Security Framework for Web Applications

Submitted By

Tejas Chauhan

13MCEI05

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2015

Security Framework for Web Applications

Major Project

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering

(Information & Network Security)

Submitted By

Tejas Chauhan

(13MCEI05)

Guided By

Prof. Vipul Chudasama

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2015

Certificate

This is to certify that the major project entitled **"Security Framework for Web Applications"** submitted by **Tejas Chauhan (Roll No: 13MCEI05)**, towards the partial fulfillment of the requirements for the award of degree of Master of Technology in Information & Network Security (CSE) of Institute of Technology, Nirma University, Ahmedabad, is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Mr. Asrar Malik
External Guide,
Immix Solution, Ahmedabad.

Prof. Vipul Chudasama
Guide & Assistant Professor,
Institute of Technology,
Nirma University, Ahmedabad.

Prof. Sharada Valiveti
Coordinator M.Tech - INS,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Sanjay Garg
Professor and Head,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr K Kotecha
Director,
Institute of Technology,
Nirma University, Ahmedabad.

Statement of Originality

I, **Tejas Chauhan**, Roll. No. **13MCEI05**, give undertaking that the Major Project entitled "**Security Framework for Web Applications**" submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Computer Science & Engineering** of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

_____

Signature of Student

Date:

Place:

Endorsed by

Prof. Vipul Chudasama

(Signature of Guide)

Acknowledgements

First and foremost, sincere thanks to **Mr. Asrar Malik**, from Immix Solution, Ahmedabad for his valuable guidance. Throughout the training, he has given me much valuable advice on project work. Without him, this project work would never have been completed. I enjoyed his vast knowledge and owe him lots of gratitude for having a profound impact on this report.

It gives me immense pleasure in expressing thanks and profound gratitude to **Prof. Vipul Chudasama**, Assistant Professor, Computer Science Department, Institute of Technology, Nirma University, Ahmedabad for his valuable guidance and continual encouragement throughout this work. The appreciation and continual support he has imparted has been a great motivation to me in reaching a higher goal. His guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

It gives me an immense pleasure to thank **Dr. Sanjay Garg**, Hon'ble Head of Computer Science and Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his kind support and providing basic infrastructure and healthy research environment.

A special thank you is expressed wholeheartedly to **Dr K Kotecha**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation he has extended throughout course of this work.

I would also thank the Institution, all faculty members of Computer Engineering Department, Nirma University, Ahmedabad for their special attention and suggestions towards the project work.

- Tejas Chauhan

13MCEI05

Abstract

As the advancement and progression in internet technologies and computers, one of the main issue which came in the limelight is web security. Every person in the world is using internet as their daily routine but they are not aware with the risks of it. People are doing online transaction without knowing that if they don't use some security mechanisms, someone can steal their session ids and can empty their accounts. Attackers are still able to fool people with some phishing mails which looks like a mail from their bank, asking for their account details.

After every few years, Open Web Application Security Project, publishes a list of top 10 vulnerabilities which are mostly seen in web applications. The top two in the list is SQL Injection and Cross site Scripting [1]. Many websites are also hacked just because of bad file permissions are assigned to website's file and folders and attackers are easily able to put their mailicious code in some core files.

There are various security mechanisms proposed, to make the websites more secure and to detect the attacks. In this paper ideas proposing a web security framework to prevent/detect the attacks related to web applications have been discussed.

There are many types of authentication mechanisms available like passwords, biometric (voice, face and fingerprint) etc, but passwords are easy to implement and easy for users to understand. Passwords also have many drawbacks like if passwords are short or a dictionary word, it can be easy guessed or brute forced by an attacker. It is a difficult task to create and remember passwords that are hard for an attacker to guess. So here we are also proposing a usable password generator which generates usable and memorable passwords for users.

_

Contents

List of Figures

Chapter 1

Introduction

1.1 Security Framework

Web Security deals with the security of websites and web applications. It can be a website of a company with tells a user about them or a question/answer forum to help programmer to solve their issues or a social networking site which a person uses to stay connected to his friends. Few years back most websites were static, but nowadays they became much more interactive. Users can see sliders, popups and much more interactive content which makes a website more user friendly. But as this user friendliness is developed by some client side scripting languages, a new world of vulnerabilities also comes into picture. Nowadays attackers are mostly using client side scripting languages to create their exploits as it runs in client's machine, so they can fetch every details related to user like cookies, session ids, web browsing history etc.

1.1.1 Cross Site Scripting (XSS)

Cross-Site Scripting is the most common vulnerability which exists in web applications and it occurs because of data that is entered by the user is not properly filtered. Most modern websites ask for user input for many purposes like feedback form, contact us form, search query etc. Web site owners thinks that user will write text in those input fields for which they are meant to be but attackers takes advantage of that and writes malicious codes in those input fields. If proper security mechanisms are not in place, then attacker can execute those scripts in another user's machine as javascript and other client side scripting languages runs in user's machine.

In most common way, a well crafted link is presented to user in the form of mail. This link will have valid domain, but some other scripting code is also attached with that link. As the malicious scripting code is converted into some non readable format using some encoding mechanisms and domain link looks genuine to the user, most users clicks on that link. when user clicks, attached code runs into user's browser without user's consent and steals some confidential information.

There are mainly two kinds of XSS attacks: Stored and Reflected. In stored XSS, the malicious code will persistently stored in database and which can affect other users also, who visits the pages which contains that malicious code. For example, Suppose we have a ecommerce website and we allow the user to write some reviews about our products. Legitimate users will write reviews related to product, but attackers can use those input boxes to inject malicious code. If an attacker is able to inject those malicious code and if proper filtering mechanisms is not in place, it will be stored in the database. Now every person who visits that same product page, can be attacked as that malicious scripting code will run in visitor's browser. Now suppose attacker had wrote some code to steal session ids of user, it can represent himself as any visitor he likes. In Reflected XSS, the malicious code is not persistently stored in the database, but it is reflected back to the user. For Example, Attacker can send the user a link which shows some search results related to a keyword. That link contains the URL of search results with which some scripting code is also attached. When user clicks on that link, he will see a list of search results but he will not be aware of some scripting code is also executed without his consent.

1.1.2 SQL Injection

SQL Injection is an attack on web pages when user input contains some sql statements. In this attack some sql queries are executed in the database results in theft of some confidential infomration and even database crash. For example, Suppose we have a login form, which contains two input fields, one for username and another for password. Now whenever user enters a valid username/password combination, a sql query is executed in database to check that combination is valid or not. The query looks like this,

```
SELECT * from user_data where username='USERNAME' and password = 'PASSWORD'
```

Here, USERNAME and PASSWORD are the user input. now suppose an attacker try to put " ' or '1'='1' – " as username, then sql query becomes,

```
SELECT * from user_data where username='USERNAME' or '1' = '1' -- and
password = 'PASSWORD'
```

By above query, attacker is able to acccess the system as '1' = '1' is always true, and in sql "–" means comment so, sql server will ignore the remaining password matching part.

1.1.3 .htaccess

.htaccess is a configuration file for web servers. .htaccess file can be used for authentication, url rewrites, block some ip addresses etc. In our implementaion we used .htaccess file to create some rules. This rules will check all form values which is passed through GET/POST method. If any values matches with some predefined rules it will be blocked immediately. We also used .htaccess file to create some rules which will block some ip addresses to access our website. Web owner can set these ip addresses directly from admin panel in our demo implementation.

1.1.4 Damn Vulnerable Web App

DVWA is a web application, which is developed for teaching purposes. DVWA is the demo environment in which students or professionals can learn how we can secure websites. It enables a programmer to think like an attacker and allows him to attack a website in a demo environment. Programmer can learn basic security measures and can apply it to real websites.

1.2 Usable Password Generator

Users generally create passwords which can be easily guessed or brute forced. To enforce users create secure passwords, websites started to use password policies and password strength meters. Unfortunately, password policies does not help to create secure passwords as users may fulfill policy requirments in predictable way like, creating their new

password based on old password and also using same password in multiple domains.

A passphrase is a password which contains a sequence of words. They are longer compared to oridinary passwords. Passphrases are more secure and easier to remember. Usable password generator automatically creates a passphrase from the input data supplied by the user. It is like a sentence which can be easily remembered but can not be brute forced easily. We can also make it more secure by some extra policies of adding uppercase, lowercase, special symbols.

Chapter 2

Literature Survey - Security Framework

This chapter provides an overview of the research done in filtering and validating user input. The first section provides a brief overview of various approaches which are already applied. The later part of this chapter contains a table which contains a list of papers that we have read and approaches which are already used.

2.1 Different Approaches

2.1.1 Dynamic Data Tainting

In dynamic data tainting approach, some confidential information like cookies or session ids, are first marked as secret. If any client side script, tries to access those secret values, it will be carefully monitored and some logs will be generated for that [2]. In some cases, access to this values will be blocked also. The problem with this approach is, many web owners put some scripts on their websites to know statistics about it. Statistics includes traffic on the website, number of hits per page, user tracking etc.[3] This scripts usually accesses some cookie values and pass it to servers which provides web statistics information like Google Analytics. Now in this case, this cookie value transfers are not done by any cross site scripting attacks. So if we apply dynamic data tainting approach in this scenario, it will result in large number of false positives. [4]

2.1.2 Static Code Analysis

Static code analysis is a commonly used technique to find security weaknesses in source code of the website. In this approach, source code of a web application is manually examined and existing loopholes need to be finding out. It ensures that secure programming practices are used or not while developing a web application [5].

However, web applications can also store untrusted data to external resources and later on access and reuse it, a problem that is overlooked in this approach. It is also a tedious task for web owners to check the source code of entire website and remove the vulnerabilities one by one.

2.1.3 Dynamic Analysis Techniques

In dynamic analysis techniques, a list of XSS attack values are designed for each input of every form of the website and submitted to the Web Application [6]. It will check that, a perticular web application is vulnerable to that XSS attack value or not. This strategy may be supported by a Web Application testing tool that automatically executes the test cases. The problem with this approach is, website owners don't check it in real environment for every input field on their website. [7]

2.2 List of papers and some approaches which are alreday used

Paper Title	Approach Used
Cross-Site Scripting Prevention with Dynamic Data Tainting and Static Analysis	Confidential information like cookies or session ids, are first marked as secret. If any client side script, tries to access those secret values, it will be carefully monitored and some logs will be generated for that. In some cases, access to this values will be blocked also.
Noxes A Client-Side Solution for Mitigating Cross-Site Scripting Attacks	A standalone firewall is developed to check every request and response of the browser. Firewall is pre configured with some security policies and each request/response will be matched against those policies.
Identifying Cross Site Vulnerabilities in Web Applications	Both static and dynamic analysis techniques are used. In static analysis, a control flow diagram is developed. CFG nodes are created for each input and output values. The page is considered as vulnerable if there is a path in CFG, which connects a input and output node. In Dynamic Analysis, a set of XSS attack values are submitted to the web application.
BLUEPRINT: Robust Prevention of Cross-site Scripting Attacks for Existing Browsers	A whole new javascript parser is developed which replaces browser's javascript parser. Parser contains many security mechanisms which tracks the behavious of client side scripts and blocks it, if it seems vulnerable.

Implementation of Automatic Detection for Cross-Site Scripting Vulnerability	It mainly targets reflected XSS attacks. A user side proxy is used to check every request/response combination. If any request contains some special characters which are pre configured in proxy, request will be stored and the response of that request will be checked. If the same special characters exists in the response message, it will be blocked and client side script will not be allowed to execute.
Static Detection of Second-Order Vulnerabilities in Web Applications	In data flow analysis, if data is read from the data store, it will be tainted. All taintable writings are used to detect second order vulnerabilities.
MUTEC: Mutation-based Testing of Cross Site Scripting	It is a testing technique in which a faulty implementation called mutants is generated against the original program. A list of vulnerable XSS strings are injected to both, original and faulty application. A mutant will be killed if both outputs differs otherwise it will be called as live mutant and some changes are needed in the original program to kill that.
XSSDS: Server-side Detection of Cross-site Scripting Attacks	It is a server side technique in which HTTP traffic is monitored. It checks input parameters and generated output to secure a website against reflected XSS attacks. It also keeps track of each javascript which is deployed by the website owner and it differentiate all other javascript which is injected by user and will not allow it to execute.

Detection of SQL Injection and Cross-site Scripting Attacks	Create regular-expression based rules for detecting XSS attacks. Apply those rules to open-source IDS Snort.
Session Shield: Lightweight Protection against Session Hijacking	It checks http header of each request, whether they set cookie value or not. If any attempt is detected to set a session id as cookie value, it will be stored in the internal database. In later client request, it will add the session id information. Client side script will not be allowed to access the internal database where session ids are stored thus securing user against XSS and session hijacking attacks.
One-Time Cookies: Preventing Session Hijacking Attacks with Disposable Credentials	Session hijacking attack occurs as same session id is used for each request of the user. OTC prevents that scenario, and for each request generates a different session id. So, if any attacker eavesdrop the ongoing communication and able to fetch the session id, he will not able to hijack the user's session as it changes for every new request. It uses cryptographic hash functions to generate a new random value for each request.

Chapter 3

Literature Survey - Usable Password Generator

3.1 Usable Security

While designing a security system, we should consider the users which are going to use it and how they will use it. It should not be a frustrating part for users.
Measuring Usability

- Speed : How quickly can the task be accomplished?
- Efficiency : How many mistakes are made in accomplishing the task?
- Learnability : How easy is it to learn to use the system?
- Memorability : Once learned, how easy is it to remember how to use the system?
- User Preference : What do users like?

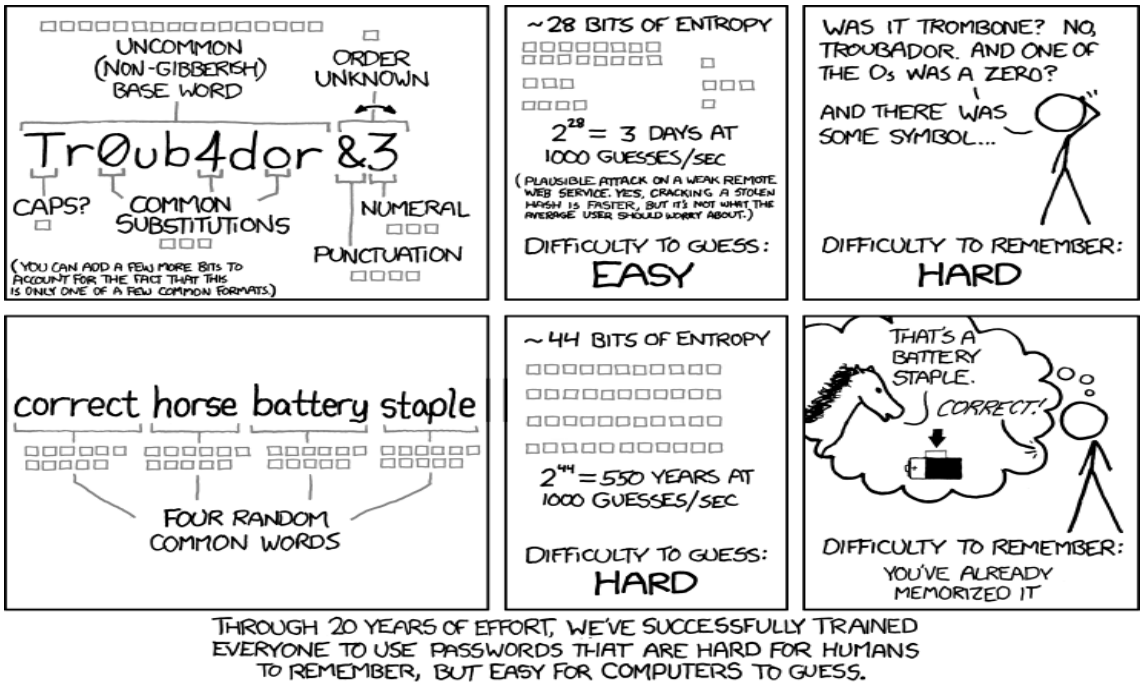
3.2 Secure Memorable Passwords

3.2.1 Secure Memorable Passwords

We reviewed many of the password generator available on firefox addon store. Usually all password generators provide a popup where user can select some options and according to the selected parameters it generates a unique password. Though those passwords are secure, they were not memorable.

Figure 3.1: Secure Password Generator

Figure 3.2: Memorable Passwords



3.3 Password Entropy

Password entropy is defined as a passwords randomness, in regards to how difficult it would be to crack. [8]

3.3.1 NIST Guidelines

NIST Special Publication of June 2004 [9] suggests the following scheme to roughly estimate the entropy of human-generated passwords:

- The entropy of the first character is four bits;
- The entropy of the next seven characters are two bits per character;
- The ninth through the twentieth character has 1.5 bits of entropy per character;
- Characters 21 and above have one bit of entropy per character.
- A "bonus" of six bits is added if both upper case letters and non-alphabetic characters are used.
- A "bonus" of six bits is added for passwords of length 1 through 19 characters following an extensive dictionary check to ensure the password is not contained within a large dictionary. Passwords of 20 characters or more do not receive this bonus because it is assumed they are pass-phrases consisting of multiple dictionary words.

Using this scheme, an eight-character human-selected password without upper case letters and non-alphabetic characters is estimated to have 18 bits of entropy.

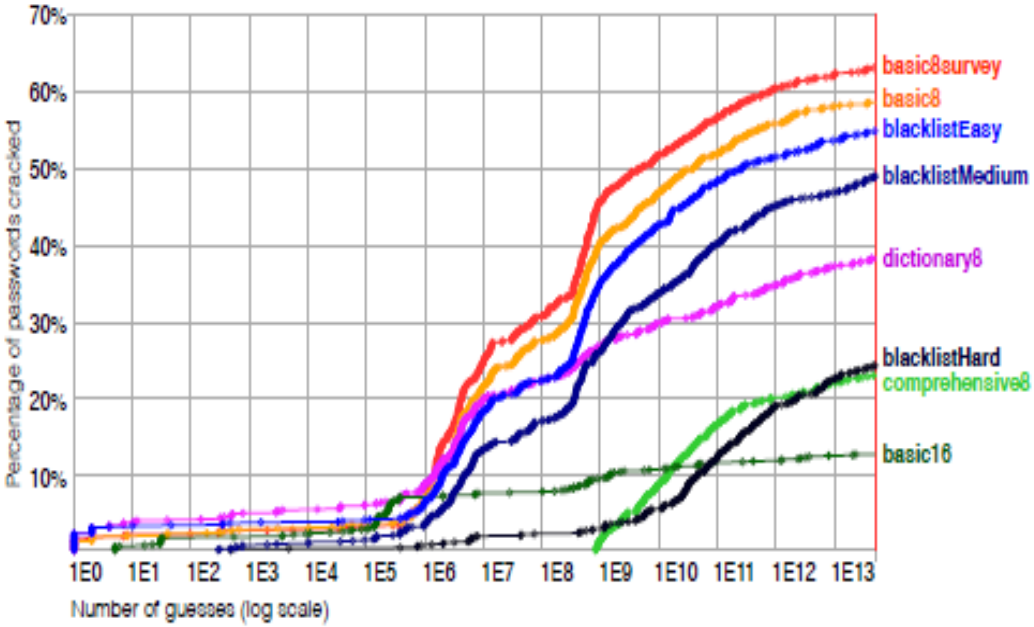
3.3.2 Shannon's Formula

If we have a set of N symbols and if we select L number of symbols from that set then, the number of possible passwords can be calculated by raising N to power L. By incrementing either L or N we can more strengthen the generated password. A password's entropy H,

Figure 3.3: Shannon's Formula

$$H = log_2 N^L = L \log_2 N = L \frac{\log N}{\log 2}$$

Figure 3.4: Password Policies



can be given by following Shannon's formula

where N is the number of symbols and L is number of symbols in generated password.

3.4 Comparison of Password Policies

The eight conditions are detailed below [10].

- basic8survey: Password should be atleast 8 characters long. It is in survey scenario.
- basic8: Password should be atleast 8 characters long. It is in email scenario.
- basic16: Password should be atleast 16 characters long.
- dictionary8: Password should be atleast 8 characters long. It should not have a dictionary word.

- comprehensive8: Password should be atleast 8 characters long. It must have an uppercase and lowercase letter, a symbol, and a digit. It should not have a dictionary word.
- blacklistEasy: Password should be atleast 8 characters long. It should not have a dictionary word. Here, the password is compared against simple unix dictionary.
- blacklistMedium: Password should be atleast 8 characters long. It should not have a dictionary word. Here, the password is compared against paid Openwall list.
- blacklistHard:Password should be atleast 8 characters long. It should not have a dictionary word. Here, the password is compared against a five-billion-word dictionary.

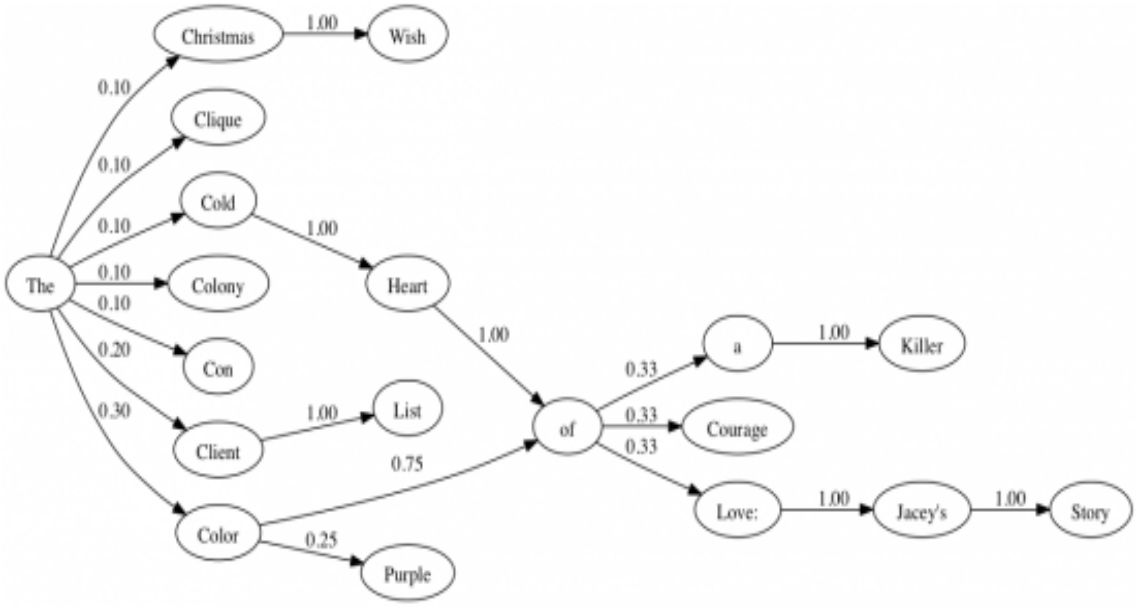
3.5 Markov Chains

Markov chains, named after russian mathematician Andrey Markov is a stochastic process in which outcome of an experiment depends only on the outcome of the previous experiment, means the next state of the system depends only on the current state and not on other previous states. [11] A stochastic process is a mathematical model that evolves over time in a probabilistic manner. For example, if we model a babys behavior as a markov chain we can consider eating, crying, sleeping and playing as different states. We can also list out all other possible states which will become our state space. Now a markov chain tells us the probability (chances) of transition from one state to another e.g, the probability of a baby currently playing will sleep in next ten minutes without crying.

In our demo implementation, we used a list of titles which is used to generate our markov chain model. Now we can use that statistical information to generate new movie titles by selecting the first word at random and then selecting other subsequent words with probability of how they are arranged in the original list. This gives us a list of different movie titles which are randomly created and will not be in that list.

First we list out all the unique words which appear in those movie titles. Next we calculated the probability of one word following the other word. In this way whole graph

Figure 3.5: Markov Chains



is generated, vertex represents unique words and value above each edge represents the probability of one word following the other word. In above graph we can say the word The is followed by Colony with the probability of 0.1, while it is followed by Client and Color with the probability of 0.2 and 0.3 respectively. Now as we can see, we can traverse through any path of graph and generate different unique titles.

Chapter 4

Implementation - Security Framework

To overcome problems as discussed in previous chapter, here is an approach proposed in which a standalone security framework is used to check user input and allow/reject those as per preconfigure rules. By standalone, we mean it can be used as the sub part of any website that we want to secure. Every request or user input which comes to the website will pass through our security framework. Instead of using some proxy servers and some strict firewall rules to check each and every request, Our security framework can just be installed as a sub folder of any website.

4.1 Security Framework Implementation

It is a demo setup of standalone framework which can be used to secure website.

The installation takes following steps:

- Copy our security framework as subfolder of website which we want to secure.
- In first step we need to enter database details, it will check those details and create a database and some required tables.
- In second step, it will ask for domain URL which we want to secure.
- In Third step, it will allow the website owner to create an admin account by which he can login to our secure framework.

Figure 4.1: Proposed System

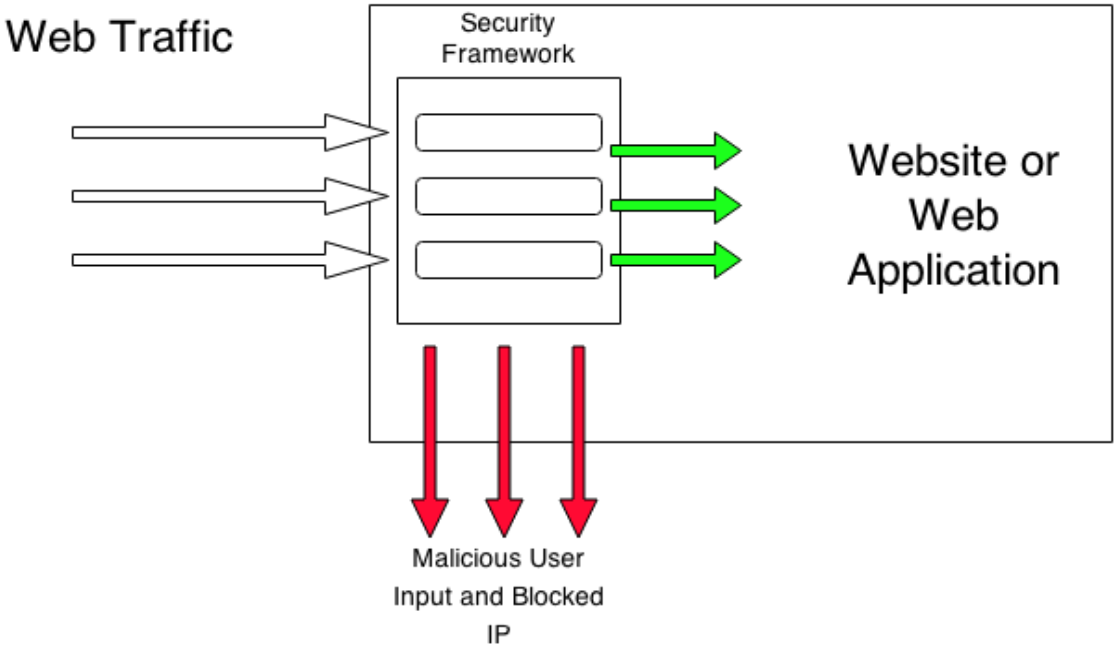
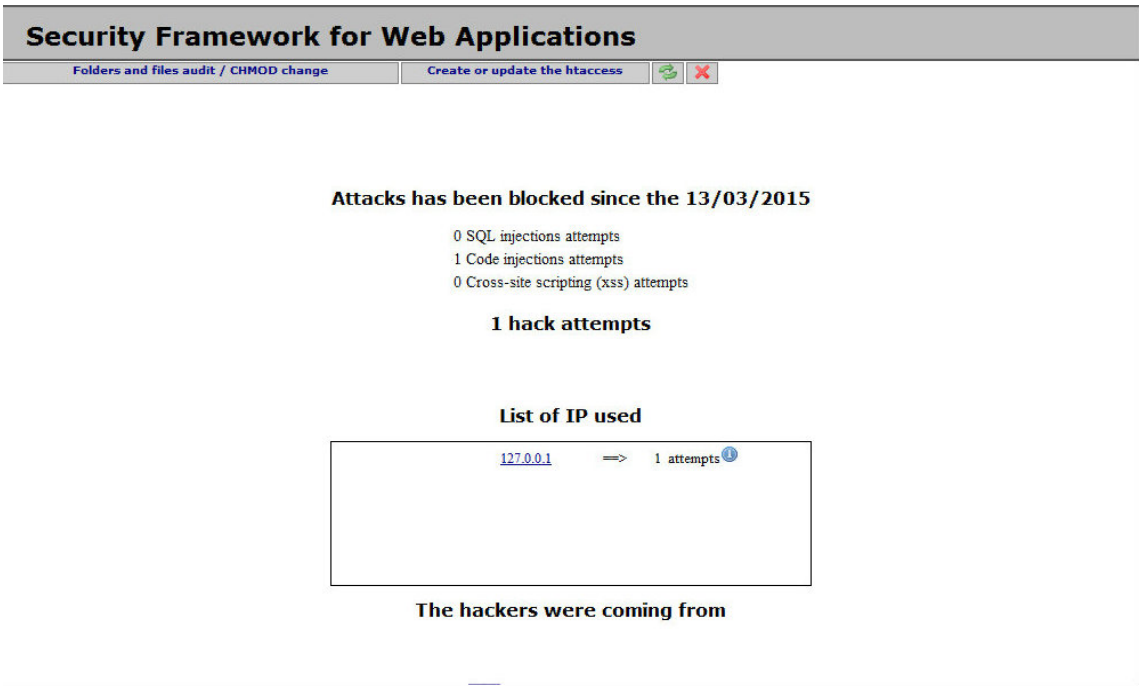


Figure 4.2: Admin - Attack Overview



Now in admin area, we added an option to create .htaccess file. While creating .htaccess file, we allows the website owner to set some parameters, according to which that file will be created. We allow the web site owner to select following options while creating .htaccess file.

- Trusted IP: Only by mentioned IP, website owner can login to our secure framework.
- Some keywords related to protection from cross site scripting and sql injection.
- IP Addresses that website owner wants to block.
- Secure access to some folders and files only to trusted IPs.

According to selected parameters, it will create .htaccess file and will put that into root folder. Now if any attacker tries to attack the website it will be blocked if it matches with the rules created in .htaccess. Website owner can see from admin panel, from which IP address attack has been performed and can block those IP addresses.

I have also tested this security framework for a set of input values with Damn Vulnerable Web App, a very popular vulnerable web application. Our framework correctly identifies xss and sql injection attacks which are predefined in .htaccess file.

We added the functionally in which user can set file permissions of files and folders of a website directly from admin. By this a website owner can prevent invalid file permission related attacks.

4.2 Input Filtering Class

I have also developed a standalone PHP class that can filter malicious user input. It allows the programmer to predefine a set of tags and attributes which are malicious. So, whenever any attacker comes to our site and try to attack it with some malicious input, it will be compared with our predefined values and if it seems like an attack, our class will filter it and remove those tags and attributes from input. So in this way, it can be used to stop cross site scripting(XSS) attacks.

Figure 4.3: File Permissions

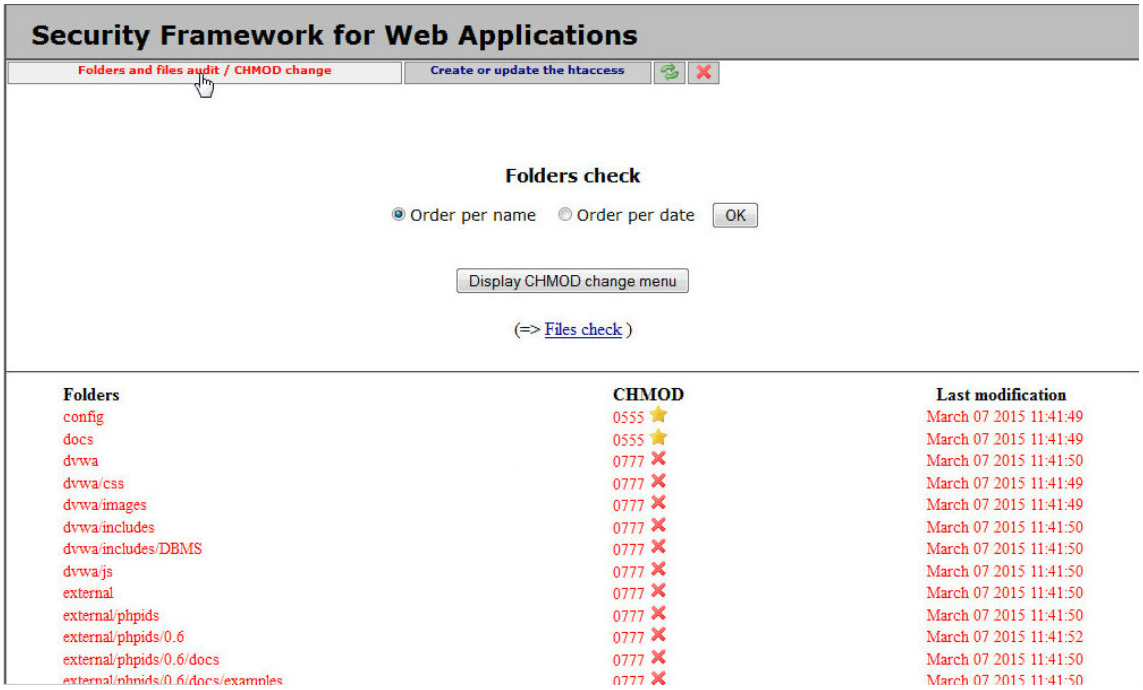


Figure 4.4: Input Filter Class

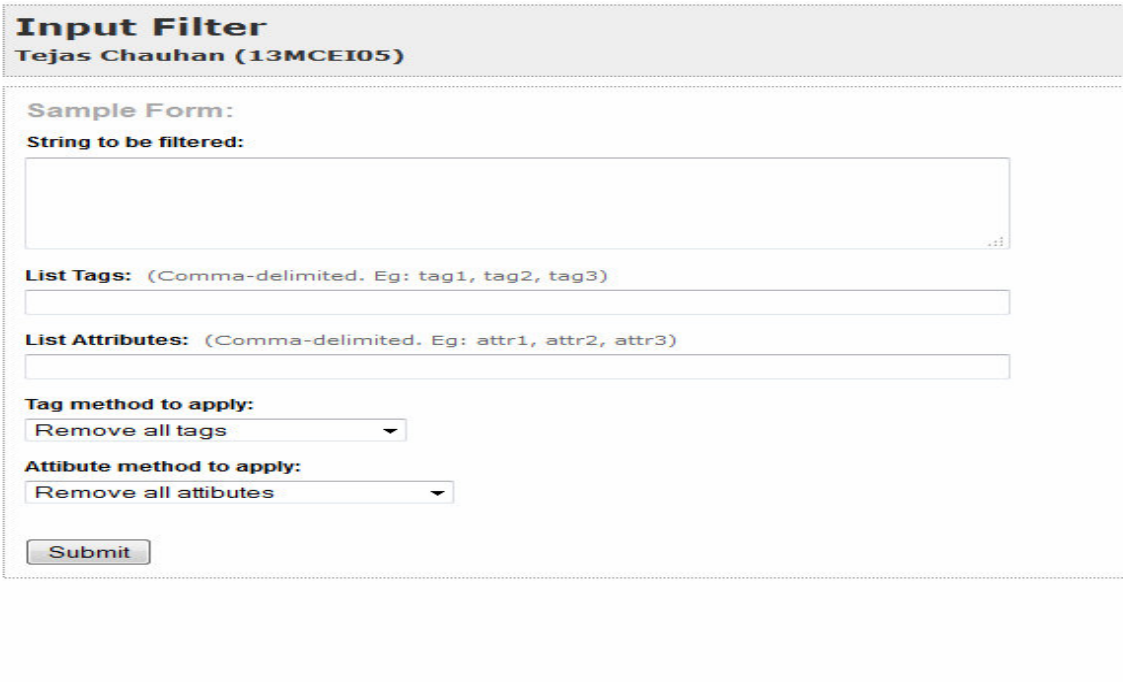
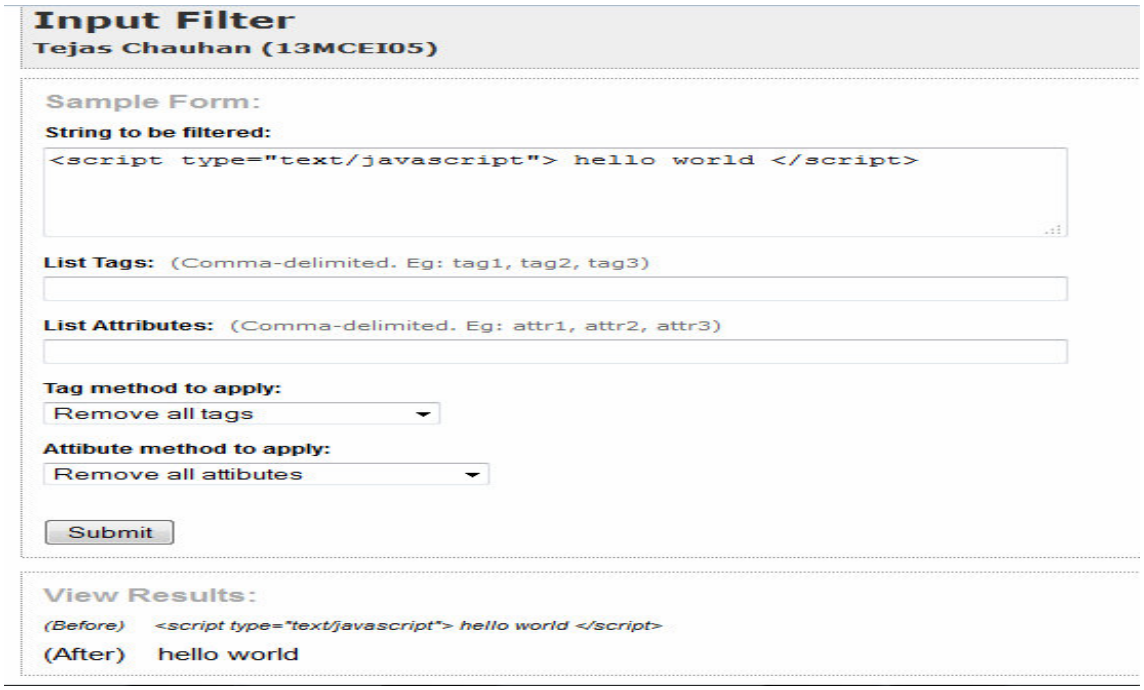


Figure 4.5: View Results



Chapter 5

Implementation - Usable Password Generator

We implemented a demo version of password generator which generates sentences as password. It is implemented as a Firefox extension so anyone can install it in their browser. To create a Firefox extension, we need to get familier with two technologies,

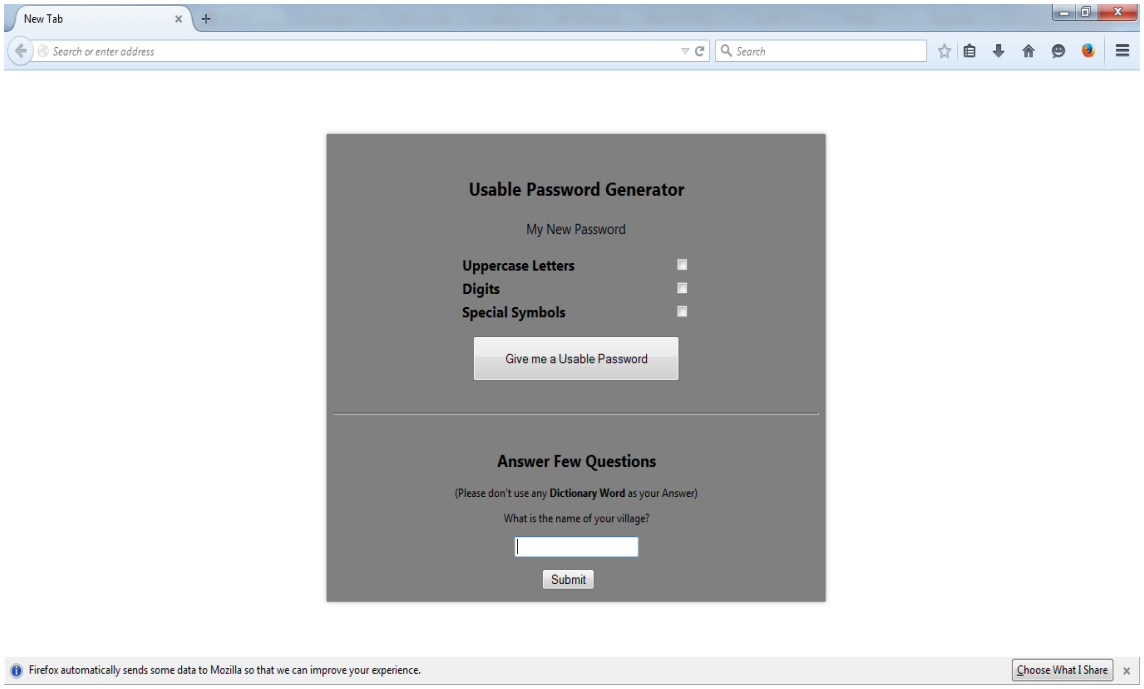
5.1 XUL

XUL (XML User Interface Language - pronounced "zool") is one of many technologies used for creating user interface of Mozilla extensions. XUL is used to create portable and cross platform user interfaces. It takes so much time to develop an application for one platform, with XUL we can develop user interfaces which can be modified quickly and easily across multiple platforms.

5.2 Javascript

JavaScript is a client side scripting language. It will run at client side (web browser) and used to develop interactive web pages. In Mozilla extensions, it is used to write application related functions.

Figure 5.1: Usable Password Generator

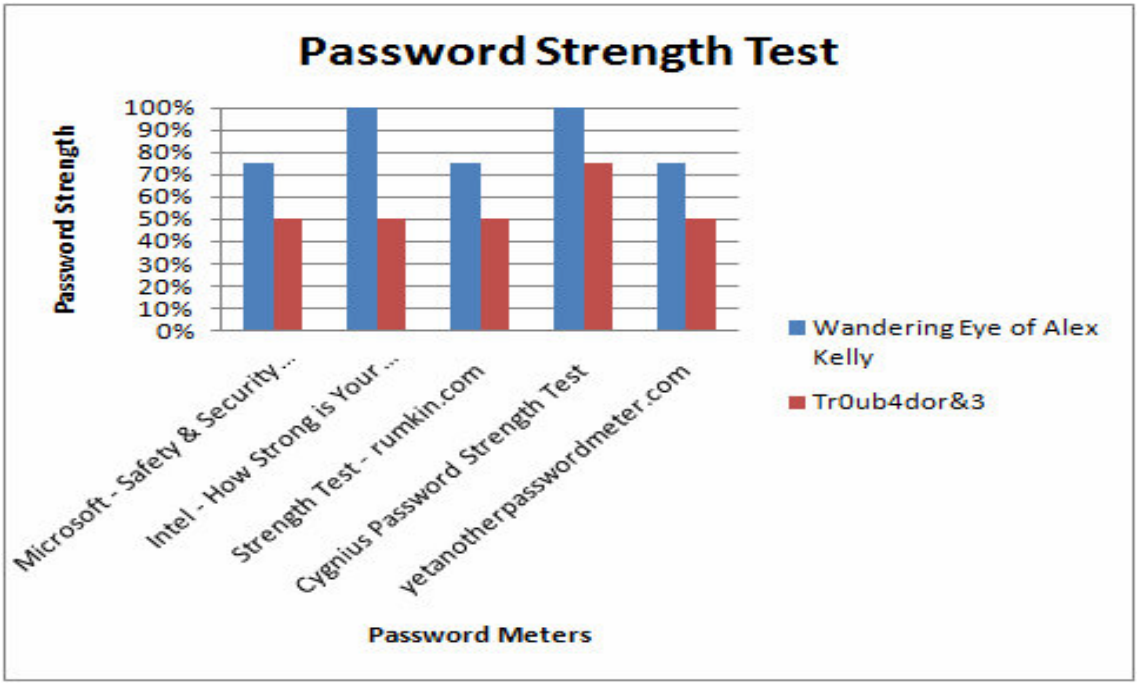


5.3 Password Strength Test

We used some well known password meters to check strength of password created by usable password generator which is compared with password created by a password generator which is available on firefox addon store.

Here, "Wandering Eye of Alex Kelly" is the password generated by our password generator and "Tr0ub4dor3" is the password generated by other password generator available on addon store.

Figure 5.2: Password Strength Test



Chapter 6

Conclusion and Future Work

6.1 Conclusion

All web application should validate and filter any user input and block malicious code. In our demo setup we have designed and developed an security framework which contains various mechanisms that can detect malicious code from user input and it also adds mechanisms to block some IP addresses from accessing our website. Proper usage of our security framework will result in the target system being much more secure.

For Usable Password Generators, As per the different password conditions we tested, basic16 provides more security compared to other policies. Our demo implementation successfully generates passphrases which can be more memorable and hard for an attacker to break.

6.2 Future Work

In Usable Password Generator, We will further implement random sentense generator in a firefox extension and upload it on addon store.

References

[1] "Top 10 - 2013 (the ten most critical web application security risks (2013))," *OWASP*, 2013.

[2] N. J. E. K. C. K. Philipp Vogt, Florian Nentwich and G. Vigna, "Cross-site scripting prevention with dynamic data tainting and static analysis," *In NDSS*, 2011.

[3] M. G.A.Di Lucca, A.R.Fasolino and P.Tramontana, "Identifying cross site scripting vulnerabilities in web applications," *IEEE*, 2004.

[4] G. V. Engin Kirda, Christopher Kruegel and N. Jovanovic, "Noxes: A client-side solution for mitigating cross-site scripting attacks," *ACM*, 2006.

[5] J. Dahse and T. Holz, "Static detection of second-order vulnerabilities in web applications," *USENIX Security Symposium*, 2014.

[6] M. T. Louw and V. Venkatakrishnan, "Blueprint: Robust prevention of cross-site scripting attacks for existing browsers," *IEEE Symposium on Security and Privacy*, 2009.

[7] B. E. Martin Johns and J. Posegga, "Xssds: Server-side detection of cross-site scripting attacks," *Annual Computer Security Applications Conference*, 2008.

[8] S. K. M. L. M. B. U. T. V. L. B. N. C. Richard Shay, Patrick Gage Kelley and L. F. Cranor, "Correct horse battery staple: Exploring the usability of system-assigned passphrases," *ACM*, 2012.

[9] U. W. Klaus Peter Jochum and B. Stoll, "Determination of reference values for nist srm 610617 glasses following iso guidelines," *23rd USENIX Security Symposium*, 2012.

[10] P. G. K. M. L. M. L. B. N. C. L. F. C. Saranga Komanduri, Richard Shay and S. Egelman, "Of passwords and people: Measuring the effect of password-composition policies," *ACM*, 2011.

[11] M. L. M. R. S. T. V. L. N. C. L. F. C. Patrick Gage Kelley, Saranga Komanduri and J. Lopez, "Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms," *CMU-CyLab-11-008*, 2011.