occur in an infrastructure which was demonstrated in both SQLi and XSS attack. This model also helped the network security administrators prioritize on the most critical threats in an organization and determine the most effective countermeasures. The major advantages of the Attack Tree model include scalability and its effective qualitative approach from the risk management perspective.

However, the level of details in the Attack Tree model could be accompanied with an increase in cost especially in large systems. Another limitation noticed in the design of this model is the manual input required to build up this model. Human errors are unavoidable; while the possible attacks are being highlighted in the make-up of the model, some attacks might be unrepresented which could be prevalent in the system in some cases. It would be more insightful to have additional option on the SeaMonster modelling tool to be able to properly model the Attack Tree and perhaps automate a larger percentage of the steps required to execute the attack in scenario one and two instead of carrying out manually. The generation of an Attack Tree model could be a bit challenging with the number of nodes and sub-nodes involved. In some cases, it might be tardy to identify a valid attack threat. Although, some researchers have invested several algorithms to ease off the burden. Examples of these algorithms are the Monte Carlo algorithm, Breadth Fast and Depth Fast algorithms. The manual input required in designing the Attack Tree could be time consuming and stressful for the security administrator or developer in charge.

CHAPTER 6: CONCLUSION AND FUTURE WORK

In this study three attack modelling techniques were examined and analyzed through the SQLi attack and the XSS attack. For both scenarios, steps required to execute the attack were outlined and conducted in a virtual environment with proper documentation of the processes involved. It demonstrated that the web application attacks can be modelled using different techniques. Each of the threat models used gave different insights about the web application architecture and the steps required from different perspectives. For example, the STRIDE model focused solely on the architecture rather than modelling the steps involved, the Kill Chain model created its assets and resources from a risk management perspective and the Attack Tree built a tree structure through the steps of each attacks in both scenarios before determining its countermeasures. Each of these threat models were unique in their own different ways as they highlighted their mitigation techniques from different angles. Evaluating each of the threat models based on accuracy, completeness and clarity, the STRIDE seemed the most accurate as it provided options for various forms of attacks feasible in a web application. The STRIDE model is also complete according to the capabilities of the model as it required the analysis of the six categories. The Attack Tree model was also complete according to the requirements of the model and easy to understand in terms of clarity as most of the tasks of designing the model were put in place by the security administrators and developers. Hence, it was easy to navigate through the process to identify the vulnerabilities. Although, the Kill Chain model was developed from the risk management perspective it also seemed complete as it assessed the entire application before declaring the assets at risk.

Based on the analysis done in this study, the Attack Tree model was identified as the model with the best framework because of its in-depth analysis. It was also identified as the best

approach as it handled each attack in a separate manner and optimized each section appropriately. The three threat models with the appropriate models in place resulted in the elimination and mitigation of SQLi and XSS attack. Although the study compared the three commonly used threat modelling techniques, more work still needs to be done.

For future research, a hybrid of two of these threat models - STRIDE and Attack Tree model leveraging on the limitations identified should be conducted and an evaluation algorithm should be created to analyze the effectiveness of the model based on the existing threat models. A hybrid threat model with a structured approach and optimum detail would result into a productive and comprehensive threat modelling. The data would be represented in the best way for completeness, accuracy and clarity. The implementation of the hybrid threat model would combine the STRIDE model and Attack Tree model to improve the security posture of web applications taking into consideration the strengths and weaknesses of both models. The evaluation algorithm was developed to assess the performance of the model and automate the major steps required to build the model.

Considering the major steps in the Kill Chain model highlighted in this study, it would be insightful to upgrade the existing model with additional features on what would be done after an attacker successfully breaks into a web application. This new Kill Chain model would include the recovery phase with a structured plan to report incidents and events with prepared defense to boost the cybersecurity posture of a web application.

REFERENCES

Abraham, S., & Nair, S. (2015). A Predictive Framework for Cyber Security Analytics Using Attack Graphs. *International Journal of Computer Networks & Communications (ICNC)* Vol. 7, No. 1. Retrieved from: arXiv:1502.01240v1

Acunetix (n.d.). Cross Site Scripting Attack. Retrieved from: https://www.acunetix.com/websitesecurity/cross-site-scripting/

Al-Mohannadi, H., Mirza, Q., Namanya, A., Awan, I., Cullen, A. & Disso, J. (2016). Cyber-Attack Modeling Analysis Techniques: An Overview. *IEEE 4th International Conference on Future Internet of Things and Cloud Workshops.*

Alderman, R. (n.d.). Shrinking the Kill Chain. Military Embedded Systems. Retrieved from: http://mil-embedded.com/guest-blogs/shrinking-the-kill-chain/

Armeding, T. (2016) The 17 Biggest Data Breaches of the 21st Century. CSO. Retrieved from: https://www.csoonline.com/article/2130877/data-breach/the-biggest-data-breaches-of-the-21st-century.html

Axelrad, E.T., Sticha, P.J., Brdiczka, O., & Shen, J. (2013). A Bayesian Network Model for Predicting Insider Threats. *IEEE Security and Privacy Workshops*. Retrieved from: https://www.ieee-security.org/TC/SPW2013/papers/data/5017a082.pdf

Baadshaug, E., T., Erdogan, G., Meland, P., H. (2010). Security Modeling and Tool Support Advantages. IEEE International Conference on Availability, Reliability and Security. Retrieved from: http://doi.ieeecomputersociety.org/10.1109/ARES.2010.11

Bay, K. (n.d.). Fixating on the Kill Chain Model is Misleading. InfoSecurity. Retrieved from: https://www.infosecurity-magazine.com/opinions/fixating-on-the-kill-chain-model/

Bertino, E., Bruschi, D., Franzoni, S., Nai-Fovino, I. & Valtolina, S. (2005). Threat Modelling for SQL Servers. *IFIP Conference on Communications and Multimedia Security*. Retrieved from: https://www.cerias.purdue.edu/assets/pdf/bibtex_archive/p4a4.pdf

BlackHat USA (2016). Using an Expanded Cyber Kill Chain Model to Increase Attack Resiliency. FusionX. Retrieved from: https://www.blackhat.com/docs/us-16/materials/us 16-Malone-Using-An-Expanded-Cyber-Kill-Chain-Model-To-Increase-Attack Resiliency.pdf

Caltagirone, S., Pendergast, A. & Betz, C. (2013). The Diamond Model of Intrusion Analysis. Retrieved from: http://www.activeresponse.org/wp-content/uploads/2013/07/ diamond.pdf

Davis, S. (2015). Threat Modeling with STRIDE. Webtrends. Retrieved from: https://www.webtrends.com/blog/2015/04/threat-modeling-with-stride/

De Cock, D., Wouters, K., Schellekens, D., Singelee, D., Preneel, B. (2005). Threat Modelling for Security Tokens in Web Applications. *IFIP Conference on Communications and Multimedia Security*.

Death, D. (2018). The Cyber Kill Chain Explained. Forbes. Retrieved from: https://www.forbes.com/sites/forbestechcouncil/2018/10/05/the-cyber-kill-chain -explained/#6114e8316bdf]

Dekker, M. (2014). Using Attack Trees in Cybersecurity for Threat and Risk Modeling. LinkedIn. Retrieved from: https://www.linkedin.com/pulse/20140529230342-18705719 -using-attack-trees-in-cybersecurity-for-threat-and-risk-modeling/

Desmet, L., Jacobs, B., Piessens, F., & Joosen, W. (2005). Threat Modelling for Web Services Based Web Applications. *8th IFIP TC-6 TC-11 Conference on Communications and Multimedia Security pp 131-144 Volume 175.*

Dionach (2016). The Real Impact of Cross Site Scripting. Retrieved from: https://www.dionach .com/blog/the-real-impact-of-cross-site-scripting

Eddington, M., Larcom, B. & Saitta, E. (2005). Trike v1 Methodology Document. Retrieved from: http://www.octotrike.org/papers/

Eng, D. (2017). Integrated Threat Modelling. Department of Informatics Faculty of Mathematics and Natural Sciences. University of Oslo.

Engel, G. (2014). Deconstructing The Cyber Kill Chain. Dark Reading. Retrieved from: https://www.darkreading.com/attacks-breaches/deconstructing-the-cyber-kill -chain/a/d-id/1317542

GoldSim (n.d.). Simulation Primer. Retrieved from: https://www.goldsim.com/web/ introduction/

Goodwin, M. (2017). Open Source Threat Modeling. Core Infrastructure Initiative. Retrieved from: https://www.coreinfrastructure.org/blogs/open-source-threat-modeling/

Greene, T. (2016). Why The 'Cyber Kill Chain' Needs an Upgrade. Network World From IDG. Retrieved from: https://www.networkworld.com/article/3104542/security/why-the-cyber -kill-chain-needs-an-upgradesecurity-pros-need-to-focus-more-on-catching-attackers -aft.html

ImpervaIncapsula (2017). SQL (Structured Query Language) Injection. Retrieved From: https://www.incapsula.com/web-application-security/sql-injection.html

Ingols, K., Chu, M., Lippman, R., Webster, S. & Boyer, S. (2009). Modeling Modern Network Attacks and Countermeasures Using Attack Graphs. *IEEE Annual Computer Security Applications Conference.*

Jaganathan, V., Cherurveettil, P., & Sivashanmugam, M.P. (2015). Using a Prediction Model to Manage Cyber Security Threats. *The Scientific World Journal, Article ID 703713*. Retrieved from: https://doi.org/10.1155/2015/703713.

Jagannathan, V. (n.d.). Threat Modeling: Architecting & Designing with Security in Mind. OWASP. Retrieved from: https://www.owasp.org/images/a/a6/Advanced ThreatModeling.pdf

Kelly, V. (2017). AFP illegally accessed journalist's phone records under new metadata laws. Mumbrella. Retrieved from: https://mumbrella.com.au/afp-illegally-accessed-journalists-phone-records-new-metadata-laws-441378

Khan, R., McLaughlin, K., Laverty, D. & Sezer, S. (2017). STRIDE-based Threat Modeling for Cyber-Physical Systems. *IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*.

Kordy, B., Pietre-Cambacedes, L. & Schweitzer, P. (2013). DAG-Based Attack and Defense Modeling: Don't Miss the Forest for the Attack Trees. University of Luxembourg, France. Retrieved from: https://arxiv.org/pdf/1303.7397.pdf

Korolov, M. & Myers, L. (2018). What is the Cyber Kill Chain? Why It's Not Always the Right Approach to Cyber Attacks. CSO. Retrieved from: https://www.csoonline.com/article/2134037/cyber-attacks-espionage/strategic-planning-erm-the-practicality-of-the-cyber-kill-chain-approach-to-security.html

Kotenko, I. & Stepashkin, M. (2006). Attack Graph Based Evaluation of Network Security. *IFIP International Federation for Information Processing.*

Kozik, R., Choraś, M., Renk, R., Hołubowicz, W. (2014). A Proposal of Algorithm for Web Applications Cyber Attack Detection. *13th IFIP International Conference on Computer Information Systems and Industrial Management (CISIM).* Retrieved from: https://hal.inria.fr/hal-01405662

Krishnan, S. (2017). A Hybrid Approach to Threat Modelling. Semantic Scholar. Retrieved from: https://pdfs.semanticscholar.org/550f/aeb15d64020c6a995c291b6b9f44f8be656d.pdf

Lin, X., Zavarsky, P., Ruhl, R. & Lindskog, D. (2009). Threat Modeling for CSRF Attacks. *IEEE International Conference on Computational Science and Engineering.*

Liu, Y. (2016). Analysis and Detection of Cyberattacks in Smart Home Cyber-Physical Energy Systems. Department of Electrical Engineering. Michigan Technological University.

Lockheed Martin (n.d.) Cyber Kill Chain. Retrieved from: https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

Lucidchart (n.d.) What is a Data Flow Diagram. Retrieved from: https://www.lucidchart.com/pages/data-flow-diagram

Ma, T., & Schmittner, C. (2016). Threat Modeling for Automotive Security Analysis. *Security Technology Conference*. Retrieved from: 333-339. 10.14257/astl.2016.139.68.

Manzoor, S., Zhang, H. & Suri, N. (2018). Threat Modeling and Analysis for the Cloud Ecosystem. *IEEE International Conference on Cloud Engineering (IC2E).* DOI: 10.1109/IC2E.2018.00056

Meland, P.H. (2008). SeaMonster: Providing Tool Support for Security Modeling. Retrieved from: https://pdfs.semanticscholar.org/ac6e/278f9115b898a7ea0a75bbbcce92c97960dd.pdf

Michalsky, R.J. (2014). Four Reasons Why We Shouldn't Kill the Kill Chain. NJVC A Chenega Company. Retrieved from: http://www.njvc.com/blog/cyber-security/four-reasons-why-we-shouldnt-kill-kill-chain

Microsoft (n.d.). SDL Threat Modeling Tool. Security Development Life Cycle. Retrieved from: https://www.microsoft.com/en-us/sdl/adopt/threatmodeling.aspx

Milton, L. C. (2015). User Behavioral Modeling of Web-based Systems for Continuous User Authentication. University of Maryland, College Park.

Mindsight (n.d.). Understanding The Cyber Kill Chain. Retrieved from: https://www.gomindsight.com/blog/understanding-cyber-kill-chain/

Muscat, I. (2017). Prevent SQL Injection Vulnerabilities in PHP application and fix them. Acunetix. Retrieved from: https://www.acunetix.com/blog/articles/prevent-sql-injection-vulnerabilities-in-php-applications/

Noel, S., Wang, L., Singhal, A. & Jajodia, S. (2010). Measuring Risk of Networks Using Attack Graphs. NIST. Retrieved from: https://www.nist.gov/publications/measuring-security-risk-networks-using-attack-graphs

Nostro, N., Ceccareilli, A., Bondavalli, A. & Brancati, F. (2014). Insider Threat Assessment: a Model-Based Methodology. *ACM SIGOPS Operating System Review Volume 48 Issue 2.*

Oladimeji, E.,A., Sapakkul, S. & Chung, L. (2006). Security Threat Modeling and Analysis: A Goal-Oriented Approach.

Omotunde, H., & Ibrahim, R. (2015). A Review of Threat Modelling and Its Hybrid Approaches to Software Security Testing. ARPN Journal of Engineering and Applied Sciences. Vol. 10, No. 23. ResearchGate.

OWASP (n.d.). Threat Modeling. Retrieved from: https://www.owasp.org/index.php/Category:Threat_Modeling

Palanivel, M. & Selvadurai, K. (2014). Risk-driven Security Testing Using Risk Analysis with Threat Modeling Approach. Retrieved from: doi: 10.1186/2193-1801-3-754.

Poniatowski, K. (n.d.). Is the STRIDE Approach Still Relevant for Threat Modelling? Security Innovation. Retrieved from: https://blog.securityinnovation.com/stride

Priya, S.,S. & Arya, S., S. (2016). Threat Modeling for a Secured Software Development. *International Journal of Advanced Research in Computer Science*.

Rouse, M. (2017). Cross-Site Scripting (XSS). TechTarget. Retrieved From: http://searchsoftwarequality.techtarget.com/definition/cross-site-scripting

Rubens, A. (2016). The Ultimate Cheat Sheet on Threat Modelling. CheckMarx. Retrieved from: https://www.checkmarx.com/2016/11/08/ultimate-cheat-sheet-threat-modeling/

Rutherford, J.R. & White, G.B. (2018). Cyber Kill Chain Model Needs a Makeover. The Cyber Edge. Signal. Retrieved from: https://www.afcea.org/content/cyber-kill-chain-model-needs-makeover

Saini, V.K., Duan, Q. & Paruchuri, V. (2008). Threat Modeling Using Attack Trees. ResearchGate. Retrieved from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.878.9000&rep=rep1&type=pdf

Scandariato, R., Wuyts, K. & Joosen, W. (2015). A Descriptive Study of Microsoft's Threat Modeling Technique. Springer. Retrieved from: DOI 10.1007/s00766-013-0195-2

Security Boulevard (2018). The Cyber Kill Chain: What You Need to Know. Retrieved from: https://securityboulevard.com/2018/08/the-cyber-kill-chain-what-you-need-to-know/

Shandilya, V., Simmons, C.B. & Shiva, S. (2014). Use of Attack Graphs in Security Systems. Journal of Computer Networks and Communications. Hindawi. Retrieved from: https://www.hindawi.com/journals/jcnc/2014/818957/

Shar, K.S., Tan, H.,B.,K., & Briand, L.,C. (2013). Mining SQL Injection and Cross Site Scripting Vulnerabilities using Hybrid Program Analysis. *35th International Conference on Software Engineering (ICSE)*.

Shostack (2014). Threat Modeling: Designing for Security. *John Wiley & Sons Incorporation.*

Shostack, A. (2018). Threat Modeling: What, Why and How? MIS Training Institute. Retrieved from: https://misti.com/infosec-insider/threat-modeling-what-why-and-how

Shull, F. (2016). Cyber Threat Modeling: An Evaluation of Three Methods. SEI Blog. Retrieved from: https://insights.sei.cmu.edu/sei_blog/2016/11/cyber-threat-modeling-an-evaluation-of-three-methods.html

Sivula, A. (2015). Security Risk and Threat Models for Health Care Product Development Processes. Jamk University of Applied Sciences.

Stanganelli, J. (2016). Selecting a Threat Risk Model for your Organization, Part Two. eSecurity Planet. Retrieved from: https://www.esecurityplanet.com/network-security/selecting-a-threat-risk-model-for-your-organization-part-two.html

Su, Z., & Wassermann, G. (2006). The Essence of Command Injection Attacks in Web Applications. *ACM POPL Conference Volume 41 Issue 1 Page 372-382.*

ThreatModeler (2016). 7 Benefits of Continuous Threat Modeling. Retrieved from: https://threatmodeler.com/2016/04/15/7-benefits-continuous-threat-modeling/

Val, A. R. (2016). Expanding the Cyber Kill Chain for Embedded System Architectures. Department of Cybersecurity, Utica College.

Vijayan, J. (2018). 7 Threat Modeling Mistakes You're Probably Making. CSO. Retrieved from: https://www.csoonline.com/article/3254135/network-security/7-threat-modelingmistakes-you-re-probably-making.html

Wang, J., Phan, R.C.W, Whitley, J.N., Parish, D.J. (2010). Augmented Attack Tree Modeling of SQL injection attacks. *2nd IEEE International Conference on Information Management and Engineering (ICIME)*.

Wrozek, B. (n.d.). Cyber Kill Chain Methodology. Optiv. Retrieved from: https://www.isaca.org/chapters3/Charlotte/Events/Documents/Event%20Presentations/12062017/Cyber_Kill_Chain_Wrozek.pdf

Yadav, T., & Rao, A.M. (2015). Technical Aspects of Cyber Kill Chain. Security in Computing and Communications. SSCC 2015. Communications in Computer and Information Science, Vol. 536. Springer, Cham.

Zaeem, R.,N., Manoharan, M., Yang, Y. & Barber, K.,S. (2017). Modeling and analysis of identity threat behaviors through text mining of identity theft stories. *Elsevier Computers & Security Volume 65 Pages 50-63*. https://doi.org/10.1016/j.cose.2016.11.002