



**A MODIFIED SECURITY APPROACH FOR WEB APPLICATION USED
FOR THE REGISTRATION OF THE STUDENT AT
ÇANKAYA UNIVERSITY**

ALI SAMI SOSA

DECEMBER 2014

**A MODIFIED SECURITY APPROACH FOR WEB APPLICATION USED
FOR THE REGISTRATION OF THE STUDENT AT
ÇANKAYA UNIVERSITY**

**A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED
SCIENCES OF
ÇANKAYA UNIVERSITY**

BY

ALI SAMI SOSA

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF
MASTER OF SCIENCE**

IN

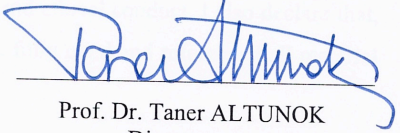
**THE DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE
/INFORMATION TECHNOLOGY PROGRAM**

DECEMBER 2014

**Title of the Thesis: A Modified Security Approach For Web Application Used For
The Registration Of The Student At Çankaya University.**

Submitted by **ALI SAMI SOSA**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya
University.



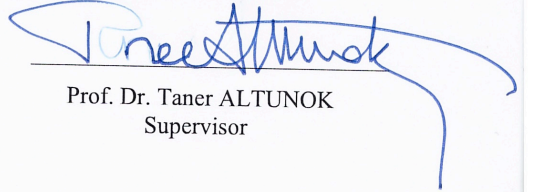
Prof. Dr. Taner ALTUNOK
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of
Master of Science.



Prof. Dr. Billur KAYMAKÇALAN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully
adequate, in scope and quality, as a thesis for the degree of Master of Science.



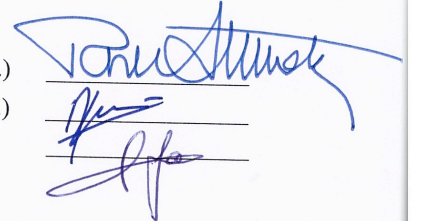
Prof. Dr. Taner ALTUNOK
Supervisor

Examination Date: 26.12.2014

Examining Committee Members

Prof. Dr. Taner ALTUNOK
Assist Prof. Dr. Abdül Kadir GÖRÜR
Assoc.Prof. Dr. Fahd JARAD

(Çankaya Univ.)
(Çankaya Univ.)
(THK Univ.)



STATEMENT OF NON-PLAGIARISM PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Ali Sami Sosa

Signature :

Date : 26.12.2014

ABSTRACT

A MODIFIED SECURITY APPROACH FOR WEB APPLICATION USED FOR THE REGISTRATION OF THE STUDENT AT ÇANKAYA UNIVERSITY

SOSA, Ali Sami

M.Sc., Department of Mathematics and Computer Science / Information Technology
Program

Supervisor: Prof. Dr. Taner ALTUNOK

December 2014, 85 pages

This thesis goal is to design and implement a modified security approach for web-based system for student's registration in Natural and Applied Science Institute of Çankaya University Web application that help students to register by using the Internet. The employee can also use this application to track follow student status by monitoring the student information and give a report for each user, this proposed application can reduce time and effort for the users. The proposed application system can also be applied on tablet and smartphone (Iphone and Ipad).

Keywords: Web Application, Web Security, Injection Attacks, SQL Injection, XSS, Attack, PHP.

ÖZ

ÇANKAYA ÜNİVERSİTESİ ÖĞRENCİ KAYDI İÇİN KULLANILAN WEB MODİFİYE GÜVENLİK YAKLAŞIMI

SOSA, Ali Sami

Yüksek Lisans, Matematik-Bilgisayar Anabilim Dalı / Bilgi Teknolojileri Programı

Tez Yöneticisi: Prof. Dr. Taner ALTUNOK

Aralık 2014, 85 sayfa

Bu tezin amacı, Çankaya Üniversitesi Fen Bilimleri Enstitüsü'ne başvuran öğrencilerin Internet kullanarak kolayca başvurularını yapabilmelerine yardımcı olmak için, web tabanlı kayıt sisteminin daha da güvenli olabilmesi adına bir ileri düzey sistem dizayn edilmesi ve uygulamasıdır. Sistemi kullanan görevli kişi bu sistemde, her bir öğrencinin detay bilgilerini, durumunu, kolayca raporlayabilir ve her bir sistem ortak kullanıcısı ile paylaşabilir. Bu uygulama önemli ölçüde zaman tasarrufu ve kullanım kolaylığı sağlar. Önerilen bu uygulama, bu sistem, I-phone ve I-pad, gibi akıllı elektronik cihazlarda da kullanılabilir.

Anahtar Kelimeler: Web Uygulaması, Web Güvenliği, Enjeksiyon Saldırıları, SQL Enjeksiyon, XSS, Saldırı, PHP.

ACKNOWLEDGEMENTS

First and foremost, I would like to praise the Almighty Lord for all the blessings and opportunities bestowed upon me. I would like to express my sincere gratitude and deep appreciation to my advisor Prof. Dr. Taner Altunok for his unlimited guidance, assistance, encouragement and tremendous patience throughout this research. Besides my advisor, I would like to thank the rest of my thesis committee for their constructive criticism and insight. Also, I would like to thank Çankaya University, for their support and facilities. Here I would also like to thank my parents and my siblings who encouraged me with their best wishes and moral support. Finally I would like to thank my wife and sweet little daughter for their unconditional support, friendship and love through good times and bad.

TABLE OF CONTENTS

STATEMENT OF NON PLAGIARISM	iii
ABSTRACT	iv
ÖZ	v
ACKNOWLEDGMENTS	vi
TABLE OF CONTENTS	vii
LIST OF TABLES	ix
LIST OF FIGURES	X
LIST OF ABBREVIATIONS	Xiii
LIST OF SYMBOLS	Xiv

CHAPTERS

1.INTRODUCTION	1
1.1. Background	1
1.2. The Literature Survey	2
1.3. Aim of the Thesis	5
1.4. Thesis Outline	5
2. WEB AND SMARTPHONE APPLICAION, SPECIFICATIONS, AND SECURITY	
2.1.Web Application	6
2.1.1.Introduction	6
2.1.2.History	6
2.1.3.A Web application interface	8
2.1.4.Core structure	8
2.1.5.Scope of business	9
2.1.6.Writing the web applications	10
2.1.7.Some applications	11
2.1.8.Web application benefits	11

2.1.9. Web application development	11
2.1.10. Web languages	12
2.1.10.1. PHP	12
2.2. Web Application Security	13
2.2.1. Kinds of web applications attacks	15
2.2.1.1. When the users provide the information	15
2.2.1.2. Human attacks	15
2.2.1.3. Automated attacks	16
2.2.1.4. When the information is specified to users	18
2.2.2. Injection attacks	19
2.2.2.1. A Cross site scripting (XSS) attack.....	20
2.2.2.1.1. Introduction	20
2.2.2.1.2. Preventing XSS	23
2.2.2.1.3. Types of XSS	24
2.2.2.1.3.1. A persistent (stored) XSS	24
2.2.2.1.3.2. A non persistent (reflected) XSS	27
2.2.2.1.3.3. Non persistent demo (reflected) XSS	27
2.2.2.1.3.4. DOM based XSS	29
2.2.2.1.4. The advanced techniques	29
2.2.2.1.5. XSS prevention	30
2.2.2.1.5.1. XSS prevention strategies	31
2.2.2.1.5.1.1. Make and encode to HTML entities in non-HTML	
output	31
2.2.2.1.5.1.2. Make a Sanitize for every user-submitted URIs	32
2.2.2.1.5.1.3. Use a proven XSS filter on HTML input	33
2.2.2.1.5.1.4. Make a private API strategy for sensitive	
transactions	34
2.2.2.1.5.1.5. Predict some actions that expect from users	36
2.2.2.2. SQL Injection	37
2.2.2.2.1. Introduction	37

2.2.2.2.2.Fundamentals	38
2.2.2.2.3. Strike process	39
2.2.2.2.4. Terminology	40
2.2.2.2.5. Security services	40
2.2.2.2.6. Strike methods	41
2.2.2.2.7. Pre-conditions	42
2.2.2.2.8.SQL inserting security model	44
2.2.2.2.9.SQL inserting strike samples	44
2.2.2.2.10. Preventing SQL injection	46
2.2.2.2.10.1.Demarcate each value in the queries	46
2.2.2.2.10.2.Checking the values types of users' submitted	47
2.2.2.2.10.3.Abstract to improve security	47
2.3.Smartphone application	48
2.3.1. Introduction	48
2.3.2. History	49
2.3.3. Home screen	50
2.3.4. Folders	50
2.3.5. Multitasking	51
2.3.6. Development	52
2.3.7. App store (IOS)	52
2.3.8 IOS SDK	53
3. DESIGN AND IMPLEMENTATION TOR WEB APPLICATION	
3.1. Introduction	55
3.2. System Features	56
3.3. System Function	57
3.4. System Implementation	57
3.4.1. An Environment of system development	58
3.4.2. System development languages	58
3.4.3. Main system structure	60
3.4.4. Database design	66

3.5. Smartphone Application	66
3.5.1. Introduction	66
3.5.2. Main function	67
3.5.3. Application development language	67
4. RESULTS AND DISCUSSION	
4.1. Web Application	68
4.1.1. Home page	68
4.1.2 Student register	68
4.1.3 Login	71
4.2. Mobile Application	83
5. CONCLUSIONS	
5.1 Conclusions	84
5.2 Future Work	85
REFERENCES	R1
APPENDICES	A1
A. CURRICULUM VITAE	A1

LIST OF FIGURES

FIGURES

Figure 1	Stored messages with XSS attack	21
Figure 2	There are some steps for a XSS attack with reflection	22
Figure 3	A Persistent (Stored) XSS PHP code	25
Figure 4	A Persistent (Stored) XSS	26
Figure 5	A Persistent (Stored) XSS	26
Figure 6	PHP code suffers from non-persistent XSS	27
Figure 7	Non-persistent demo (reflected) XSS	28
Figure 8	An XSS alert in non-persistent Demo (reflected) XSS	29
Figure 9	PHP code that reflects an XSS	30
Figure 10	PHP code that encode to HTML Entities in non-HTML output	31
Figure 11	PHP code sanitize for every user-submitted URIs	33
Figure 12	PHP code use API strategy for sensitive transactions	35
Figure 13	PHP code use API strategy for sensitive transactions	35
Figure 14	Main system function (Use Case) diagram	57
Figure 15	PHP retrieves the data from MySQL database	58
Figure 16	Main system layout	60
Figure 17	Main system structure	61
Figure 18	Home page	68
Figure 19	Student register	69
Figure 20	Student register	69
Figure 21	Student register	70
Figure 22	Student register	70
Figure 23	Student register	71
Figure 24	User login	72
Figure 25	User logged in to the status page	72
Figure 26	User logged in to the status page	73

FIGURES

Figure 27	Employee login	74
Figure 28	Employee login	74
Figure 29	Employee functions	75
Figure 30	Employee edit user	75
Figure 31	Employee convert to manager	76
Figure 32	Manager login	77
Figure 33	Manager page	77
Figure 34	Edit users to manager	78
Figure 35	Edit full user information to manager	78
Figure 36	Manager decision	79
Figure 37	Decision done	79
Figure 38	SQL injection attack	80
Figure 39	SQL injection attack	80
Figure 40	SQL injection attack	81
Figure 41	SQL injection attack	81
Figure 42	SQL injection attack	82
Figure 43	SQL injection attack	82
Figure 44	Invalid login or attack	83
Figure 45	Mobile application	83

LIST OF ABBREVIATIONS

PHP	Personal Home Page
HTML	Hyper Text Markup Language
XSS	Cross Site Scripting
DOM	Document Object Model
URL	Uniform Resource Locator
SAAS	Software As A Service
SQL	Structured Query Language
3D	Three Dimensions
ASP	Active Server Pages
CGI	Common Gateway Interface
JSP	Java Server Pages
HTTP	Hyper Text Transfer Protocol
CSS	Cascading Style Sheet
SW	Software
AJAX	Asynchronous JavaScript and XML
IDE	Integrated Drive Electronics
GUI	Graphical User Interface
RDBMS	Relational Database Management Systems
JSON	Java Script Object Notation
SSL	Secure Sockets Layer
RAD	Rapid Application Development
HTTPS	Hypertext Transfer Protocol Secure sockets
BEA	Business Enterprise Architecture
Wi-Fi	Wireless Fidelity
UTF	Unicode Transformation Format
OS	Operating System
IOS	Internetwork Operating System

GB	Gigabyte
MB	Megabyte
SDK	Software Development Kit
IDEP	Idaho Dental Education Program
OOP	Object Oriented Programs

CHAPTER 1

INTRODUCTION

1.1. Background

Web application is one of the important services [1], which can be offered especially by Computer Science educational institution and in the world generally by communities where these applications can offer a lot of services for the users. Moreover, this kind of applications may convert local operations to global one by linking local institutions in the world by using (Internet), where all web users can access to various institutions around the world by making one click without the issue or significance of time, place and language. On the other hand, there are a lot of problems faced by users of these technologies especially the important problems of protection and viruses, which have become a source of inconvenience for all Internet users and websites designers. Here this thesis will offer Internet application used in the student's registration at Çankaya University protected by two of the security problems faced by users and website designers. The problem is shown below [2].:

1. SQL Injections.
2. XSS – Cross-Site Scripting.

To supplement this work, we designed the mobile application [3]. by using the IOS platform and this provides more services for students wishing to plan to register at the university, where smart phones users can register at the university by using this application on their mobile and that allows them to track follow the cases of registration in the event of pre - acceptance, in processing and when it is unacceptable.

1.2 The Literature Survey

In 2003, U. B. Landsman and D. Stromberg [4], *Web Application Security : A Survey of Prevention Techniques*. For analysis purposes, the classification additionally presents that attack strategies, stipulations, vulnerabilities and countermeasures it used the general criteria and security model to confront one existing technique. Additionally, it conferred gathered techniques and projected countermeasures into a matrix. The results of these processes disclosed weaknesses: not all hindrance techniques projected by authors is claimed effective, since completely different techniques may gift different countermeasures and also the authors' underlying discussions does not cowl all attack strategies, stipulations, vulnerabilities and countermeasures conferred in our model.

In 2003 M. Yoshihama, H.Tanaka [5], displayed the completion of registry of students which has been executed by Lector (OCR) which reads the registry form that the studier has submitted. In their paper, they present the system, which supports the process of a check of completion of registry and the content of the registry on the Web browser.

In 2007, E. Merlo, D. Letarte [6], an innovative approach, which mixes static and dynamic analysis, the re-engineering of code to protect the applications which are written in PHP language from SQL injection given attacks. PHP BB web application has been mechanically protected by the planned approach. The final results appear with a high rate of success and low range of false positives and negatives The approach shows that plan appears appropriate in the test PHP BB suit, however an analysis and assessment square measure is required in judging it on larger and additional diversity systems.

In 2007, J.Lin¹ and J. Chen¹ [7], Proposed and developed a tool that produces a suitable function for validation of an input which is depend on the DB server and application layout. The source code is needed for adding input functions to injection revise vulnerabilities. Also it presents a good adaptation of the security of application-level to solve problems, which cannot change the source code.

In 2008, J. Sulaiman, R.H. Mat [8], proposed an E-SAS to design a conversion for the manual way in the system of managing the data of student with the evaluation of the assessment. This proposed system can provide time for teachers to control the student assessment process. This computerized way can be better for data management and data manipulation. In it the main profile of the student is kept and

saved in a protected and complete database, which is compared to the previous system file.

In 2008, G. M. Wassermann [9], Displayed a general framework to learn and prevent input validation based attacks in a multiprogramming setting, viz. the Dominion of web applications. This essay explains the first formal, realistic characterization of SQL injection and XSS and presents principles, functional analyses for identifying vulnerabilities and stop attacks. The analyses can discover and block real attacks and uncover undiscovered vulnerabilities in real-world code.

In 2010-2011, B Bonné [10], displayed many methods of sit implantation attacks that are on the web now. It discussed how sitting identifiers can be accessed, and which properties a sheltered sitting identifier should own.

The users have to rely on the developers of all web applications from which they are using to accomplish the security at the server side, which solve this problem, it needs to improve a client-side solution to sit implantation attacks.

The proposed client-side solution supply security against both sessions, implantation and session hijacking attacks, where an attacker uses entrusted channels to access or qualify the session cookie, to complete this solution as an add-on to the Firefox Web browser.

In 2012, Y. Liu¹, F.Gao² [11], presented a Student Registration System for Universities (SRSU) as a type of management systems for information which not only register the students information as fast and effectively, but also make the statistics on the information about the student, registration, payment for student and output of the analysis results. Analyses were done first and then the development technologies were included.

In 2012, M. H. Abu Hamada [12], proposed a tool to be implemented by using Python language that is called a (XSS Detection). To solve the problem of client side vulnerabilities and reduce the Cross-Site Scripting danger with attacks, it uses two factors to evaluate it: performance and accuracy. When compared with (XSS Detection) other tools it gives 90.24% that satisfies and is acceptable to the need of users.

In 2012, D. Hauzar and J. Kofroň [13], proposed a new way to bugs discovery within web-based applications. Whereas based on some techniques by primarily merging them into one then making an improvement to cover almost all the issues, which are very critical. It proposed a model that is quite precise to detect weak points and is

capable to test most of the presented vulnerabilities in the literature.

In 2012, A. Manchanda [14], proposed in building this expensive pursuit software package tool, in which the major focus was to form this tool less user intensive and more user productivity. The primary version of this application is only applicable to the USA. It could be employed in alternative countries if currency converters within the application are used, that Manchanda improvised in the later version. Boundary problems were featured whereas implementing this tool and numerous vital things were unbroken in mind. Core Data was chosen over SQLite to persist the info that is incredibly helpful although the info resides on the device domestically.

In 2013, R. DacuycuyPacio [15], presenting on this study, displayed the following points:

1. An existent student information system in Benguet State University is made manually by using paper-supported action.
2. Problems usually experienced at the existent Student Information System were ineffectual, errors, and costly saving of the data of the student.
3. Check measures and security that are required to be adopted in the student information system of Benguet State University was limited of safe password accessing into the system, access level to accredited users, physical server protection, and trail of audit.
4. Profit of an online student information system would be able to maintain the cost effectiveness as well as managing information of student for Benguet State University.

In 2014, M. Sp. ngmyr [16], proposed a mobile application which was successfully developed, that it was absolutely potential to use for its meant purpose which EDCA and every element of the implementation was free below i.e. an ASCII text file license, so like this the most important aim of the project was reached, with certain reservation. The research began the bottom map supplier, including the API employed in EDCA to access it, is an exception that's not free as ASCII text file, but that was chosen as a compromise owing to the quality of its map content and the less long development.

1.3. Aim of the Thesis

This thesis goal is to design and implement a modified security approach for web-based system with smartphone application for student's registration in Natural and Applied Science Institute of Çankaya University for the following purposes:

- 1- Provide the ability of the online registration process.
- 2- Archiving student's information.
- 3- Provide sufficient capacity to facilitate the work of institution's employees.
- 4- Provide ability for students who wish to register.
- 5- Provide a data privacy by making a level of security.

1.4. Thesis Layout

The thesis layout is as given below:

- Chapter one presents the background and literature survey of secures web-based system and smartphone applications.
- Chapter two includes web applications and security as well as their definitions, benefits, uses, and properties as well as Smartphone applications in IOS platform.
- Chapter Three contains design and implementation of the proposed system for student registration.
- Chapter four includes the work results.
- Chapter five includes the conclusions and the recommendations for future work.

CHAPTER 2

WEB AND SMARTPHONE APPLICATIONS, SPECIFICATIONS AND SECURITY

2.1. Web Application

2.1.1 Introduction

The web-based application is computer application software, which typically operates in a browser that is created in a web computer programming language such as the combination of JavaScript [17]. CSS and HTML and relies upon a web browser to make it the application [18].

Web applications tend to be well known due to Internet browsers' ubiquity, having the convenience using the Internet browser as a client who is occasionally called a "thin client". The capability to provide new updates as well as maintain web applications without the need for installing and distributing computer software relating to a many thousands of clients computers which is basically a key point cause for their acceptance and popularity, as well as the intrinsic full support fit for cross-platform assist. Typical Internet applications involve web mail, online commerce sales, on-line auctions, wigs and several additional capabilities [19].

2.1.2 History

In much earlier computing types, for example in the whole client-server, an application load happens to be contributed between code upon the web server and code set up on every browser. Therefore, an application took its private program for the client, which actually served as its own user graphical user interface in addition to the need to be installed separately on every user's computer. A particular upgrading into the server-side script of one's application could be typical in addition, will require an up-grade in the client-side code set up on every user's workplace,

contributing to the cost support and minimizing the productivity. Additionally, both of the client and web server parts of the application was also frequently securely bound to a certain computer structure as well as operating systems while porting them into many others was also in most cases prohibitively more costly for all those except the major applications.

Contrastingly, web applications applied web site documents wrote and published within a typical data format such as HTML in addition to JavaScript, which could be established by various Internet browsers. Web applications can possibly be regarded as a particular type of client-server program in which the client application software definitely is downloaded into the machine of the customer once choosing the considerable web-page, making use of the usual procedures for instance HTTP. Consumer web- application software up-dates could happen each and every time the web page is visited. Through the session, the web explorer interprets and certainly displays and shows the pages, as well as can be the primary customer for every web application.

The web in its early days of every individual web-page appeared to be handed to client just like a static document, however the sequence of web pages could actually remain as a flexible experience, like user input also came back via internet form as basic elements established in within the page markup. Nonetheless, almost every note modifies onto the web page necessarily a makes round trip returning to the web server to refresh web page content.

In 1995 presented a client-side scripting language, which named JavaScript, which enabled programmers to include and add many dynamic components and elements for user interface, which ran on the client- side. Therefore rather than giving and sending the information for web-server to be able to make a complete website, the stuck scripts of the saved page can do numerous jobs for example validation of input or show/hide some areas of the page.

The Macromedia was presented in 1996, which display, a vector movement player that may be put into web-browsers as a plug-in for adding the animations on the Internet. It permitted the utilization of a scripting-language to program the relationships upon the client without any need to make a communication with server. Web application main concept was presented in the Java language in Servlet Specification version 2.2 [20, 21]. During those times each XML and JavaScript had been already developed, however the Ajax had not even been coined and the object

of XMLHttpRequest had just recently been presented on Web server5 being an ActiveX object [22].

In the year 2005, the definition of Ajax was presented, and many applications such as starting the Gmail to make their clients more and more interactive. A web-page script can be contacted with the server retrieving and storing all information without accessing an entire page of Internet.

In the year 2011, HTML 5 was completed, which supplies graphic and a multimedia functions without the necessity or need of the client to use plug-ins. HTML5 also gives the documents of semantic contents. The APIs and Document Object Model (DOM) are no more afterthoughts, but the essential specifications parts of HTML5. WebGL API that paved a way, in which for an advanced 3D design, which is based on the canvas of HTML5 JavaScript language. These have an importance in producing real platform and the browser rich web applications.

2.1.3. A Web application interface

Through the JavaScript, Java, DHTML, Flash Silverlight and many technologies, application-specific strategies for example drawing on screen, playing an audio and usage of mouse and the keyboard are all typical possible. Several services have been employed to combine many of this right into a common interface, which adopts an appearance of the operating- system. Basic function for example moves and decline may also be supplied and supported by these technologies. Web designers usually use client-side scripting for adds functionality, particularly to generate an interactive knowledge that does not need page to make a reloading. Technologies recently have been produced to coordinate client side scripting with server-side for instance PHP. Ajax, the web progress and development method applies a combination of different technologies, is a typical example of technology which generates an even more active experience.

2.1.4. Core structure

Applications are generally broken into logical portions, which are called a "tiers", where each tier is given a role. Standard applications that consist of just one-tier live on the client unit, but the web applications themselves can be provide an n-tiered method by nature. However many modifications are probable, the most typical structure could be the three-tiered application. In their most typical form, the three of

tiers are named presentation, storage and application, accordingly. A browser is the initial tier; an engine with a use of dynamic content for web technology (like the ASP.NET, ASP, CGI, JSP/Java, ColdFusion, PHP, Perl, Ruby, Python) is the center tier (application logic), the database, which is the 3rd tiers, is a storage. The Internet browser directs and sends demands to the center tier, which can make services by creating updates and queries from the database and creates a user-interface. For more complicated applications, a 3-tier alternative solution may possibly fall short, and it could be useful to make use of an n-tiered method, where the greatest gain is breaking the logic of the business that exists on the tier of application into a very fine-grained design or model. Still another good benefit might be putting an integration tier, which divides the data information tier from rest of tiers by giving simple interface to get into the data. Like, a client information could be reached by calling the function "list_clients()" instead of creating a query in SQL directly from the client on the database. This enables the main database to become replaced and changed without making any other tiers [23].

There are several who see a web application as a two-tier architecture. That make a client as "smart" to perform an overall of the work and make queries a "dumb" of server, or perhaps a "dumb" of client who which utilizes a server that can be "smart". The client might manage the tier presentation; the server might have storage tier of the database, where an application tier of the business logic could be on one or on each one. While that advances the scalability of applications and divides the show display and the database however, it does not permit true specialization of layers; therefore many applications can outgrow that model [23].

2. 1.5. Scope of business

A strategy for the companies of application software is to supply web access to computer software that previously distributed as "local application". Depending on the application kind, it could require the progress of various browser-based interface, or just adapting a current application to make use of different technology of presentation. These programs can allow the user to pay for a regular every month or every year fee for utilization of a software application and never having to do the installation on hard-drive. A company, which uses that strategy, is recognized as "application service provider – ASP", and ASPs now obtaining much attention in an industry SW.

The security on such applications really is a significant concern since it can include both enterprise data with the private data for client. The protection of these assets is an essential part of any web application and there are several important key operational parts and areas that must definitely be contained in the development method and process [24]. Including techniques for authorization, authentication, asset managing the input and recording logging and auditing. Developing security to applications right from the start could be more efficient and less disruptive in the long running. In web applications the processing model for it is software as a service (SaaS). The business of applications can be presented as SaaS for enterprises for usage or fixed dependent fee. Other applications in web are given a free of charge; more often generating an income of money from the advertisements is shown in an interface of web application.

Several businesses can make by opening a web application source for example e-commerce software that simplifies producing online store retail. Today many businesses do not need to get or buy hardware of data center for example the servers since they are affordable for a brief rent and short expression that forms a basis for various hosting organizations that provide a turnkey of web application implementations. It is popular for hosting suppliers to also provide hardware and all required computer software to supply all needs to the business company. The innovations in most of the aspects of the web applications can provide a great economic value by making an increase in the competition by reducing the barriers to access for new companies.

2.1.6. Writing the web applications

The development of applications writing is often simple by open -sources software for example Drupal, Django, Symfony named web application frameworks or Ruby on Rails. These can rapidly facilitate a development of applications by allowing a team for making this development to focus on the components and parts of these applications which are unique goals and which never have to handle common issues for this development for example the user management [25]. While a number of these frameworks are open- source, which certainly are not a requirement.

The web application utilization frameworks may usually reduce the numbers of problems and errors in the program, Both by letting a team to focus on the framework, and by making the code easier, while another that focuses on a certain

use case. The applications that are confronted with continuous hacking on the Web, they can have related security problems and which could be caused by program errors. Frameworks also can improve the use of most useful and better practices for example GET after POST. Furthermore, a potential for application development of on Web operating systems, while currently there are a very few viable systems and platforms, which fit that model.

2.1.7. Some applications

Examples of application browsers are very simple software office for word processors, (online spreadsheets, word processors and tools for presentation), but also can contain more advanced and complex applications for example task management, design of computer-aided, point-of-sale and video editing.

2.1.8. Web application benefits

1. Web applications do not need any complex or complicated technique to deploy in big organizations. An appropriate web browser is all that is needed.
2. Web browser applications usually need little or no disk space on the client.
3. Web applications might require no update or upgrade method since new features are applied on the web server and immediately delivered to the end users.
4. Web applications include easily integrate another server-side Internet procedures for example searching and email.
5. Additionally they give cross-platform compatibility generally (i.e., Mac, Windows, Linux, etc.) since they run within a browser window.
6. With an introduction of HTML5, programmers can make richly active environments within browsers. With addition to improved error handling the set of new characteristics and features are native video, audio and animations.
7. Recently, web applications support great increase and great interactivity by technologies usability for example AJAX that successfully exchanges the data between the server and the browser.
8. Web applications can be allowed easy introduction of new devices for users (for example tablets, smartphones) because of their integrated built-in browsers.

2.1.9. Web application development

Tools of web development can allow the developers to check and debug their code.

But they are different from web site builders and IDEs for the reason they do not support in the direct development and creation of a website, relatively they are methods and tools that require for the user interface of a web application or website. Tools of web development come as browser built-in features in web browsers. The most popular Internet browsers nowadays like, Chrome, Google, Firefox, Opera, Web Explorer, and Safari [17]. have integrated built in tools to simply help the developers, and additional add-ons are found within their particular plug-in centers of download.

Tools of web development can allow the developers to utilize a number of web systems and technologies, including CSS, HTML, the DOM, JavaScript, and others, which are treated by the web browser. As a result of increasing need from web browsers to accomplish more [18]. Common web browsers have included many features targeted for developers [19].

2.1.10. Web languages

2.1.10.1. PHP

PHP is a server-side scripting language, which is made for the improvement of web and is also applied as a general-purpose programming language. Initially developed in 1994 by Rasmus Lerdorf, Originally PHP is stand for “Personal Home Page” today PHP stand for “Hypertext Preprocessor” [23].

PHP code is treated by the PHP -interpreter, which is implemented as a web server's module or as a Common Gateway Interface (CGI) that could be compatible. When the PHP code is elucidated and performed, Web server sending the executable output to its client, often in kind of a part of a created web page, PHP code could create and generate a web HTML pages code and generate an image or any other data [26].

It is really a general-purpose programming language, which is suitable for the web development of web server; generally PHP can be run on a server side or web server. PHP code in a requested code or file is performed by the PHP runtime, often to generate potent content of web page or images utilized on websites or anywhere [27]. PHP can also be used for command-line scripting and client-side Graphical User Interface (GUI) applications. PHP can as well be implemented on more servers, several platforms, and operating systems and may as well be used with most relational database management systems (RDBMS). PHP works mainly as a filter,

[28]. getting an input from the file or text stream or/and instructions of PHP and outputting another data stream. Generally the result of output will be an HTML, while it could be XML, JSON or binary data for example audio or image formats.

Originally developed to generate impressive web pages, PHP today focuses mostly on server-side scripting language, [29]. And furthermore it like server-side scripting languages can facilitate a dynamic page content from a web server onto the client for example Sun Micro systems Java Server Pages, Microsoft's ASP.NET. an [30].

PHP in addition has attracted good development of many computer software frameworks, which contributes in building blocks and a structural design to boost Rapid Application Development (RAD).

2.2. Web application security

Web is the recreational ground of 800 million Internet citizens, residence to 100 million of Web sites and mediator of billions of dollars on a daily basis. As a universal influencer, web made global economies have transformed into reliant to web. The time has passed very quickly as Web mail, messaging and chat platforms, auctions, news, online shopping, banking, and other Internet-based software have placed a significant role in digital existence. Internet users deliver all of their personal information such as addresses, social security numbers, credit card information, telephone numbers, mother's maiden name, yearly income, date of birth or their favorite color to take fiscal accounts, day trade stock, or tax records [31].

Internet sites have enormous security subjects taking this data at risk. Systems are intimidated by lately security warnings only recognized as Web Application Security, after title used to identify the paths of ensuring internet-based software.

The structures, which compile individual and special data, are accountable for guarding it from curious eyes. We have to consider widely as the web app security goes bigger. We are in front of the comparative irritations of phishing, code kiddy defacements, and overall revelation antics. Current internet sites release command state-wide power grids, utilize hydroelectric dams, occupy prescriptions, execute pay sheet for the exterior of institutive America, function institutive networks, and execute other solely crucial occupations. Assume that what an unfavorable compromise of any of these systems might orient towards. It is not easy to picture a field of information security, which is extremely crucial. Web apps have transformed the clearest, straightforward, and relatively the most manipulated path for system

compromise.

Till lately, everybody considered firewalls, SSL, invasion-spotting systems, network scanners, and pass codes as the solution to network security. Security experts took over from simple armed forces tactic where one establish an edge and protect it with all of the things one has. The strategy was efficient, that is till the web and e-business eternally altered the aspect for the exterior part. E-business demands firewalls to permit in Web Hypertext Transfer Protocol (HTTP) and Hypertext Transfer Protocol Secure sockets (HTTPS) intensity.

Web formatives are now in charge of security and forming apps that need to be altered basic software design methods, which empowers Web business.

Formatives now build software, which operates on Internet-attainable Web servers to supply services for everyone, everywhere. The range and extent of their software conveyance has risen incrementally, and the security problems have also expanded. Today, hundreds of millions of users have straightforward attainability to institutive servers globally, which might be an evil opponent.

Current titles like cross-site scripting, Structured Query Language (SQL) injection, and tons of other current basic Web-enabling attacks need to be figured out and handled. Web application security is a huge subject that consists of lots of fields, techs, and design methods of servers like JBoss, IBM Web Sphere, BEA Web Logic, and a dozens of others. Later, there are the merchant and an open source web applications such as PHP Nuke, Microsoft Outlook Web Access and SAP. Other than these, there are some inherent personalized Web apps, which institutions build for themselves. This is how things are outlined in terms of understanding of Web app security. One of the major risks that web app formatives need to comprehend and manage is whether how to moderate XSS attack, as XSS is a comparatively minor section of the web application safety area, it demonstrates the as the most risky for a normal internet user. Just one little badger on a web app might cause a hazardous browser through which an attacker is able to take away the information, absorb a user's installing background and furthermore. Yet still, lots of people do not fully comprehend the risks of XSS weaknesses and how to accustom to attack injured parties. This paper's primary purpose is to inform readers via arguments, samples and representations as to observe the actual risk and vital impression that XSS might possess [32].

2.2.1. Kinds of web applications attacks

It is possibly not invisible that any sort of web app, which compiles data through users is weak to machine-controlled attack. It might not be visible to that websites, which inactively transmit data to users making it even more vulnerable and weak. There are three types of weaknesses.

2.2.1.1. When the users provide the information

The most mainstream types of web apps permit users to go into data. Afterwards, that data might be reserved and regained. The users are anxious at the moment, yet, basically with the information, assumed to be unobjectionable, which people write in it [34].

2.2.1.2. Human attacks

People are competent enough to utilize technology in both dangerous and practical paths. While you are generally not officially in charge of the activities of the people who perform your online apps.

In addition, in operations, to handle dangerous users might occupy a crucial measure of resources, and their activities might make an enormous injury to the position of the site that you have tried very heavily to build. The exterior part of the under mentioned actions might be thought irritations instead of attacks, since they do not contain a real violate of app security. Yet these gaps are still violations of regulations and of the social agreement, and in a way that the programmer may intimidate them they are luminary of discussion in here [35].

1. Abuse of storage: Most of the web sites permit their users to remain a diary or send photos because of the stylishness of internet-blogging and message panel systems. Web sites like those might pull abusers who require reserving, without being afraid of that, which it might follow to their own servers, not because of diary access or photos but instead of unauthorized or provocative substance. Or abusers might basically require empty memory space for big amounts of information. If not, they need to compensate for that.
2. Sock puppets: At any site which entreats user views or back indication is weak to the perfectly called Sock Puppet Attack, where one physical user enters below a deceptive false name or numbers of various false names to

swing views or stuff a ballot. People who post misleading comments on Amazon.com are involved in sock puppetry; so are irritable parties on message panels who form various accounts and utilize them to form the impression of broad-ranging promotion for specific views. As this type of attack is more efficient when it is machine-controlled.

3. Lobbyist groups are well-known non-digital samples of the Sock Puppet Syndrome. Few of these are recently forwarded into the digital area, appearing with flattering names and intending to propose neutral view, whereas covering highlighting the institutive and funding ties which modify assumed data into political distinctive requesting. This the expanding trend to browse free civil Wi-Fi networks possesses. To illustrate, it has delivered series of ‘research institutes’ and ‘study groups’ combined in defiance to rivalry with the for-profit telecommunications sector.
4. Defamation: Connected to sock puppetry the attacker’s utilizes ones app to post malicious staff regarding various groups and people. By posting as an unknown user is normally not an issue; the poster’s unknown identity demotes the possibility of its being accepted, and no matter what, it may be deleted if there would be an uncovering. Yet a suitable posting with your name, even it is deleted as quickly as it is recognized, might intend that you are going to need to verify in justice (or leastwise to your Board of Directors) that one was not the generator of the post posted. This position has proceeded so far even most of the lists are posting authorized refutations and cautions for any possible abusers at the forefront on their lists.
5. Grievors, trolls, and pranksters: The users who are identified as griefer or trolls or pranksters are more irritating by an element of 10, can instantly be controlled by joining an online community, even if they are not very vital as the vicious liars mentioned prior. Grievors contain users who like to attack users. The bullies one sees as a fresh user in any sort of online role-playing game are grievors who take cover by using nicknames, might be brutally vicious. Trolls, in other respects, like having fun by just attacking. Such people post horrendous affirmations and post-absurd opinions in order to arouse your interest; even so it is not positive. Pranksters can inject HTML or JavaScript guides into what documents need to be plaintext, to twist page display; or they may not act like themselves; or they may understand various

ways to divert from what had been meant to be strict business. These users damage a group by pushing concentration away from opinions and onto the characteristics of people who post these [36].

2.2.1.3. Automated attacks

In order to broaden human control, attacks in that stage manipulate the power of computers. These coded attacks or automatons decelerate services, put in error logs, feed bandwidth as well as pull other harmful users by publicizing that the site has been agreed on. They are especially risky because of the effectiveness of themselves.

1. Viruses and worms: It is possible that the worm or virus is the most disreputable and spectacular little program which browses itself onto computer without users information. It is installed by links that are within the e-mails or by involvement into an installed app. There is a slight variance among the worm and virus; A worm manages to remain by its own, on the other hand, virus needs to piggyback onto a manageable or text file. The main objective of a virus or a worm is to copy itself by distributing to other tools. A junior-grade objective is to destroy its server computer, removing or altering documents, sneaking to backdoors (that strangers may be accustomed to, to illustrate, forward span through the device), or instantly make messages appear of different kinds. A worm or virus may release on its own via digital on line means very quickly when it operates a broad-ranged weakness [37].
2. The Spam: It is the forwarding of undesired messages in enormous amounts. It is a machine-controlled attack in a distinctive way, since it looks normal, even extravagant in its utilizations. It acquires short for users to be educated to know spam (at least the exterior part of spams); it acquires servers (that holds the difficult part of the transmission) for considerably prolong time. Yet spam results both being exposed to an undesired charge of service.
3. An automated user- input: Are another types of attacks, which control the supplying entry (assuming from users) in different settings.
4. A Group operating Internet portal services may determine to get attention of users by proposing costless services such as e-mail accounts or offsite memory. These services are quite charming to authorized users and abusers who are , for instance, are able to operate costless e-mail accounts to execute

spam.

5. Common interest groups may form a web app in which users are permitted to reflect their choices by prospects and problems for an expected designation. Such group means to allow people using it' reflected choices counseling common views regarding which issues show better approach than the opposites, and which issues show or capture the public. Surveys like these are a standard objective for a harmful group or person, who can originate a machine-controlled attack to assign enormous number of votes for or opposing a specific nominee or issue. Rigging the elections like this, could originate a false image of public's views.
6. A group can form a website to attend liking to a brand new and costly product. For example, an electronic device, a car, almost everything. It may determine to arise curiosity in the new product by arranging raffles, where a new product is handed out to a random person who enters the raffle.
7. It is quite common for specific types of web applications, which supply potential users to write views or messages in a chat group or in visitor's book. It can seem innocent to fill up meaning in these sort of positions, because the entry does not look depending to the original value. Yet, as a matter of fact, messages that include slight or none except connections to a website can create a crucial complication, for they may expand tremendously the website's search engine sorting, that possess quite clear value. Machine-controlled mass answers that exploits of a system, which prevails, in another manner, for the public wellness, even if not with this fiscal approach.
8. A similar prospective weak places in any sort of the website enrollment is necessary, even if there are no costless services available. It might look like that there is a slight dot in an attack which enrolls 10,000 imagined names for associate ship in a group, but a person is not able to generalize that abuse as innocuous [37].

2.2.1.4. When the information is specified to users

This may appear as if founders of a web application that's job is to yield certain information to users would be pleased that information is truly yielded. Yet given the usage of such information might occasionally be placed, consuming the information often not by force, particularly when it arouses being consumed machine-controlled

proceed.

1. Harvesting an email addresses: It is not unique for web sites to contain an e-mail address. Considering that e-mail is more extensible than a form, businesses might select to propose users the probability of communicate via e-mail, instead of form. Individuals and groups of several types are going to supply e-mail addresses accurately since they want users to converse straightforward and easily with core specialist. Websites like these are easy objectives for machine-controlled harvesting of e-mail addresses. Gathered lists of such addresses are commercialized to people who spam and other mass e-mailers, and e-mail messages that are compiled from these scrounged lists which build a vital amount of an Internet intensity.
2. Email address flooding: Frequently, a website demonstrates an exclusively made e-mail address prepared just for attaining user addresses, generally like `contact@something.org`. In this regard, harvesting is not similar to a simple flooding of a singular e-mail address. An instant illustration of server e-mail logs demonstrates how peak a percentage of e-mail messages to these addresses contain spammers' proposes of inexpensive house deals, sexual accessories, bank accounts of Nigeria as well as others.
3. The screen scraping: Commercial websites which frequently conduct certified or personalized information presented to all workers of the commercial, who can be extensively spread geographically. In another way, it is not available to attain the information personally. Machine-controlled attacks may involve in screen scraping, basically getting all the information out of the screen and afterwards identifying what has been covered for the things of liking to the attacker: for example, business outlines and things information.
4. Alternating, the attackers may be curious about operating screen scraping is not too much for the evident component of a website page as of the information included in URIs and document titles. Knowledge of such a kind that may be identified with perceptiveness in the design as well as group of a commercial's web applications, ready to release more of an in-depth attack for upcoming days.
5. Improper archiving: Search engines are rare considerations of machines. Controlled abusers who use commercial websites consist of temporal

constraint information, exclusive proposes or membership component filing cannot be thought suited. They might be conducting invalid information accessible as if it were valid, or demonstrating exclusive prices to a broaden target market than it was thought, or supplying information costless that other people needed to pay [38].

2.2.2.Injection attacks

The lists of injections and the Cross-Site Scripting (XSS) as the most mainstream security dangers to we apps. Certainly, they pass around since XSS attacks are dependent on a prosperous Injection attack. As this is the most visible association, Injection is not restricted to permitting XSS [35].

Injection is a whole stratum of attacks that depend on injecting information to a web app to assist the management or paraphrasing of harmful data in an unwelcomed approach. Cross-Site Scripting (XSS), Header Injection, SQL Injection, Log Injection and Full Path Disclosure are the samples of these attacks in that stratum.

They are the most extensive and accomplished attacks on the web because of their several kinds, wide attack area, and the complication occasionally had to defend counter to them. All apps require information from someplace to operate.

2.2.2.1. A Cross Site Scripting (XSS) attack

2.2.2.1.1. Introduction

When an application that makes process to the input without verifying and validating it is possibly vulnerable to the code injection because one client could present the output of another client, which can cause vulnerabilities to attacks. When the injection code is a scripting for example as standardized a JavaScript in it is known as Cross Site Scripting (XSS).

An HTML code that embedded into some posted message on a susceptible bulletin board for example, having an exploit as shown in Figure 1. The successful attack steps are shown in Figure 2 [39].

Firstly, Attacker writes and stores an XSS message on a bulletin board that is vulnerable. The victim authenticates the bulletin board and is identified by a cookie, which is set in the browser. Now the victim needs to request an attacker message to read it. The malicious code is delivered back within the message that is performed by the web browser. The XSS program can send this stored cookie to the attacker. Then

the attacker can recognize and identify all information by gaining the privileges of the victim with the session cookie. The Reflected XSS” process works on the link that contains the malicious code as shown in Figure 1. The executed steps for this attack is shown in Figure 2. This example considers that, the victim firstly authenticates himself at the susceptible web. The attacker can send the hyperlink to the victim included in an email, or a message of a newsgroup that contains the link. When the user make clicks on the hyperlink, the Internet page is sent back by the vulnerable application that contains the HTML code from the hyperlink. The script that included the code is then performed by the Internet browser and the cookie will be transferred to the website of the attacker. Additionally, the attacker, which can use the session cookie of the victim to make an authentication to himself as the victim to the vulnerable Internet application and gets all privileges of the victim. Many attacks may be prevented by the same-source origination policy which is incorporated some scripting safety and secure models. This strategy can prevent loading of a document from a web site may have an attacker who can be includes with a malicious code. A security design is bypassed as the host variance is no further probable and the script is performed in the safety of the web page.

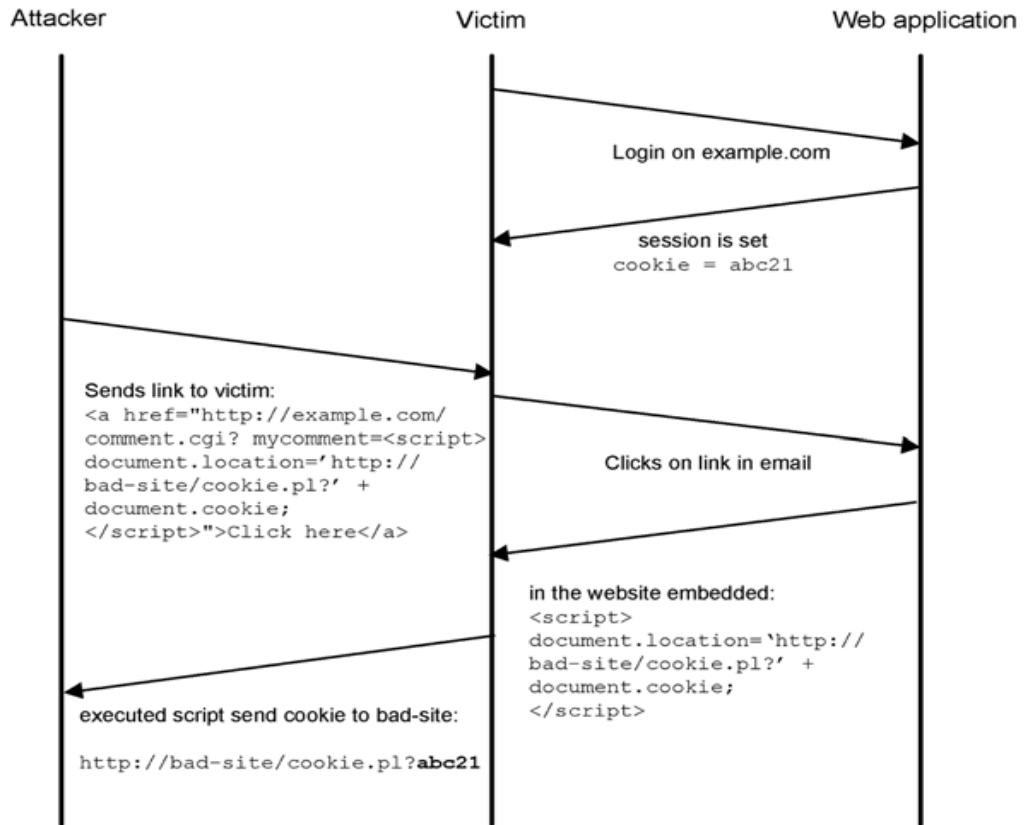


Figure 1 A stored message with XSS attack

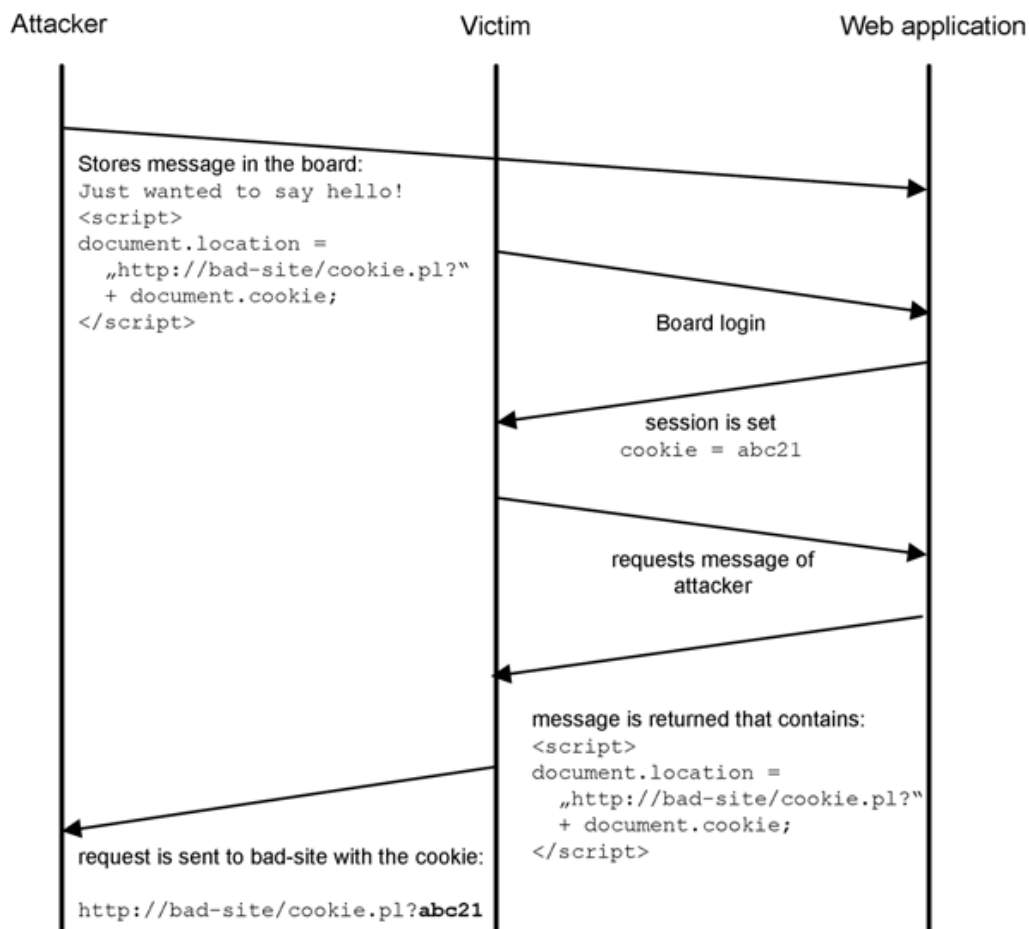


Figure 2 Some steps for XSS attack with reflection

<A HREF="http://example.com/comment.cgi? mycomment=<SCRIPT>
Alert (`xss`); </SCRIPT>"> Click here

Nevertheless, within a XSS attack, a web script has access to data inside the context on the document this is certainly an attack. When the attacker contains the malicious scripting code into hyperlink, the security design can possibly become bypassed as the web hosting distinction is no longer able and also web script is actually performed in a security context in the website page. Shows that people using must break the performance of languages that scripts into the Internet browser. Although actually, this specific option can affect each website page if those are exposed and sensitive also maybe it cannot decrease the efficiency of the website page to the point in which it can not be usable in any way. For example, website pages which can be utilizing AJAX (in other words, combining methods improves interacting with each other within website pages) depend on JavaScript [40].

2.2.2.1.2. Preventing XSS

Mitigating the actual problem is accomplished by establishing the character encoding for each produced webpage [35]. These specific character markings the start of an HTML tags. The attacker is capable of easily injecting code which makes use of a UTF-7 encoding of "<". The particular code is actually sent to the internet web application and in case the symbol belongs to the result of a webpage without having a pre-specified character encoding, the receiving internet browser may possibly translate that particular encode such as the accepted ASCII encoded "<". This because Internet browser makes use of UTF-7 like the normal character encoding. The HTML typical enables that the web browser comprehend tags inside alternate character encoding so does not really establish the character encoding injections of HTML tags.[41].

Most of the specialized characters (for example., "<", ">", "&") have to be known also and encoded when they might be involved into particular output, or even possibly they must be checked through the web application. Alternatively, simply injecting scripting code can pass by security mechanisms inside the web application. Checking characters ought to be done using a white-list, i.e., just characters identified to never create issues processing them and that will be encoded during the output when allowed. To exemplify, the birth year just requires digits. Thus, no alphabetical characters are required. Whenever they are countered with, some mistakes must be shown, as well as whatever not necessarily becomes a digit must get deleted out of this particular input. This approach will be much more safety than attempting to delete just characters from input which may be used in a XSS attempt for example, delete every possible tags of HTML such as page is sent to some other page, web application may not be sensitive by a specific kinds of attacks only if one of pages of server has some potential problems for security. For an example, a page can read some input and store it to some field of database.

Here static analysis can tell that the page includes vulnerability. But another page, which can read the field of data that make, encodes of the page output and in general the web application is not vulnerable. In a dynamic examination, executed many attacks that against the web applications. Either for a particular web application a database with generated attacks can be utilized or perhaps a database that contains some general attacks. More specifically, the server pages that can be vulnerable by a static analysis are tested in a dynamic test with some attacks for the sensitivity. A

crawler explained makes test of the black box that has a common database. It explores the created web application pages and then performs selected attacks. This approach checks web applications without interaction of user and reads web application response to such attack. Software analysis is really a good approach to reveal some of vulnerabilities. However necessarily for the static analysis to work the source code, dynamic tests allows, checking code with no need of source code however, it can just only test the obvious. The test of a website approaches such as a crawler utilized by the writers who need to analyze the Internet page and might not discover every optional input which may be used for attacks. XSS is very risky and its strictness is extreme, it might alter the website DOM and might guide to taking away certification of the executives, due to these circumstances the attacker may command and compromise the overall app [41].

What possibly the attacker get or be successful in getting?

1. Altering the Setting
2. A Cookie fraud
3. The Deceptive Publicizing
4. Steal a Form Tokens

2.2.2.1.3. Types of XSS

There are actually three kinds of XSS

1. The Persistent (Stored) XSS

The attack is stored on the server of website.

2. A Non Persistent (reflect) XSS

The user has to go over a special link to be exposed to it

3. DOM-based XSS

This problem exists inside the client-side script

2.2.2.1.3.1. A persistent (stored) XSS

The insistent (preserved) XSS weakness is a more harmful type of the cross-site scripting fault: it happens once the information is supplied by the attacker which is preserved by the server, and after that perpetually shown on ‘normal’ pages reverted back to other users in the stage of routine installation, without accurate HTML escaping. A standard illustration of that is along with the online boards

message where the users are permitted to send HTML designed messages for other users to scan. Basically, persistent XSS happens at the time when the formative reserves the user input data in to database server or filling it in a document without an accurate infiltrating, afterwards forwarding them to the client browser once more.

2.2.2.1.3.1. A persistent (stored) XSS

This PHP code as shown in Figure 3 which suffers from Persistent (Stored)XSS:

```
1 <?php
2 if(isset($_POST['btnSign']))
3 {
4     $message=trim($_POST['mtxMessage']);
5     $name=trim($_POST['txtName']);
6     // Sanitize message input
7     $message = stripslashes($message);
8     $message = mysql_real_escape_string($message);
9     // Sanitize name input
10    $name = mysql_real_escape_string($name);
11    $query = "INSERT INTO guestbook (comment,name) VALUES (
12    '$message','$name')";
13    $result=mysql_query($query) or die('<pre>'.mysql_error().'</pre>');
14 }
15 ?>
16
```

Figure 3 A Persistent (Stored) XSS PHP code

A two measurements ‘message’ and ‘name’ in that script are not cleared out accurately. We keep these measurement in the visitor’s book chart, so when we are showing these measurements right back to the browser, it can harm the JavaScript code in this situation as shown in Figure 4 and Figure 5.



Figure 4 A Persistent (Stored) XSS

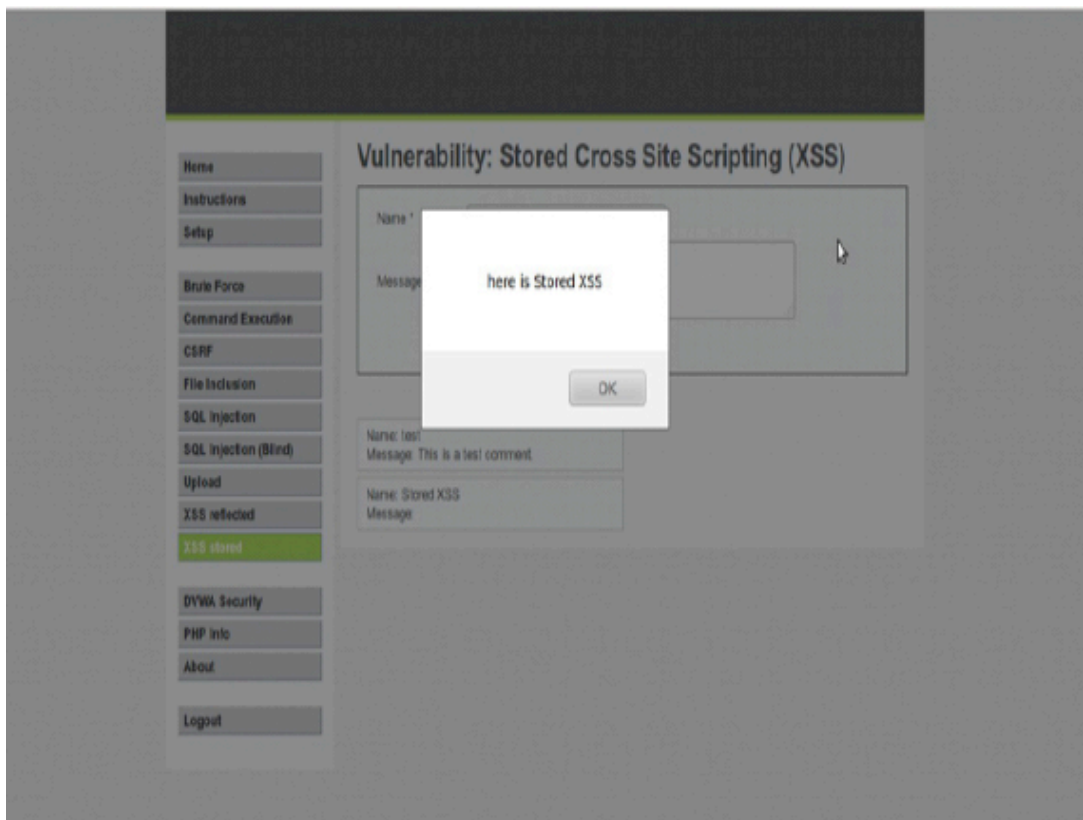


Figure 5 A Persistent (Stored) XSS

2.2.2.1.3.2. A Non persistent (reflected) XSS

The XSS non-persistent (reflected) weakness in the utmost mainstream kind. These hollows appear when the information supplied through a web user, generally in HTTP inquiry measurements as well as in an HTML form entries, are utilized instantly by the server-side codes which yields a page of consequences for that customer without accurately clearing out the demand [35].

2.2.2.1.3.3. Non persistent demo (reflected) XSS

As shown in Figure 6 the PHP code which suffers from Non Persistent XSS.

```
1 <?php
2 if(!array_key_exists("name",$_GET) || $_GET['name'] == NULL || $_GET['name']==''){
3     $isempty=true;
4 }
5 else{
6     echo '<pre>';
7     echo 'Hello' . $_GET['name'];
8     echo '</pre>';
9 }
10 ?>
11
```

Figure 6 PHP code suffers from Non Persistent XSS

As it can be comprehended that the ‘name’ measurement does not clear out and reminds of something to the user, because it will manage when the user injects a JavaScript. Here it can inject and can be harmful for the script.

As an aim to illustrate, here is injection `<script> alert (/XSS/)</script>` as shown in Figure 7 and Figure 8.

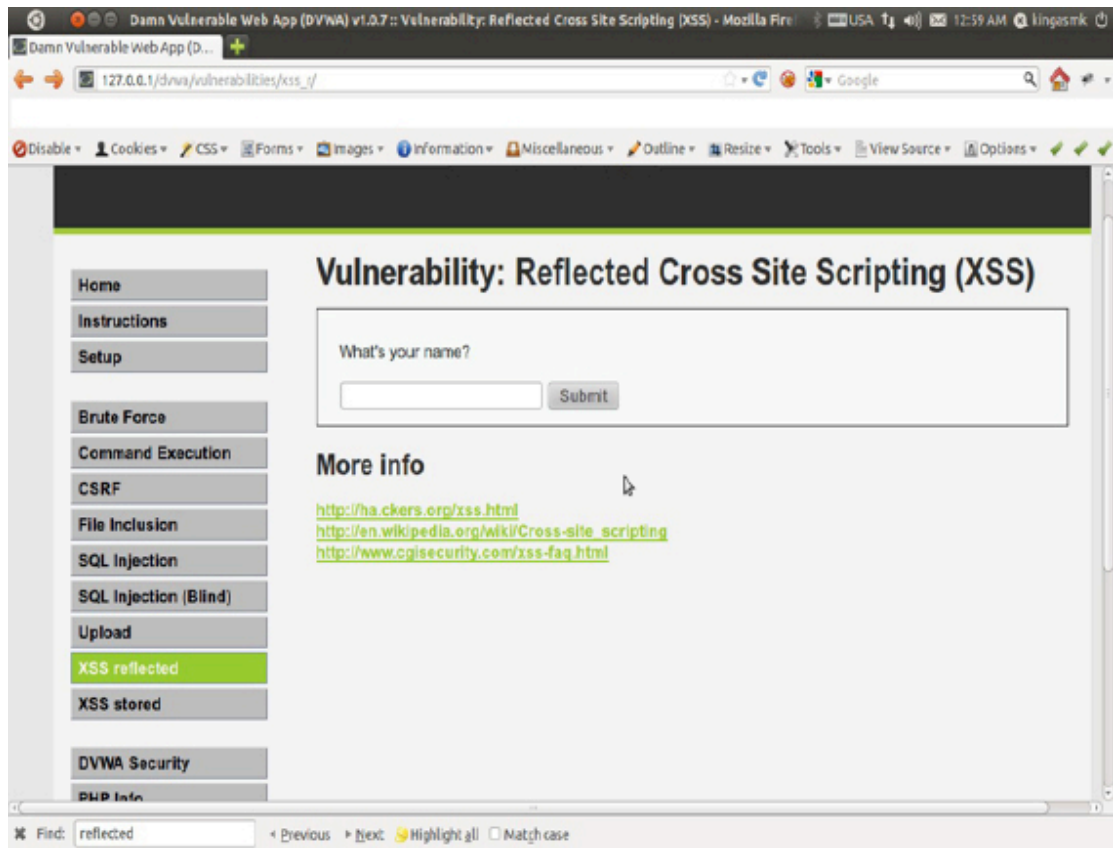


Figure 7 Non Persistent Demo (reflected) XSS

Here the application will inject an alarm in a box code “<script>alert ("xss")</script>”.

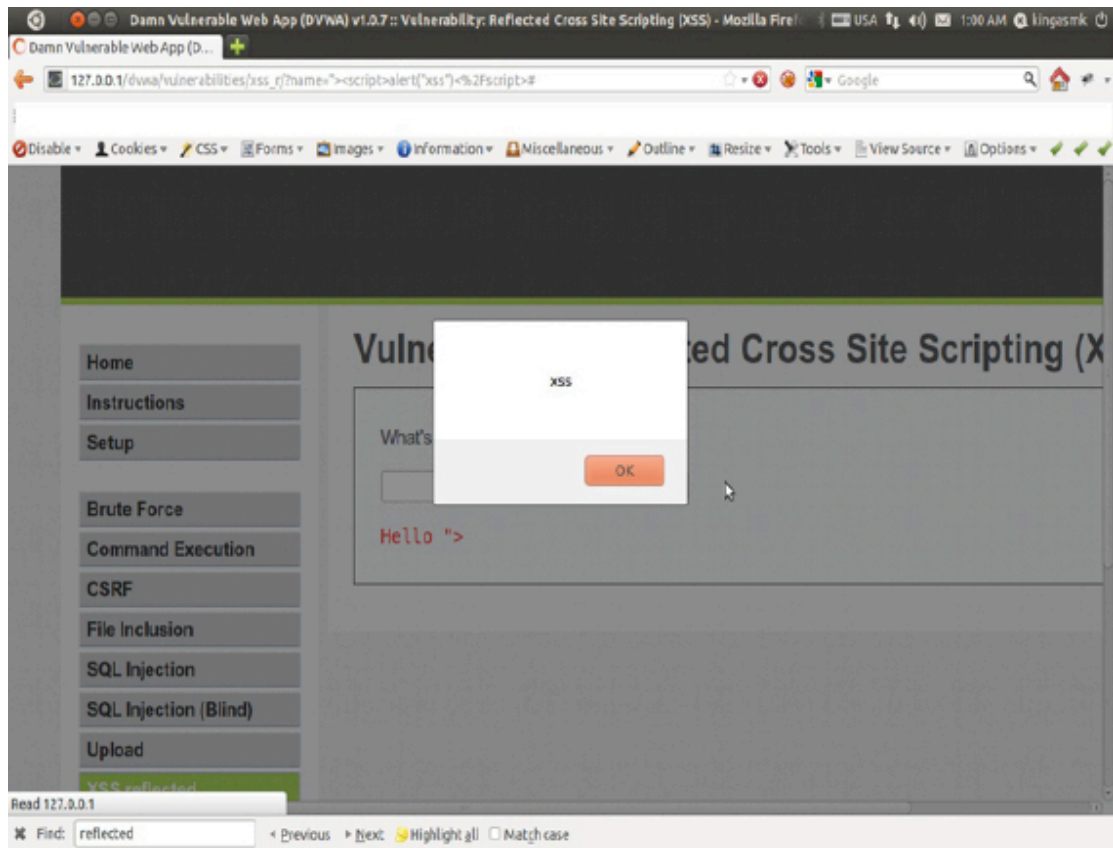


Figure 8 An XSS alert in Non Persistent Demo (reflected) XSS

2.2.2.1.3.4. DOM based XSS

DOM-based weaknesses exist within the component proceeding stratum that are operated through the client, usually in client-side JavaScript. This name allows to fix the model for demonstrating XML or HTML components that is entitled (DOM) JavaScript programs exploits this phase of a web page and occupies along with actively calculated information previously by performing upon the DOM. Basically this kind exists on the JavaScript code formative operation in the user side [42].

2.2.2.1.4. The advanced techniques

There are ignored techniques, which may be taken to guard against XSS manipulations; however, they are not applied accurately for instance;

Lot of sites might look weak but cannot manage the code that happens due to some sort of infiltrations techniques and those might be passed by and which shall be presented here.

METHOD 1: change <script> with a null string ""

This is a weak script, which is exposed to reflections of an XSS as shown in Figure 9. that has an infiltration:

```
1 <?php
2 if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name'] == ''){
3 $isempty = true;
4 } else {
5 echo '<pre>';
6 echo 'Hello ' . str_replace('<script>', '', $_GET['name']);
7 echo '</pre>';
8 }
9 ?>
```

Figure 9 PHP code reflects an XSS

In the prior script, the formative substitutes which is the string and is called "<script>" with Null string "" .

About popular tactics to pass by infiltration is that you need to substitute the string "<script>" with "<SCRIPT>" since the formative look for small letters is this"<script>", so to bypass it one needs altering out of code to <SCRIPT>...</SCRIPT>

METHOD 2: The filtration of magic quotes

This method, the formative operation technique which is termed magic quotes infiltration, by operating a function in PHP named ‘ ’ "add slashes ()" which unites slash before any other special characters. Therefore, our usual JavaScript code does not perform well. So various methods to avoid that filtrate are:

1. The most effortless way of bypassing is DON'T USE magic quotes, it is basic, for instance, notifying a characters and assigning it to a number.
2. This is the misleading part. This way it can operate a built-in function that inserts denary values into ASCII values. JavaScript inserts this to ASCII; this function is named as "String. From Char Code ()", and utilizing this with notifications this is going to show or message ‘XSS’, therefore this technique is quite handy to bypass magic quotes [43].

2.2.2.1.5. XSS Prevention

Building a website that is strong against the cross-site scripting contains app formatives, server executives and browser producers. Even though it is efficient at

diminishing the damage of such an attack, the recommended methods are not the entire answers. It is significant not to forget that web app security need to be a gradually improving journey. People who want to apply a secure web app alter their methods as the hackers alter theirs [44].

2.2.2.1.5.1.XSS Prevention Strategies

The basic strategy to avoid XSS is to never permit user entry to maintain originality. There are five approaches to message or filtrate user entry is to make sure that there are not eligible of forming the XSS exploit [45].

2.2.2.1.5.1.1.Make an Encode to HTML Entities in Non-HTML Output

As mentioned the one popular way to execute an cross site scripting attack includes injecting the HTML item functions and releases the attacking code. PHP's `htmlspecialchars()` function converts all of the chars with an HTML entity matching to these entities, therefore depicting them as not harmful. As shown in Figure 10, its associate `htmlspecialchars()` is more restrained, and should not be applied.

```
1  <?php
2  function safe( $value ) {
3      htmlentities( $value, ENT_QUOTES, 'utf-8' );
4      // other processing
5      return $value;
6  }
7  // retrieve $title and $message from user input
8  $title = $_POST['title'];
9  $message = $_POST['message'];
10 // and display them safely
11 print '<h1>' . safe( $title ) . '</h1>
12 <p>' . safe( $message ) . '</p>';
13 ?>
14
```

Figure 10 PHP code that Encode to HTML Entities in Non-HTML Output

This section is incredibly basic. When fetching the user entry, we forward it to the function `safe()`, that basically utilizes PHP's function `htmlspecialchars()` to each value that executes into it. That definitely saves the form of a HTML by being implanted

which saves JavaScript implanting too. After demonstration the resulting secure forms of an entry.

The function `htmlspecialchars()` also commands single and double quotation chars to entities, that makes secure management for each of the next probable form items:

```
<input type="text" name="my-val" value="<?safe($myval)>" />
```

```
<input type='text' name='your-val' value='<?safe($yourval)>' />
```

The second type entry –single quotation marks- is totally authorized markup, and if `$value` has a apostrophe or a singular quotation mark which is left behind in it after coding, the entry area may be divided, it will show a mark injected to the page. Hence, the `ENT_QUOTES` measurement need to be used with `htmlspecialchars()` on all occasions. There is a measurement which says `htmlspecialchars()` to transfigure the singular quotation mark to the entity (`'`); and the double quotation mark for the entity (`"`). As lots of browsers are going to depict this, some of the senior user will not, which is why `htmlspecialchars()` proposes a selection of quotation mark paraphrasing charts. This `ENT_QUOTES` setting is more traditional and more modifiable than `ENT_NOQUOTES` or `ENT_COMPAT` [46].

2.2.2.1.5.1.2. Make a sanitization for every user-submitted URIs

The permission for clients to particularize a UTI, (for example, the particularizing a personal image, or to build icon-based links as in a target index) it need to guarantee that it cannot operate URIs corrupting with JavaScript; descriptions. PHP's function `parse_url()` divides an URI in to a dependent range of parts. That makes it simple to control what the chart key items are. (Something allowed such as JavaScript. The `parse_url()` function also handy as it includes an inquiry key, which shows a supplement inquiry string (a `$_GET` inconstant or serials of them) if one occurs. Therefore, it gets simple not to permit inquiry strings on URIs's.

A hypothesis behind the advocating for a normal user would consider for the second time before affecting the link to an unsecured and unfamiliar site, particularly one with an unfavorable name. It might create a filtrate for the end user entered URI's like this, which may be also as shown in Figure 11, filter URI, PHP.


```

1 <?php
2 $trustedHosts = array(
3     'example.com',
4     'another.example.com'
5 );
6 $trustedHostsCount = count( $trustedHosts );
7 function safeURI( $value ) {
8     $uriParts = parse_url( $value );
9     for ( $i = 0; $i < $trustedHostsCount; $i++ ) {
10        if ( $uriParts['host'] === $trustedHosts[$i] ) {
11            return $value
12        } }
13        $value .= ' [' . $uriParts['host'] . ']';
14        return $value;
15    }
16    // retrieve $uri from user input
17    $uri = $_POST['uri'];
18    // and display it safely
19    echo safeURI( $uri );
20    ?>
21

```

Figure 11 PHP code Sanitize for Every User-submitted URIs

The script pieces are easy. It can build a lay out of the hosts, which are secured, and the function, which checks against the host portion of the user-approved URI (taken from the function `parse_url()`) to the elements within the secured array for the host. This revert the unaltered URI for showing if matched, whereas when added to the host part of the URI to the URI itself, that revert for showing if not a match. On that occasion, you have supplied the user with a chance to look at the original host that a link indicates, and so to conduct a logical determination if they need to click a link or not [47].

2.2.2.1.5.1.3. Use a proven XSS filter on HTML input

Some situations when the user input might neatly include an HTML markup, it also has to be specifically cautious with this input. In theory, it is not unimaginable to create a filtrate to deactivate user-approved HTML, but is not very easy to layer all of the probable situations. It needs to seek for an alternative in order to enable markup, which operates an incredibly restrained marks, and that, does not contain figures, JavaScript troubleshoot features or style features. Here, one might think one has wasted the advantages of HTML, which might be of superior value to permit just textual matter, from unsecured user. Even though having accomplished in forming a procedure that looks like working properly, one might think it is not trustworthy since it relies on a particular browser or even browser type.

In addition, the versatility required Internet procedures for promoting multisystem chars and variant coding may overcome the most skilled code infiltrating tactics,

since there are several paths to show any given char. Below, there are five severe alternatives on the exact risky one-line code, each coded looks like completely unique from the other (even if they are same), yet simply a standard browser or email client may depict them:

1. A Plaintext
2. The URL encoding
3. An HTML hexadecimal entities
4. An HTML decimal entities

As it may be probable to create a filtrate that is going to take every single alternative of the same thing, as a functional issue it is not probable to me to conduct dependent on one's lifetime. Filtrations that depend on fixed representations might be specifically difficult, since the number of several approaches that need to be controlled. We do not wish to be totally negativist. The usage of markup controlling library such as PHP's Tidy module is going to progress a great path in the direction of assuring that may seize and delete efforts at combining JavaScript, style features for other types of unwanted profit from user-approved HTML code which an app is going to need to demonstrate [48].

2.2.2.1.5.1.4. Make a private API strategy for Sensitive transactions

As shown in Figure 12 and Figure 13, preserving the end-users from unknowingly demanding a URI encoding which is going to result as an unwanted attempt to be shown in your app, it is suggested to form a private port for all-important activities, instead of the public site. After that, allow only demands to that port that are approved by \$_POST inconstant (so ruling out the probability of an attacker's operating \$_GET inconstant). Unite those limitations with some control of value for every type approved to the private port, like:

```
<?php
if ( $_SERVER['HTTP_REFERER']. != $_SERVER['HTTP_HOST']. ) {
    exit( 'That form doe not need to be used outside of its parent site.' ); }
?>
```

As it is correct it is not specifically hard to fraud a referrer, this is almost unfeasible to take a victim to act in the same manner, it is in fact what is expected (because of user's browser is approving the form). The research like the earlier one is going to protect bad attempts such as under mentioned attack, that must exist on a removed

site.

```
1 <?php
2 $trustedHosts = array(
3     'example.com',
4     'another.example.com'
5 );
6 $trustedHostsCount = count( $trustedHosts );
7 function safeURI( $value ) {
8     $uriParts = parse_url( $value );
9     for ( $i = 0; $i < $trustedHostsCount; $i++ ) {
10        if ( $uriParts['host'] === $trustedHosts[$i] ) {
11            return $value
12        } }
13        $value .= ' [' . $uriParts['host'] . ']';
14        return $value;
15    }
16    // retrieve $uri from user input
17    $uri = $_POST['uri'];
18    // and display it safely
19    echo safeURI( $uri );
20 ?>
21
```

Figure 12 PHP code use API strategy for Sensitive Transactions

It has a choice to not to permit any markup in textual matter output crosswise in the whole range of the private port. This might look extremely cautious if it uses some filtrations that are text arising from the system, but the earlier form is going to be lost in Tidy as in any other form. The ultimate method to inspect t-what users place to the system to get away all markups that are demonstrated. The under mentioned code portion demonstrates a little less security but probably more realistic system, where a function for a get away from all the outputs are raised by selection [49].

```
1 <?php
2 $trustedHosts = array(
3     'example.com',
4     'another.example.com'
5 );
6 $trustedHostsCount = count( $trustedHosts );
7 function safeURI( $value ) {
8     $uriParts = parse_url( $value );
9     for ( $i = 0; $i < $trustedHostsCount; $i++ ) {
10        if ( $uriParts['host'] === $trustedHosts[$i] ) {
11            return $value
12        } }
13        $value .= ' [' . $uriParts['host'] . ']';
14        return $value;
15    }
16    // retrieve $uri from user input
17    $uri = $_POST['uri'];
18    // and display it safely
19    echo safeURI( $uri );
20 ?>
21
```

Figure 13 PHP code use API strategy for Sensitive Transactions

A function `safe()` is attained via private port (on this occasion, `private.example.com`), gets away all the entry thus any markup included in it is going to be shown securely. This tactic enables you to understand what the users are really recording, and it may aid to find out brand new markup attacks because they appear in wilderness. If the function `safe()` is not raised through private port, this operates PHP function `strip_tags()` to strip out all yet a few innocent entities. Other function `strip_tags()` have a 1,024-char limitations, thus, when entering that you require strip to be more elongated than that, it is going to be divided into properly measured chunks and manage every single of them one by one, rebuilding everything when you are finished. When the single XHTML tags like `
` and `<hr />` stripped by identifying a former HTML substitutes (like `
`) or the current XHTML forms. In another way, the XHTML tag `` can be stripped solely by an HTML substitute ``. Furthermore, as the user makes comment on page quoted recommendations that iteration is significant for strip embedded marks.

2.2.2.1.5.1.5. Predict some actions that expect from users

There is probability to decide, depending on a new request, when to reduced the number of attempts that the user is going to get in afterwards. To illustrate, if a user wants a file-modified form they may anticipate his/her following attempt to be approving modification form proceeding or calling it off. It would not anticipate his/her to be keeping an independent activity roles from other portion of the site. An anticipation system may aid to see and protect XSS attacks that tip a user to keep doing the same unforeseeable activity. How would it appear? For every single requisition, pre-origination of all of the requisition URI's that it anticipates through the end user, also reserves them as hashes or strings in the period. Afterwards, on the following requisition, check against the new URI to the reserved array of anticipated URIs. If the match is not found, we are going to have to operate logic relying on the base and logic of the app as a whole to detect whether the requisition URI is not harmful or not [49].

2.2.2.2. SQL Injection

2.2.2.2.1. Introduction

An approach that is utilized in managing host-side codes, which directs SQL queries to an RDBMS might be conducted by managing requester-side data, consisting of altering SQL values and chains of SQL expressions that are directed to a network server implanted in HTTP requests. If ever the network server obtains a demand, it dispatches the data within it to a script, which utilizes that data to set up SQL queries. The purpose of the assaulter that utilizes SQL inserting manages with the SQL query utilized by the script, in this way, it would attain undesired solutions like alluring, injecting, managing or removing guarded series or tables within the database. Here before moving on, we must considered a discourse of the range of SQL inserting is requisite. Some authors have sorted the strike tactics of SQL inserting into direct and indirect strikes. By applying direct strikes, an assaulter attempts to takeover the RDBMS. The aim of similar strikes to this is to gaining more control over other server computers and compromises a system. Initially, assaulters are able to scrutinize open interface, which database hosts are listening to. In case of the detection of these interfaces, they proceed with delivering system orders via an ordering gadget, transmitting it with the RDBMS straightly. In other respects, indirect strikes are utilized thanks to web apps. It is correct, it is feasible for using the performing commands by implanting calls to reserved routines in active SQL and this might result in destroying results assuming them as accurate. Yet, the primary aim is to straightly striking the RDBMS in overall and its reserved data in detail. Direct strikes are not referred or conversed by most of the authors. This might arise from the given fact that they are not cognizant of these spoils or they do not regard them as exposing to the range of SQL inserting. Nevertheless, direct strikes are done by the RDBMS and which seek at the network base. Now that direct strikes are mentioned, 38 authors address to direct transmission with the RDBMS and not the strikes on the RDBMS itself. While utilizing such strikes, it appears to be true that the assaulters are able to capitalize on some aspects of SQL inserting. Yet, we study the concept of direct strikes, which are somehow being misguided because it does not connect web apps. Moreover, we believe that direct strikes links to network security instead of app security, from a security-wise approach. Spoils such as interfaces which permits assaulters to transfer with the RDBMS using inconsistent

standards from command gadgets might be obstructed by existing network security precautions along with the database security conformation, for example assuring the system executer account [50].

2.2.2.2.2.Fundamentals

As mentioned a web app might be displayed as including the next levels: Desktop level, transmit level, Access level, network level and app level. Computers with web browsers performing as clients are utilized for reaching a system at the desktop level. The transmit level corresponds the web, while Access level composes the beginning point into an organization's inner system base from the web. The network level composes of the organization's inner network base. The app level comprises of web and app servers, app logic and data memory. Unluckily, the only way to conserve SQL inserting strikes is that they need to be conserved within the app logic ingredients such as codes and programs in the app level. Regardless of the assets or efforts that have been spent in the other levels, if the app security has not been performed accurately in the app level, some weaknesses may occur inside of their web apps and SQL inserting assaulters may take advantage of it as a result of these weaknesses [29, 49]. We are not ignoring the fact that precautions such as codifying, firewalls, spotting an invasion and database security are necessary. They are particularly efficient while handling with the several kinds of strikes. Yet, considering SQL inserting, they have been found deficient and unsatisfactory, therefore we are not going to examine them. For instance, codifying just conserves preserved data or data while transmit in and among lesser levels. Operator input may be coded among the client-side and operator-side in the concept of web apps. Moreover, SQL queries might be codified whilst they transmit among ingredients like codes and programs on the operator-side. However, they need to be decoded initially, for operator-side to build SQL queries and for RDBMS to manage them. The data might be codified yet SQL queries might be exploited via SQL inserting. Mainly, the exterior part of the web servers is preserved through firewalls. But, from a security point of view, web apps promos client's valid channels via firewalls into an organization's systems. This is because, when users want services from servers on the web, the rudimentary transmission stands via HTTP, and web apps are not special case. HTTP is basically firewall-convenient, e.g. it barely has few protocols that most of the firewalls permit through. That arises from the fact that HTTP

demands are seen lucid, because traffic amongst users and servers it need to be permitted for the new apps to be useful. SQL inserting exploits from this feature by implanting strikes in HTTP requests. These strikes are executed at the back of firewalls via the app level. Thus, SQL inserting does not need expertise gadgets or vital experience or know-how. As long as the assaulter possess primary information of HTTP, associational databases and SQL, a web browser is adequate to run SQL inserting strikes against web apps. Even though the RDBMS is protected by appropriate conformation, the database might still be weak for SQL inserting strikes. SQL queries are managed as long as they are legitimate and well structured and when the clients possess the desired priority [51].

2.2.2.2.3. Strike process

We have discovered that assaulters pursue a standard including of a serials of steps as a whole. Our gathered standard demonstrates all of the stages analyzed corresponding with all the possible strike tactics. Assaulters can relate tactics in the strike to meet their target, yet the period is run for each strike tactic in a repetitive approach. Some steps might be disregarded, based on the striking method. Defining the target: It does not matter if it is definite or uncertain, assaulters possess one or more targets in order to make SQL inserting strikes. An accurate illustration can be that an assaulter wants to attain the web app to reach the data about a organization's clients. This is a strike to the security service privacy, picking the method: In some instances, the assaulter just looks for reaching to the web app, thus attempts to bypass confirmation. In some other circumstances, bypassing confirmation is only one single pace before one can attempt to attain the targets. Therefore, there are various tactics to be chosen. Analyzing preconditions: To decide whether the targets are attainable or not, the assaulter automatically controls which preconditions are supported. Preconditions can be significant circumstances for a given strike tactic, or make the strike smoother to carry. Assessing weaknesses: The assaulter starts assessing for the weaknesses to take advantage, for example, trying out with entry confirmation by penetrating solitary quotes, listing priorities or appraising the information which is reverted. Selecting means: The assaulter picks out his means for the strike, based on strengthened preconditions and detected weaknesses. Planning the query: The query, which is planned by the assaulter, needs to pursue the exact shape of an SQL query anticipated by the RDBMS. Otherwise, typing errors

are made and shown in error messages. One sample of typing errors links to citation marks, for instance, if SQL inserting is feasible without missing them. Another sample is, if parentheses are utilized in the root query. Based on the target, other typing errors, which interest information recall of database structure, might have to be over-reserved like table names, column names, list of numerical and data kinds [51].

2.2.2.2.4. Terminology

This part presents our categorization of the area of SQL inserting with regards to common criteria. Studying SQL inserting from a common angle leads to review the issue regardless of factors such as RDBMS or app logic ingredients selected. Hence, we select desperate SQL inserting issues into wide range of groups of interest, to top that point, every group demonstrates some standpoint of the area and include series of associated items to think about. The rank of the items in every single group is inconsistent and we do not try to count them [49].

2.2.2.2.5. Security services

The targets of SQL inserting strikes might be reflected from the point of compounding security services discussed previously. We regard the security services below to keep resource security regarding SQL inserting. We are acquired the autonomy of changing their explanations to a minor extent:

1. Access control: It concludes assuring that clients can only reach and exploit data according to their priorities.
2. Accessibility: The services proposed by a web app needs to be ready for users when they demand them.
3. Legitimacy: Assuring that clients who log into a web app proof of who they claim to be.
4. Confidentiality: Assuring that information is and will be maintain as a secret. This security service can be separated into privacy and secrecy.
 - 4.1. Privacy: Personal information that regards workers and clients should be kept a secret.
 - 4.2. Secrecy: Sentimental Office/business-wise data needs to stay private.
5. Means: Assaulters may utilize various means to operate a strike such as given below,

1. Web page construct exploits: An assaulter can operate forms to penetrate parts of SQL claims like SQL operative words, suppress chars or data to keep actual app user-side codes or programs.
2. URL header manipulation: Likewise to m1, some sides of SQL queries might be penetrated into a page's URL, directing manipulated discussions to the user-side.
3. XSS scripting: By observing the source code of web pages and analyzing existing user-side codes, an assaulter may compose a set up code and utilize it to forward information to the actual server-side codes and programs rather than using the initial code [50]..
4. Error message inference: By analyzing the error messages that are executed by RDBMS or server-side codes, an assaulter can recover information of the database layout: table and column entitles, the number of columns and the type of the column data. Moreover, error messages may include information of SQL query typing and how the codes are built.

2.2.2.2.6. Strike methods

Various ways can be counted in terms of making SQL strikes, and the chosen ways based on what the assaulter is going to achieve, for example, which security services to threaten, and what weaknesses the web apps consist of. These ways comprise some dangers; they might be classified into two categories:

1. Data controlling: By performing data manipulation, an assaulter may bypass validation as much as recovery alter, frame or erase data within a database.
2. Order invocation: It is a tactic which assigns an assaulter to manage SQL particular system orders via the RDBMS and can permit the assaulter to gain control over other server computers within the network. This tactic also manipulates stable SQL from active SQL, for example tampering with and executing a present SQL query and inserting a fresh claim, which addresses a reserved standard. The assaulter can duplicate and email database tables to an unfamiliar account in case he of calling system standards, which happen with the RDBMS. Another path is to attempt to call reserved standards that have been made-to-order by system enhancers, database managers or app programmers for the web app. Here category a1 may be slightly split up into vivid paths as given below:

- 1.1. Bypass validation: An assaulter can utilize this path to make it appear fake to be a genuine user.

1.2. Data retrieval: On the purpose of reaching access to data to use their privileges more, assaulters might attempt to exploit to manage SELECT claims. This could be yielded through e.g. exploiting the WHERE clause. One more example is that, by performing UNION, bringing on rows from greater tables to be reverted than being determined in the original query.

1.3. Exploits of information: Assaulters may attempt to exploit or manage UPDATE claims on the purpose of changing information according to their priorities.

1.4. Formation of information: Assaulters may attempt to exploit or manage INSERT claims on the purpose of changing information according to their priorities.

1.5. Erasing information: Assaulters may attempt to exploit or manage ERASE or DECLINE claims on the purpose of changing information above their priorities. A small amount of authors consider the paths a1.3 via a1.5 and those are not mentioned in the list above, as few provide valid illustration. Instead of these, paths are contained in author's documentation indirectly. We believe that one cause for this can be their compositions and documents, which are not attempted to completely comprise all of the accessible paths. In spite of that, as stated in section 1.8, our test system has verified those paths a1.3 via a1.5 are legitimate SQL inserting strike methods [50].

2.2.2.2.7. Pre-conditions

We have discovered that distinctive SQL inserting methods require distinctive pre-conditions to be held. These pre-conditions are linked to both query managing features and other features that RDBMS strengthen as much as the features of coding languages that are performed for applying scripts and programs. The essential point in here is, which RDBMS and programming languages propose are not features. Instead, the inquire regards which features are backed up by the RDBMS and programming languages are selected in a stated web app. These pre-conditions are not significant to make SQL inserting strikes. Preferably, they need to be displayed as ingredients, which make strikes simpler to get done. For instance, pre-condition p4 can authorize an assaulter to add an INSERT query right after a planned SELECT query. Yet, the assaulter might also attempt to figure out an area in a structure where an INSERT query is already awaiting. Indications that are done in this part refer to which authors that specifically discusses them as pre-conditions.

1. Sub selections: They are multi SELECT which claim utilized untidily.

A Highest-rank SELECT claim is utilizing other lesser-level claims to recover values to be performed in a WHERE clause.

2. JOIN clause: It can be performed in case of multiple SELECT queries united within the identical query.

3. UNION clause: It can be performed in case of multiple SELECT queries united within the identical query.

4. Multiple claims: It makes reference to the aptitudes to permit managing of multiple SQL claims, where every single claim is divided into a level limiter such as semicolon.

5. End of row comments: The capability to comment out the portions of a SQL claim, so that the SQL typing does not recognize the RDBMS followed by a comment sign.

6. Exclusive (privileged) accounts: Database links use accounts that are specified in the database to attain the database. An assaulter can only make strike methods that manage SQL claims connected with specified privileges in the account used by the web app. For illustration, when the account does not define DELETE as a privilege, the assaulter is not able to use m4 as a strike method.

7. Error messages: Errors that exists in the RDBMS or in any operator-side code or program may yield an error message, which may be directed to the user and published in the web browser

8. Vulnerable data types: Various code and programming languages are utilized in web app development promote factors of vulnerable type 9, e.g., factors that may reserve data of uncertain type.

9. Data type transition: Various RDBMS's promote several type transition, for example, permitting numerical values to be transformed straightly into a thread type.

10. Preserved standards: Standards like that, promoted through some RDBMS's, permit managing of system or database orders and SQL sub-program in the RDBMS.

11. Active SQL: On the purpose of implanting operator input into SQL queries, server-side codes and programs which may utilize actively form SQL queries at the point that SQL claims are united with user inputs and after that assigned into the RDBMS for managing.

12. INTO OUTFILE promotions: If INTO OUTFILE is promoted by the RDBMS, users can publish query concluding into a document on the host computer [43].

2.2.2.2.8. SQL inserting security model

One target is stated in section 1.5.1 on page 8 was to form a common security model for SQL inserting. Here it is gathered how divergent ingredients, reflected as common criteria and specified in earlier sections and link to each other into a model like that. In this case, we begin by picking a strike method and describe it in which security services it belongs. After that, we rank the pre-conditions that are required to delegate strikes using this method, proceeding with demonstrating the weaknesses needed. Ultimately, precautions that are used for preserving web apps versus the chosen model are shown. Most of the rankings are presented under services; pre-conditions, weaknesses and precautions which include various items. It does not mean, that all of the items consisting of such rankings are unexceptionally of vital conditions. Our model basically shows an assaulter device box when preparing and dedicating SQL inserting strikes. We are going to perform this model over the course of assessing existing restraint tactics [49].

2.2.2.2.9. SQL Inserting strike samples

We have considered several strike samples as well as examined them in the system. Our attempt is not to define every single sort of strike tactic and their modification/variations, because they might be discussed in our referenced research paper. Yet, in order to be simplified, we demonstrate samples that we both present data on how SQL inserting strikes might be applied and how we precede strike samples throughout the formation of our model. Strikes on the database might be done to attain web app, hence, achieving in intimidating the app's confirmation security service. Assume that an assaulter requires to attempt and log into a system, which uses a web interface, which has not been guarded from SQL inserting strikes. This can be in the very initial stage an assaulter gets to be convenient to dedicate further strikes on an app. To do that, an assaulter can attempt to evaluate how a login page, like the figure 10 below might be manipulated. Here on the purpose of making it fully utilizable, some pre-conditions are required, and some linked weaknesses need to be remained undone. Firstly, the app should support active SQL queries and the app needs not to be confirming the user's input or check the input's kind and size. Moreover, if it just shows operator-side-generated error messages, it would be practical to the assaulter if end-of-line comments were promoted. The assaulter might initiate the strike by going into some selected signs and SQL keywords in the

login page areas and study the error messages. The query is forwarded to a SQL server database that reverts the rows that corresponds the query. The script controls if any rows are reverted. If it is correct to revert, pointing that the user is classified. If not, the script reverts fake, so that users cannot enter in the database. The system regards the user verified or not, depending on the reversion value taken from the script or program that needs the generating of the query, and enables the user to utilize it if verified. The query is then going to revert a conclusion including a users in database since the conditions in the WHERE clause is going to be assessed correct. Hereby, the script is going to be also reverting correct, and the query generates a legitimate user whose identity the assaulter would pretend when utilizing the system. The assaulter is going to possess the priorities detailed in the account that is done for reaching the RDBMS. Diverse accounts might be specified in the RDBMS and app logic might select distinct accounts for several other users. Suppose all rows are reverted, they may have been resolved by the RDBMS depending on ID, name or other measures. The app logic can conclude which user logs in because of the outcome that is set by regarding at the initial row. The system manager can be the number one user specified in the table of users thus possessing the lowest ID. So, if the app logic selects the initial row, it is probable that the assaulter is going to log in as the system manager, so that all the priorities will be delegated to that account, automatically. Assume that the assaulter would request to precede the strike, but he would try to inject a fake user ID into the database table that he could utilize for expecting strikes, rather than bypassing confirmation. Hence, the strikes could try to fake the data and could threat the app's access control security service as much as its unity. The assaulter now may require a path to exploit the app into going into the user ID into users table within the database. That can be conducted in a few days: the assaulter may attempt to entitle a preserved standard (if such happens in the RDBMS utilized by the app) that would enter a row into the users table. Another method could be to attempt and discover an area in the app, which is performed for entering data into the database, and to deal with it. One other method might be to exploit the above discussed login area query and add to it an INSERT claim. That would be applicable in case of promoting multiple claims by RDBMS. Whatever approach maybe, the account selected by the web app when the assaulter-surrounded confirmation would require having the matching priorities. To illustrate, either one of the second two are selected from any user in the world need the priority INSERT on

the table users. That could be reflected by privileged accounts p6 and v5. Assaulter only has to figure out what areas the users table includes and what types each column needs, and then it will go into the exploiting information into the login area [48].

2.2.2.2.10. Preventing SQL injection

As we have examined the SQL injection, the ways to run it and in which stage one are weak to it, it is time to analyzing methods to prevent it. Luckily, PHP possess big sources to present and it assures to anticipate that a cautious and complete app of the methods that are suggested are going to fundamentally erase any of the possible SQL injection in these codes, by clearing out the users' information long before the disfigure of anything [36].

2.2.2.2.10.1. Demarcate each value in the queries

There are suggesting to ensure about delimiting each one of the values in the inquiries. String values should be delimited obviously, and for these it needs to usually anticipate operating single (instead of double) quotation signs. First of all, making this might take writing the inquiry more simple, if there are operating double quotation marks to allow PHP's inconstant substitute in the string. Secondly, it avowedly reduces a parsing activity that PHP needs to precede. Demonstrate this with the untouched, non-injected inquiry:

```
SELECT * FROM wines WHERE variety = 'lagrein'
```

Or in PHP:

```
$query = "SELECT * FROM wines WHERE variety = '$variety'";
```

Quotation marks are practically not required for numerical measures. Yet if one determines not to attempt to place quotation marks next to a value for an area for example, the term 'old' and if the user typed a null value to the form, one will see an inquiry like that:

```
SELECT * FROM wines WHERE vintage =
```

This query is, of course, syntactically invalid, in a way that, this one is not:

```
SELECT * FROM wines WHERE vintage = "
```

The secondary inquiry is going to (assuming) revert any outcome, yet it is going to revert an error message leastwise, as an incited null value is going to (even if one have switched off all error signaling to users) [50].

2.2.2.2.10.2. Checking the values types of users' submitted

As mentioned previously that a first SQL injection, which tries source, is an unforeseeable method input by so far. When presenting a user the opportunity to approve some kind of value through a form, ones still possess the substantial specialists of recognizing earlier that what sort of entry needs to be attained. This needs to be done comparatively simply to run a basic controlling according to the availableness of the user's input. If anticipating a numeric (in order to maintaining our former sample, the year of a old wine for example), then it may perform one of those methods to assure what it takes is definitely a number of applying functions:

1. `is_int()` function.
2. `gettype()` function.
3. `intval()` function.
4. `settype()` function.

When controlling the scope of users entry, it might apply the function `strlen()`. A control if an estimated time/ date is authorized, it may apply the function `strlen()`. It is going to be nearly handy to assure that the user's input does not include a semicolon char. It may conduct it simply with the `strpos()` function, such as:
`if (strpos($variety, ';')) exit ("$variety is an invalid value for variety!") .`

2.2.2.2.10.3. Abstract to improve security

Here we are going to discuss the main abstraction, which would integrate the approval results into such function, and this function for every element of the user entry. Much more compound one might retreat even more, and materialize the total procedure of building trusted inquiry in a stage. Such abstraction has three advantages leastwise, and each of them adds a value to a developed stage of security:

1. It sets in place the script, which reduces the probability of losing procedures in that situations (a new source or stage happen to be feasible, or it can be proceed to new database with diverse composition and syntax).
2. It can makes formation inquiries for each credible and quicker, this by proceeding portion of task to the abstracted script.
3. When formed with security ad performed accurately, it is going to forbid the types of injection that we have been analyzing [47].

2.3.Smartphone application

2.3. 1. Introduction

IOS (formerly iPhone OS) is a mobile operating system that is revealed by Apple Inc. and given solely for Apple hardware. Most of the organization's IDevices are empowered by this OS. It is initially revealed in 2007, the iPhone, and expanded it to promote other Apple tools such as the iPod Touch (September 2007), iPad (January 2010), iPad Mini (November 2012) and second-period Apple TV (September 2010). Beginning from June 2014, Apple's App Store included over 1.2 million IOS apps, 500,000 of that were optimized for iPad. These apps have jointly installed over 60 billion times. It possessed a 21% share of the Smartphone mobile operating system segments transported within the last quarter of 2012, at the back of Android from Google. Until half of the 2012, 410 tools were operated. 400 million tools had been sold till June 2012 the exclusive media event that was organized by Apple on September 12, 2012. The user interface of IOS relies on the context of direct manipulation, by utilizing multiple-touch gestures. Sliders, switches and buttons cover the interface control components. Some apps operate internal accelerometers in order to reply to quiver the tool (undo command is the one mutual consequence) or revolving it in three size (re-routing from portrait to landscape style is the mutual consequence). IOS shares out with OS X some concepts like Core Foundation and Foundation; but its UI device kit is Cocoa Touch instead of OS X'S Cocoa, therefore it supplies the UIKit concept instead the AppKit concept. It is because it is not coherent with OS X aps. In addition to this, when IOS also shares out the Darwin foundation with OS X, Unix-like shell entree is not available for users and forbidden for applications, conducting IOS also not completely Unix-compatible. Vital forms of IOS are launched every single year. The present launch, iOS8, was launched on September 17, 2014. In IOS, four abstraction level exist: the Core OS level, the Core Services level, the Media level, and the Cocoa Touch level. The present form of the OS (IOS 8.0), devotes 1.3 – 1.5GB of the tool's flash memory for the system separation, operating hardly 800MB of that separation (depending on the model) for IOS itself. It operates on the iPhone 4S and after, iPad 2 and after, every models of the iPad Mini, and the 5th-generation iPod Touch [52].

2.3.2. History

When Steve Jobs start to outline the iPhone, he had two options, “shrink the Mac which could be a significant accomplishment of engineering, or expand the iPod”. Jobs took on the prior method yet pitted the Macintosh and iPod groups, under the leadership of Scott Forestell and Tony Fadell, conflicting one another into inter corporate rivalry. Forestell overcame the rivalry to form the iPhone OS. The result let this accomplishment of the iPhone as a stand for third-party formatives: operating a mainstream desktop OS as its foundation permitted a lots of third-party Mac formatives to compose software for the iPhone with slightest retrain ability. Forestell was as well as in charge of forming a software formative’s kit for programmers to construct iPhone apps, also of an App store in iTunes. The OS was launched along with the iPhone and Macworld Conference & Expo on January9, 2007, and brought out in June of the same year. Initially, Apple marketing publicity could not indicate a new name for OS, expressing basically the exact thing that Steve Jobs said: “iPhone operates OS X” and operates “desktop apps” when actually it operates an alternative of Mac OS X, which does not operate OS X software if it has not been interfaced to the inconsistent OS. Primarily, third-party apps were not verified. Steve Job’s thinking was developers might establish web apps that “might act as inherent apps on the iPhone”: On October 17, 2007, Apple broadcasted an inherent Software Development Kit (SDK) was in progress and that was designed to set it “in developers’ hands in February”. On March 6, 2008, Apple launched initial beta, along with a brand new name for the OS: “iPhone OS”.

Apple had launched the Pad Touch, which possessed most of the features of iPhone that are non-phone. Apple also sold out over one million iPhones throughout 2007 holiday term season. Apple publicized the iPad on January 27, 2010. iPad offered a bigger screen than the iPhone and iPod Touch, and planned for web browsing, media usage and reading IBooks. Apple re-tagged iPhone OS as ‘IOS in June 2010. The brand ‘IOS had been operated by Cisco for more than ten years as its OS, IOS operated on its routers. Apple got ‘IOS license from Cisco in order to prevent any possible case. IOS possessed 60% of the market share for tablet computers and smartphones towards the end of 2011. IOS possessed for 21% of the smartphone OS market and 43.6% of the tablet OS market towards the end of 2012 [53].

2.3.3. Home screen

The home screen (depicted by and also recognized as “Springboard”) demonstrates app icons and a dock at the lower part of the screen where users might attach their most often entered apps. The home screen materializes every time the user unlocks the tool or pushes the “Home” button (a material button over the tool) during in other app. The screen’s background might be concentrated with another concentrations that are accessible during jail breaking. The screen holds a status bar crosswise the top to show data, for example time, battery level and signal power. The remainder of the screen is dedicated to the ongoing app. When a password is fixed and a user opens the tool the password need to be entered at the Lock Screen before attainment to the Home Screen is given. Since IOS model 3.0, a Spotlight Search function has been accessible on the leftmost page of the home screen page permitting users to look for via media (music, videos, podcasts, etc.), apps, e-mails, contacts, messages, reminders, events (in calendar), and alike documents. Third-party application documents were not, and they are still not, ascertainable operating the Spotlight mark. In IOS 7, this mark might be reached by pulling down every place in the home screen (excluding the top and bottom borders which start Notification Center and Control Center). Research workers figured out that users plan images on home screens depending on consumption-frequency and connections of the apps, also for causes of the serviceability and artistic [54].

2.3.4. Folders

With IOS 4 appeared the entrance of a basic document format. Apps when are in “jingle mode”, any of two (excluding Newsstand in IOS 5 and IOS 6, which performs as a document) might be pulled one thing on top of another to form a document, as well as afterwards, more of apps might be attached to the document operating the exact same standard, till 12 on iPhone 4S and prior and Ipad Touch, 16 on iPhone 5, as well as 20 on Ipad. A headline for the document is directly picked by the division of apps inside yet the name might also be modified through the user. When applications inside documents get tags, the numeric demonstrated by the tags is attached and demonstrated on the document. Although an informal workaround places which permits documents to be nested in documents, they cannot be placed in other documents. IOS 7 updated documents with pages such as on Springboard.

Every single page can keep nine applications, and the Newsstand app can be put into a document now [55].

2.3.5. Multitasking

IOS multitasking was initially launched in June 2010 with the launching of IOS 4.0 Just iPhone 4, iPhone 3GS, and iPod Touch 3rd generation. The iPad could not possess multitasking till its launching of IOS 4.2.1 in November 2010. In present, multitasking is promoted in iPhone 3GS or in upgraded ones, and all of the iPad models. Operation of multitasking in IOS has been talked and pondered for its method, which restricts the job that apps in the background thus might act to a narrower function, as well as for demanding app formatives to attach explicit back up for it. Multitasking was reduced to a option of the apps Apple contained on the tool velour IOS 4. To multitask as unauthorized, users might ‘jailbreak’ their own tools. Beginning with IOS 4, on its third-generation and more recent IOS tools multitasking is provided by seven background APIs:

1. Background audio – app proceeds to operate in the background as much as possibly it can in playing audio or video content.
2. Voice over IP – app is excluded when an ongoing phone call is not in process.
3. Background location – app is informed of locale alterations.
4. Push notifications
5. Local notifications – app agendas local notifications to be carried at a preset time.
6. Task completion – app questions the system for additional time to finish a delivered duty.
7. Fast app switching – app does not perform any script and can be deleted from memory all along.

In IOS 5, three fresh background APIs were presented:

1. Newsstand – application might install meaning in the background to be prepared for the user.
2. External Accessory – app transmits with an outside component and allocates data at fixed intervals.

3. Bluetooth Accessory – app transmits with a Bluetooth accessory and allocates data at fixed intervals [54].

In IOS 7, Apple presented a fresh multitasking items, offering all applications with the capacity to operate background updates. That item advances to update the user's most often-utilized applications and favors to operate WiFi networks above a raster network, without significantly decreasing the tool's battery survival. In IOS 4.0 to IOS 6.x, double-clicking the home button sets in motion the app switcher. A transfuse dock-like port performs from the bottom, forwarding the meanings of the screen up. Picking an image switches to an app. To the left are images that function as music command, a rotation lock, and on IOS 4.2 and up, a volume commander. With the launching of IOS 7, double clicking the home button initiates the app switcher. Yet, in contrast to prior forms it demonstrates screenshots of open apps on top of the image and flat rolling enables for browsing via former applications, and it is feasible to shut apps by pulling them up, alike with how Web OS are carried on variant cards. Shortly keeping the images in the app switcher makes them "jiggle" (alike to the home screen) and enables the user to empower stop to the apps by patting the red minus round, which shows at the border of the app's image. Uncluttered apps from multitasking remain identical from IOS 4.0 via 6.1.6, to the recent version of ios6. For IOS 7, the progress has become quicker and simpler. It is not vital to keep the images shut anymore; they can basically be tapped off the screen in IOS 7. Up to three apps might be uncluttered at a time associated to one in versions up to IOS 6.1.6. Task completion permits apps to precede a determined task right after the application has been interrupted. For IOS 4.0, application might ask up to ten minutes to finish off a task in the background [55].

2.3.6. Development

The apps need to be inscribed and gathered particularly for IOS and the 64-bit arm architecture or prior 32-bit one. The Safari, web browser corroborates web apps along with the other web browsers. Licensed/authorized third-party inherent apps are feasible for tools that are operating IOS 2.0 and after via Apple's App Store [55].

2.3.7. App store (IOS)

The App Store is a digital allocation stand for mobile applications on IOS, formatted and preserved by Apple Inc. The service permits users to scan and install apps, which

are formatted with Apples IOS SDK. The applications might be installed automatically to an IOS tool, or to a customized computer through iTunes. (it is also formatted and preserved by Apple Inc.). There are other mainstream stores for Android, Windows Phone, and BlackBerry 10, so that the "App Store" is not the only store accessible. Apps within "the App Store" are objected at IOS tools, consisting iPhones and Ipads, and might get usage of particular features of these tools, e.g., motion sensors for game commands and cameras for online video calling. Applications can be installed costless or for a predetermined price, and they might consist in-app purchases (costs that are imposed via purchasable features and/or ads). Apple derives 30% of all receipts, which are taken through the applications, and 70% directly forwards to the app's publisher [56].

2.3.8 IOS SDK

The Software Development Kit for iPhone OS was disclosed at the iPhone Software Roadmap meeting on March 6, 2008. The SDK permits formatives operating Mac OS X 10.5.4 or greater on an Intel Mac to build apps utilizing Xcode which is going to inherently operate on the iPhone, iPod Touch and iPad. A beta version was unveiled right after the meeting and an ultimate version was unveiled in July 2008 with the iPhone 3G. Until March 2014, the most recent IOS SDK is for IOS 7.0 This vital Roadmap meeting (matched with a bigger allocation schedule for third party formatives), after enhanced to a iPhone Developer Program, that presently proposes two allocation paths for third-party developers: Standard, and Enterprise. Apps allocated via specific program might be merchandised privately by the iTunes Store on Mac and Windows, or on the App Store on the iPhone, iPod Touch, and iPad. Formatives who place their apps on the App Store are going to obtain 70% of sale proceeds, and that is not going to require payment of allocation costs for the app. Yet, a yearly charge is obligatory to utilize the iPhone SDK and installing of apps to the store. Apps formatted via the enterprise/endeavor schedule, formally the "IOS Developer Enterprise Program" (iDEP)", are private for premises usage and do not take place in App Store. This permits companies, NPO's and government offices to build patented "in-house" apps not for open release. The enterprise program was updated in September 14,2010, to permit any group with a DUNS associate to link. Before that date, only groups with 500 or more workers were able to unite the

enterprise program. The app needs to be contracted in order to utilize an app on the iPhone.

CHAPTER THREE

DESIGN AND IMPLEMENTATION FOR WEB APPLICATION

3.1.Introduction

As mentioned in chapter one, the aim of this project is to develop an online registration system for university students. This system is used for providing an online application form for students to fill it and insert their personal information into the database. The proposed system consists basically of two Parts: full register system and database. The register system consists of a designed Web app and program system by an HTML with PHP server-side-scripting languages. HTML provides a form, which prompts a system Interface for users, while PHP script carries out all system functions (i.e. take the data from the users and save/retrieve it to the database) by connecting to the MySQL database server. The database contains all information about student and its users that uses the system (Manager, Employees, Students). The systems have a two security check levels: security from Injection attacks (SQL attack and XSS attack) and data entry security. Also the system can be checked for the student information to see if it is completely filled or not. Many web application's security problems result from generic input validation problems. Flexibility of SQL makes it a powerful language. It allows the user to ask what information he/she wants without having any knowledge about how the information will be fetch. However vast use of SQL based databases make it the center of attention of hackers. Databases are the main storage of potential confidential information online and this is a good motivation for attackers to target them, also because of huge number of students can enter to the registration system and this make a threat for the data breach we design a secure registration system for students. The system was implemented using the facilities of Apache server, PHP & HTML languages with MySQL database.

3.2. System Features

Çankaya University has a good website that offers many services to visitors (students and employees), the E-library is one example for it, choosing courses through the Internet, as well as e-mail. These services are only best for the students and staff inside the university, "What about the new students who wants to register in Cankaya University?", this may be for international students (from Foreign countries). Students information archiving, which take a lot of time and effort because of the traditional system used in the archive. Now using this system we can:

- A. Provide the ability of registration process online.
- B. Track flow student's activities.
- C. Archiving student's information.
- D. The provision of facilities for the staff of the Institute.
- E. C. Secure registration system.
- F. Use the mobile to register and track flow the situation.

To provide a technical solution that depends on information technology in solving this problem and this fills gap in the Website of Cankaya University. This application helps to analyze the student information (Number of students, Student gender, and the student situation).

3.3. System Function

Function steps of student registration system can be distributed into these roles: administrator, the student, the manager and the employee. The administrator has the responsibility to maintain the entire data of the system, the student makes use of this system to get registered and follow-up his status. Manager is responsible for making decisions about student status, and the employee is responsible about the information of the students and the statistical as well as the analysis of data. The main system function used in this case is shown in Figure 14.

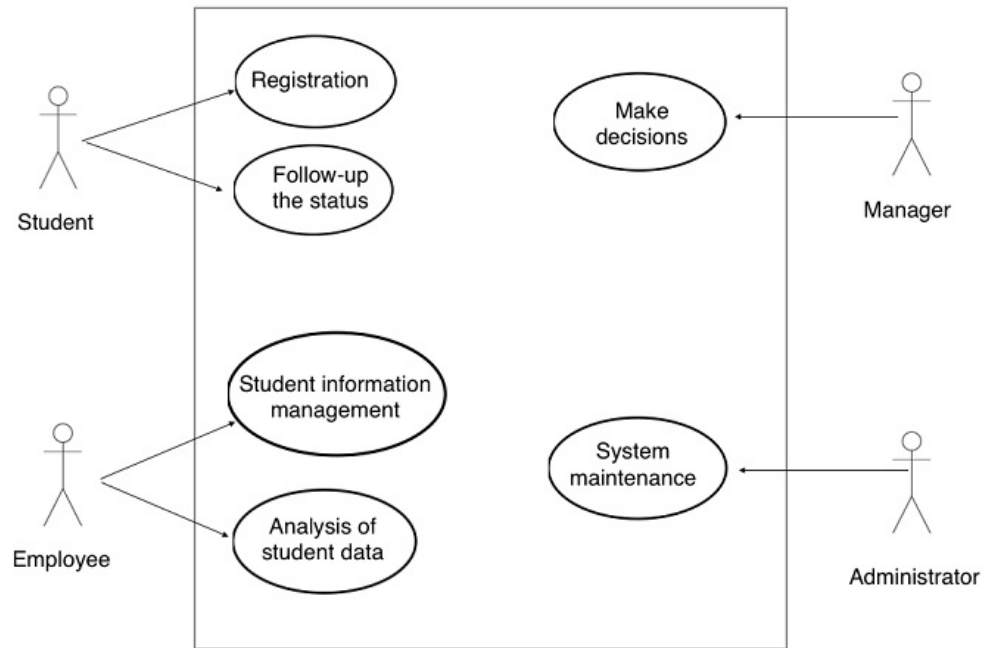


Figure 14 Main system Function (Use Case) Diagram

3.4. System Implementation

3.4.1. An Environment of system development

Web Server: XAMPP server supports MySQL Server, Apache Server that supports PHP and it can be used for both management system and as testing platform.

Database Server: MySQL database server that embedded in XAMPP web server.

The Development Tools: PHP Editor with HTML, which can be created as a user-friendly interface. PHP can interact with MySQL database data and can be interpreted by the web server as shown in figure 15.

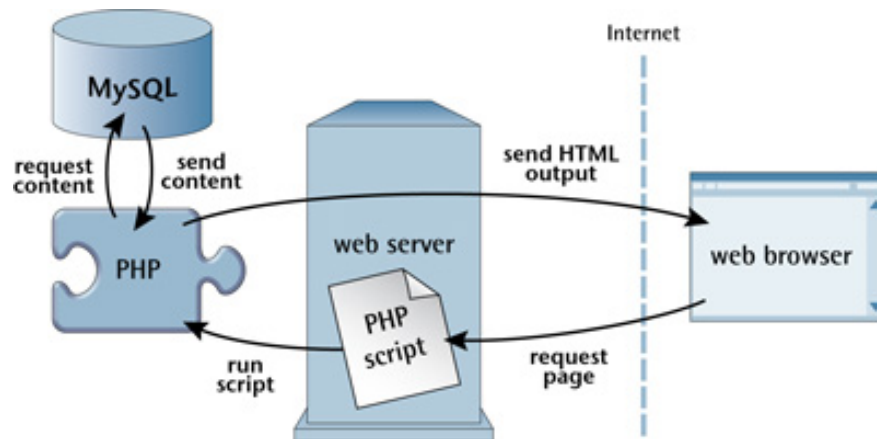


Figure 15 PHP retrieves the data from MySQL database

In figure 15 it shows when someone enters this page on the database-driven web-base system:

1. The end user requests the webpage by using a URL.
2. The Server can be recognized by the user requested file is PHP script, the web server interprets this file by using PHP plug-in and after that responding the requested page.
3. The certain commands of PHP that connected to MySQL database and retrieve the data that belongs in the web-based page.
4. MySQL database responds through sending the requested data to the PHP script.
5. PHP script can be store the database data into the PHP variables.
6. The PHP scripts can print the output data as part of Web Page.
7. The PHP plug-in stop when a copy of HTML it can be created to the server.
8. The Web server will send the output HTML to the Web Browser or end user as a plain HTML file.

3.4.2. System development languages

PHP language is an open-source; it is a server-side scripting language. Its main reason of using PHP language is to allow the Web developers to design quick and easy dynamic web pages. Also it is common for creating database-driven Web sites. PHP is used for connecting to a database; to recover, add and update information. That makes PHP a perfect language for developing large-scale websites. Single PHP template is written to recover and show all database records. HTML is short for

Hyper Text Markup Language and is a language used for creating electronic documents, particularly pages on the World Wide Web that contain connections called hyperlinks to some other pages. Every web page seen on the Internet contains HTML code, which helps in formatting and showing text and images in a simple readable format. Without HTML a internet browser cannot know possibly to format a page and can only display plain text with zero formatting that has no links. Web pages can be accessed by Web address, which is recognized as a Uniform Resource Locator (URL). A Web site's home page is an HTML document that has links to other HTML documents that can be possibly stored on the same server or on a Web server everywhere in the world. MySQL, like majority of other transactional relational databases, which are structured collection of data. To add, find, and process data stored in a computer database, you require a database management system like MySQL Server. Since computers are very good in managing large amounts of data, database management systems has a central role in computing, as standalone utilities, or as parts of other applications. Most of MySQL's appeal is because of its relative simplicity and easiness of its usage, which is made possible by an ecosystem of open source tools like PHP MyAdmin.

MySQL is used for a variety of applications, but is most usually found on Web servers. A website that uses MySQL may also have Web pages that can get information from a database. Such pages are generally referred as "dynamic," which means the content of each page is created by a database when the page loads. Websites, which uses dynamic Web pages, are mostly recognized as database-driven websites.

3.4.3. Main system structure

The student registration structure is shown in Figure 16 and Figure 17.

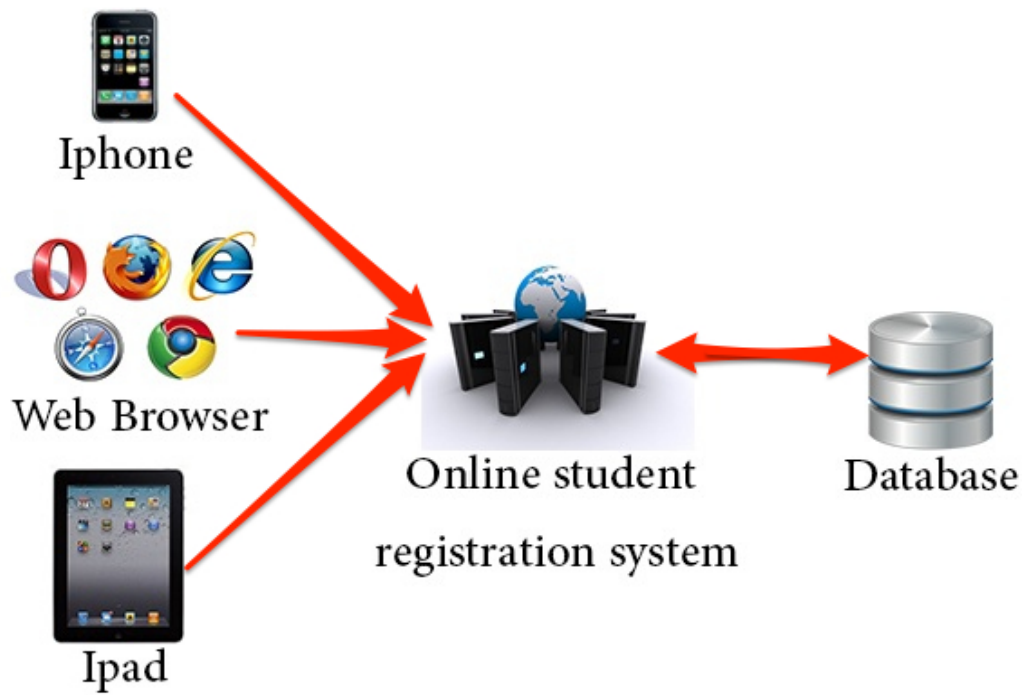


Figure 16 Main System Layout

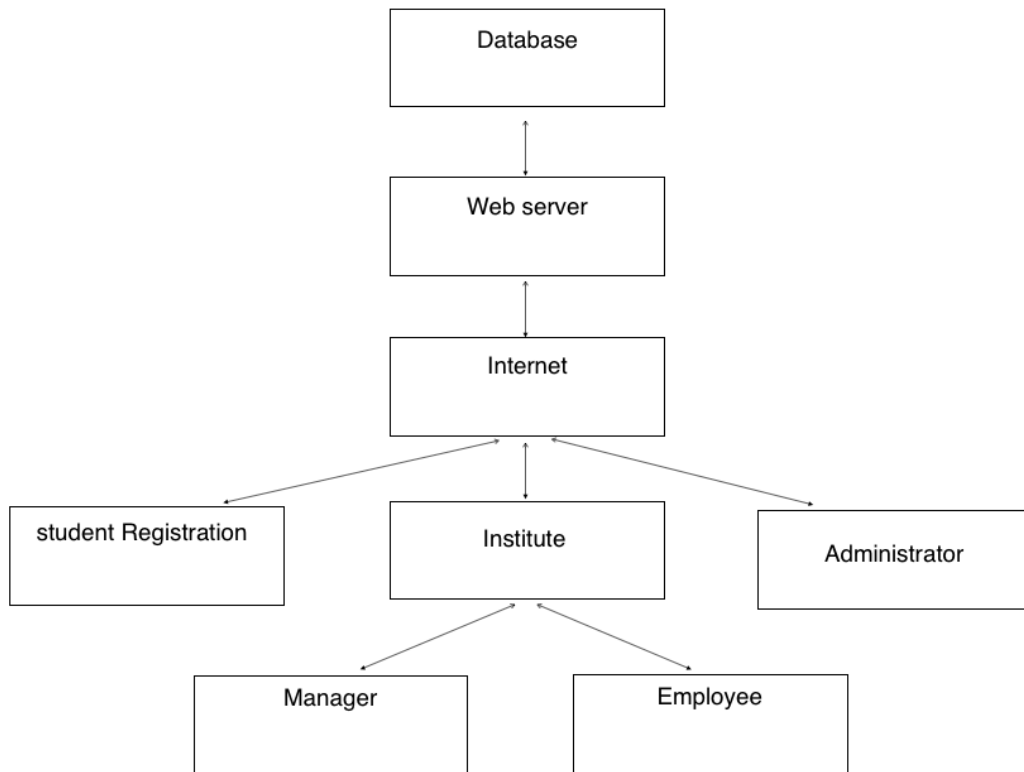


Figure 17 Main System Structure

It can be described as the main system structure by algorithms:

1. The algorithm 1 describes the main system login method.
2. The algorithm 2 describes the system security method.
3. The algorithm 3 describes the student registration method.
4. The algorithm 4 describes the follow-up status of the student.
5. The algorithm 5 describes the functions of an employee.
6. The algorithm 6 describes the manager role.
7. The algorithm 7 describes the administrator function.

Algorithm 1: Describes the main system login method.

Input: The username and password of the user.

Output: Enter to the main system.

Step 1: Check the user input.

- If ((valid input) and (secure input))
 - Check the user.
 - If (user ="Student")
 - Go to the student status page.
 - Else If (user ="Employee")
 - Go to an Employee page.
 - Else If (user ="Manager")
 - Go to the Manager Page.

Step 2: - else If not valid input or not secure input.

- Display error message or Invalid input.

Step 3: Stop.

Algorithm 2: describes the system security method.

Input: Check an SQL Injection attack and XSS Injection attack for the user login.

Output: Enter to the system.

Step 1: Check the security user input from SQL Injection attack.

- If (valid input)
 - Check an SQL Injection attack.
 - If (secure input)
 - Go to Step2.

Step: Check the security user input from XSS Injection attack.

- If (secure input)
 - Enter to the specified user page.
- Else If Display error message as invalid input or not secure input.

Step 3: Stop.

Algorithm 3: describes the student registration method.

Input: Student fills the application form.

Output: The student registered.

Step 1: - Enter the *Register* link in Home page

Step 2: - For each Field in the application form

- If not empty field.

- If (valid input)

- Continue **Step2**.

- Else if Display error message or Invalid input.

- else If Display error message or this field not filled.

Step 3: - Upload all required documents.

Step 4: - Complete the registration method and Save all Information to the database.

Step 5: Stop.

Algorithm 4: describes the follow-up status of the student.

Input: Student login and check the current status.

Output: Final student status.

Step 1: Check the user input.

- If (valid and secure input)
 - Check the user.
 - If (user ="Student")
 - Go to the student status page.
 - Go to Step 3
 - Display a certain message for this student about current status.

Step 2: - else If -Display error message or Invalid input.

Step 3: -Check the student information

- If (Complete Information)
 - Check the manager decision by check the student status in the database
 - If ("Primary accept")
 - Continue Step2.
 - Else If ("Conditional accept")
 - Continue Step2.
 - Else If ("Not accept")
 - Continue Step2.
 - Else If Display a message under processing.
 - Else If Display a message about No completed information.

Step 4: Stop.

Algorithm 5: describes the functions of an employee.

Input: Employee login.

Output: Completed employee functions.

Step 1: Check the user input.

- If (valid and secure input)
 - Check the user.
- If (user ="Employee")
 - Go to an Employee page.
 - Follow-up the register students information and check it.
 - If (Completed Information)
 - Send an information to the manager to make a final decision about the student status.
 - else If send a message to the student with
No complete information

Step 2: - else If Display error message or Invalid input.

Step 3: Stop.

Algorithm 6: describes the manager role.

Input: Manager login.

Output: Manager decision.

Step 1: Check the user input.

- If (valid and secure input)
 - Check the user.
- If (user ="Manager")
 - Go to the manager page.
 - Check the student status and make a decision.

Step 2: - else If not valid input or not secure input.

-Display error message or Invalid input.

Step 3: Stop.

Algorithm 7: describes the administrator function.

Input: Administration.

Output: An updated system.

Step 1: - Check the full system.

- Update the system.

Step 2: Stop.

3.4.4. Database Design

The main interface for working through MySQL database is to run the MySQL server in the XAMPP web server, Then it can be design main database tables and other SQL statements. In case of security, especially when it is needed to protect the interior database data from injection attacks. Even when an unauthorized person wants to finds/retrieve some data from the database, which cannot be opened because of the lack of the security PHP scripts from these injection attacks vulnerabilities.

3.5 Smartphone Application

3.5.1. Introduction

IOS (formerly iPhone OS) is a mobile operating system that is revealed by Apple Inc. and distributed solely for Apple hardware. Most of the organization's iDevices are empowered by this OS. It is initially revealed in 2007 for the iPhone, and has been expanded to promote other Apple tools such as the iPod Touch (September 2007), iPad (January 2010), iPad Mini (November 2012) and second-period Apple TV onward (September 2010). Beginning from June 2014, Apple's App Store included over 1.2 million IOS apps, 500,000 of that were optimized for iPad. These apps have jointly installed over 60 billion times. It possessed a 21% share of the smartphone mobile operating system segments transported within the last quarter of 2012, at the back of Android from Google. Until the half of the 2012, 410 tools were operated. 400 million tools had been sold till June 2012, as to the exclusive media event that is organized by Apple on September 12, 2012. The user interface of iOS relies on the context of direct manipulation, by utilizing multiple-touch gestures. Sliders, switches and buttons cover the interface control components. Some apps operate internal accelerometers in order to reply to quiver the tool (undo command is

the one mutual consequence) or revolving it in three size (re-routing from portrait to landscape style is the mutual consequence).

3.5.2. Main function

This mobile application can provide ability for the students to register and follow-up the case by using smart devices (Iphone and Ipad) and can be provided to the student for quick reach services which are provided by the university website.

3.5.3. Application development language

The used program for the application design is XCODE 6.0. Programming Language: Objective C use of objects. Objects and classes are fundamental in object-oriented programming (OOP) languages such as Objective-C, Java, C++, and many many more.

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1. Web Application

4.1.1. Home page

The home page of online registration system consists of login, student register, home, and other related links as shown in figure 18.

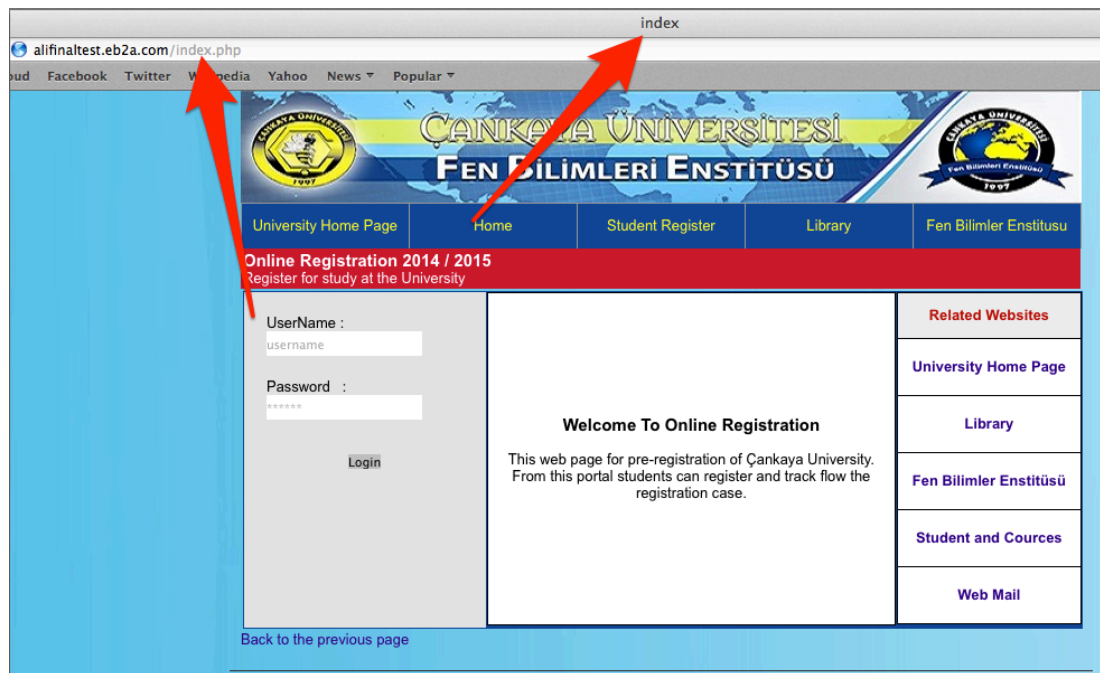


Figure 18 Home Page

4.1.2 Student Register

There are a hyperlink for student registration at the home page to transform students to the application form page as shown in figure 19,20,21,22, and 23.

alifinaltest.eb2a.com/register.php

University Home Page Home Student Register Library Fen Bilimler Enstitüsü

Note: * (Required)

Student Account Details

Username*

Password*

Confirm Password*

E-Mail*

Personal Details

First Name*

Middle Name*

Last Name*

Date of Birth*

Date of Application*

Gender* Male Female

Citizenship*

Telephone No.*

Passport No.*

Address*

Type of Study*

Background Information

The graduated university name*

Graduation date*

Grade point average*

Name of faculty*

Organization / Company in details*

Figure 19 Student Register

alifinaltest.eb2a.com/register.php

University Home Page Home Student Register Library Fen Bilimler Enstitüsü

Note: * (Required)

Student Account Details

Username*

Password*

Confirm Password*

E-Mail*

Personal Details

First Name*

Middle Name*

Last Name*

Date of Birth*

Date of Application*

Gender* Male Female

Citizenship*

Telephone No.*

Passport No.*

Address*

Type of Study*

Background Information

The graduated university name*

Graduation date*

Grade point average*

Name of faculty*

Organization / Company in details*

Name of department*

Current work information

Figure 20 Student Register

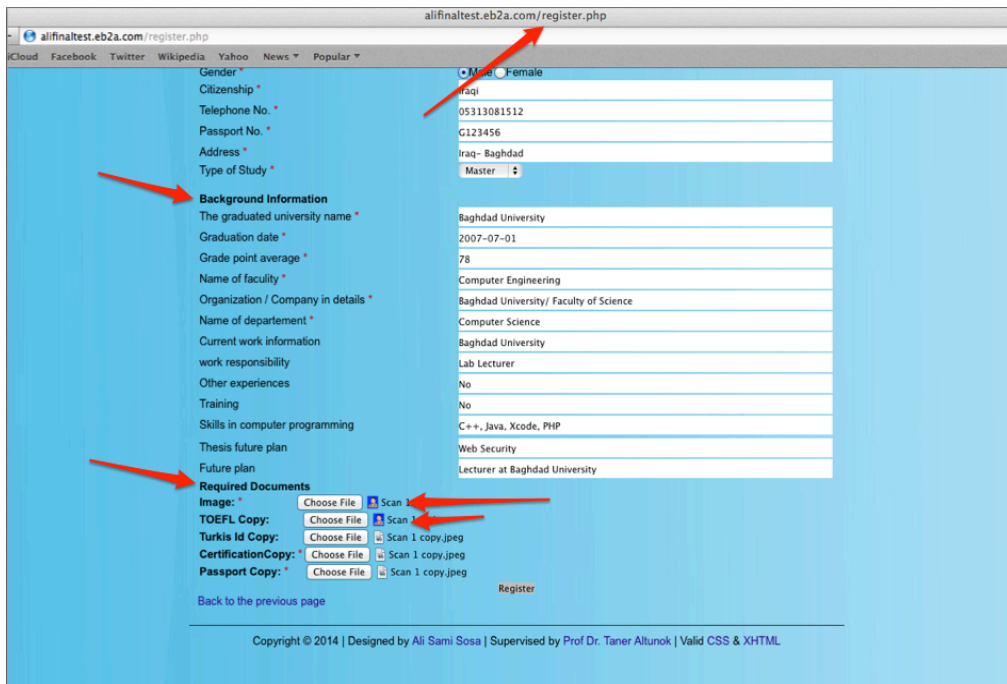


Figure 21 Student Register

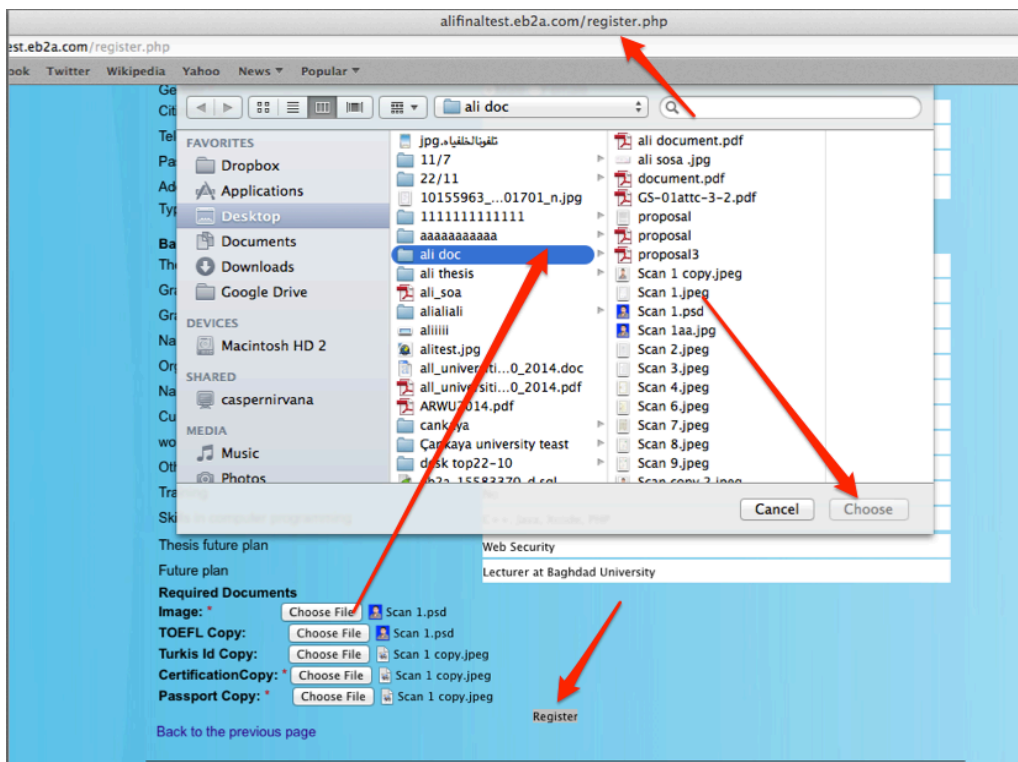


Figure 22 Student Register

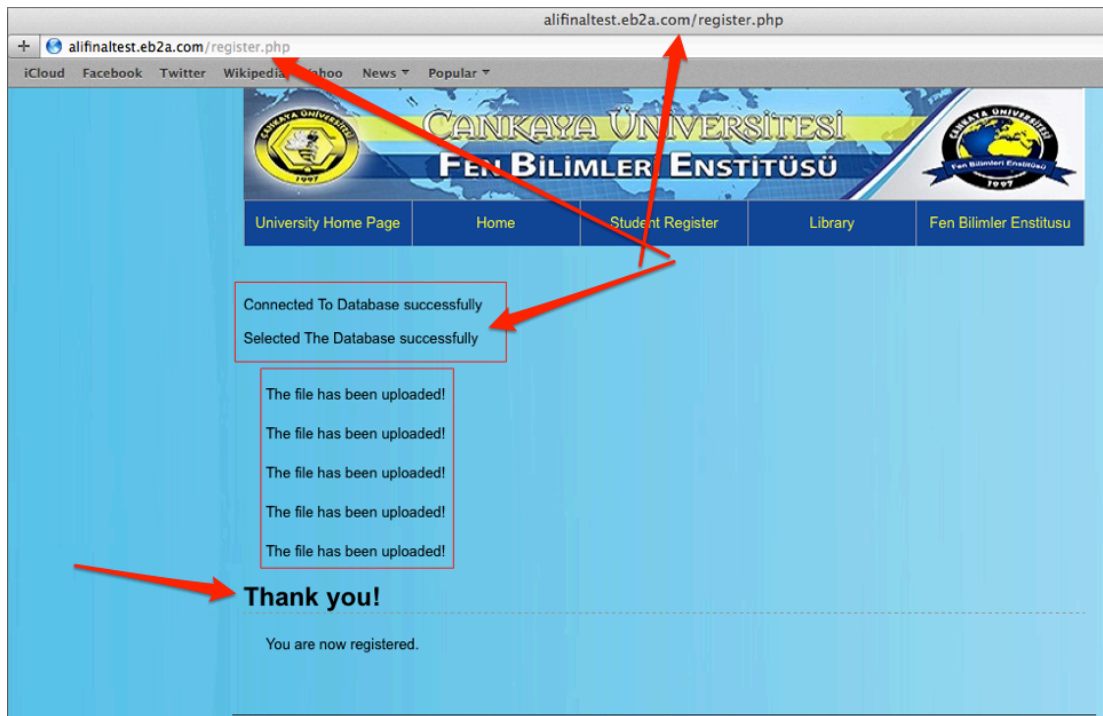


Figure 23 Student Register

4.1.3 Login

There are a hyperlink for login at the home page to transform users to the them pages; there are different users as:

1. User Page

The registered users can enter to them pages for track follow the registration case as show in figure 24, 25, and 26.

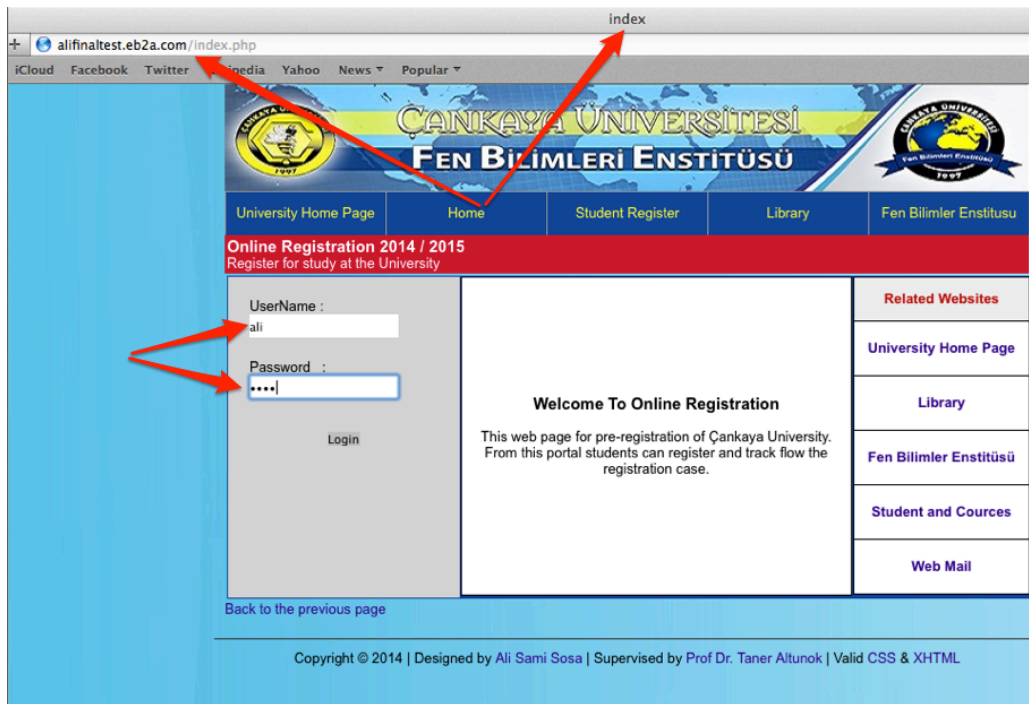


Figure 24 User login

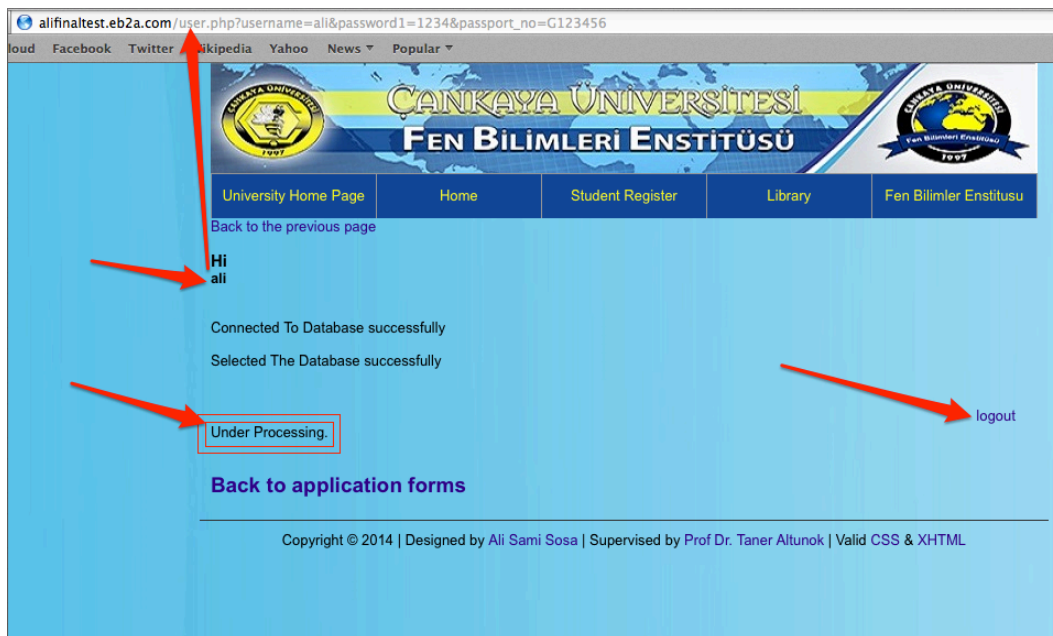


Figure 25 User logged in to the status page

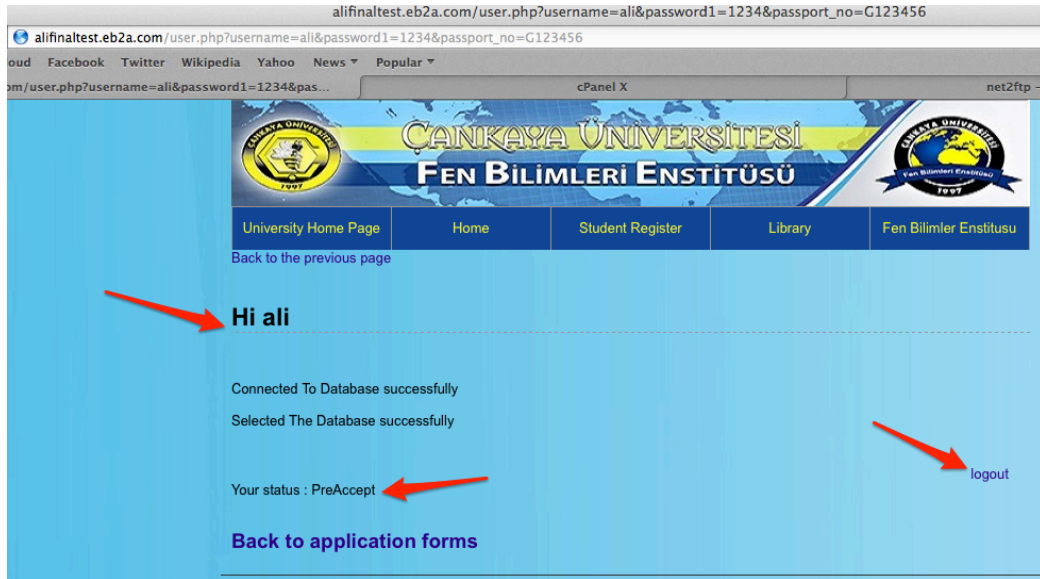


Figure 26 User logged in to the status page

2. Employee Page.

Employee can enter to page and make these functions as show in figure 27,28,29,30, and 31.

A. Show all application forms.

In this page can:

- Edit and check each user application form.
- Edit and check uploaded documents.
- Delete a user.
- Convert the completed user application forms to the manager list.
- Archive a user.

B. Search for application form of some user.

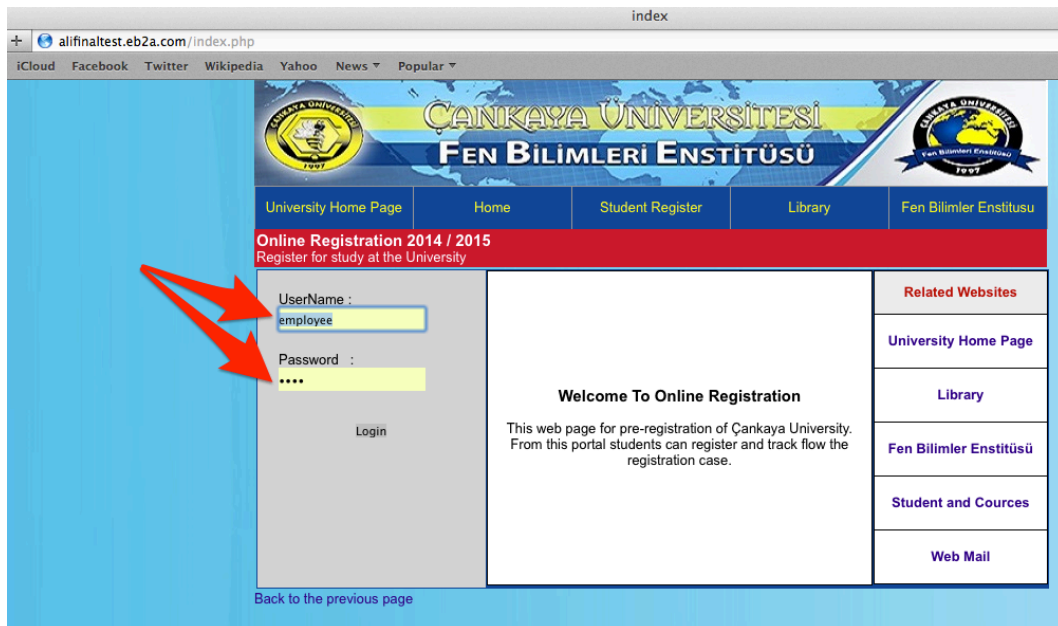


Figure 27 Employee login

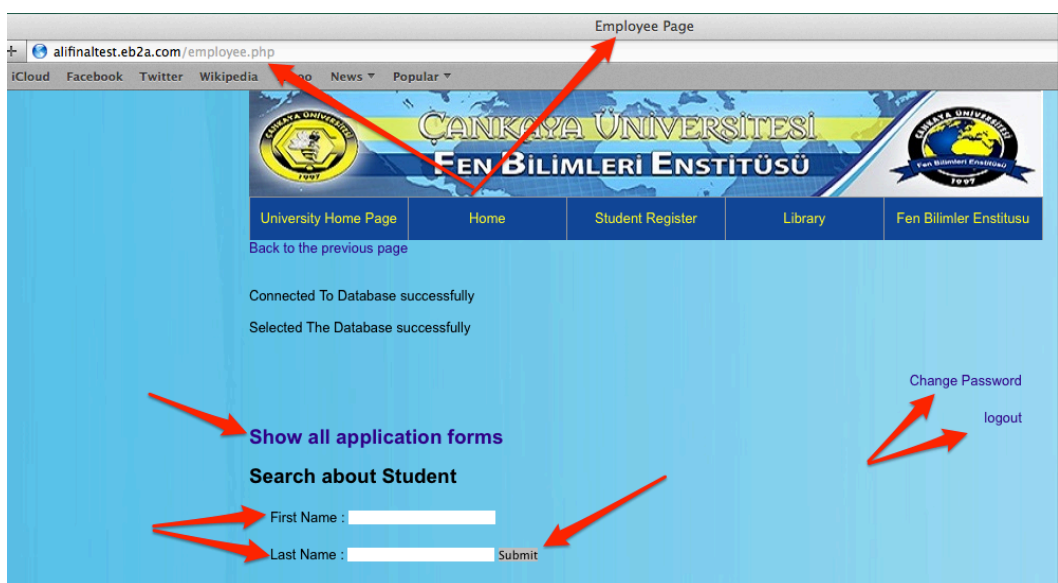


Figure 28 Employee logged in

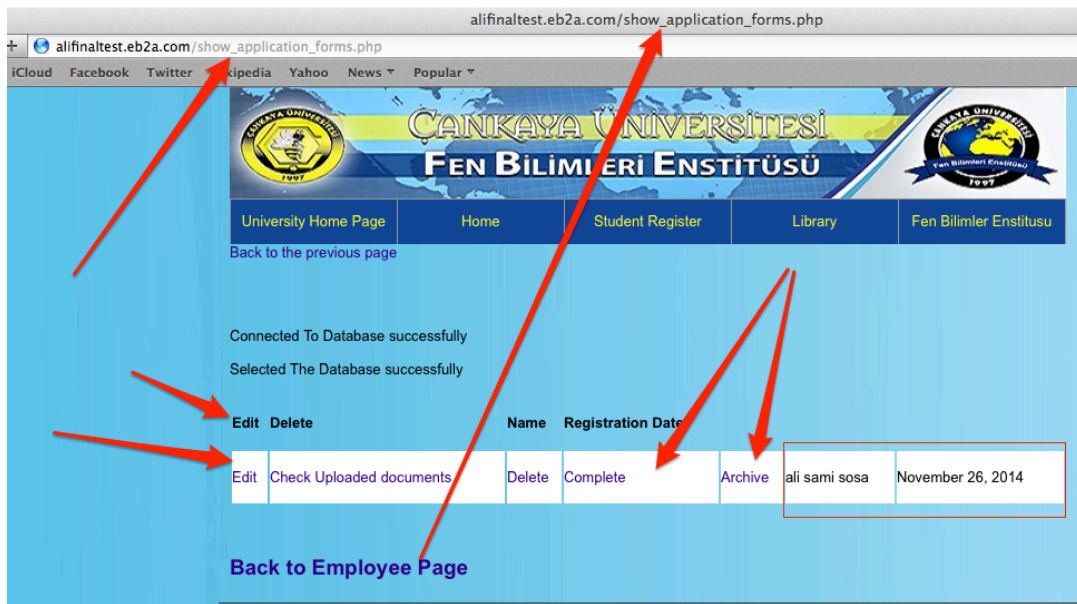


Figure 29 Employee functions

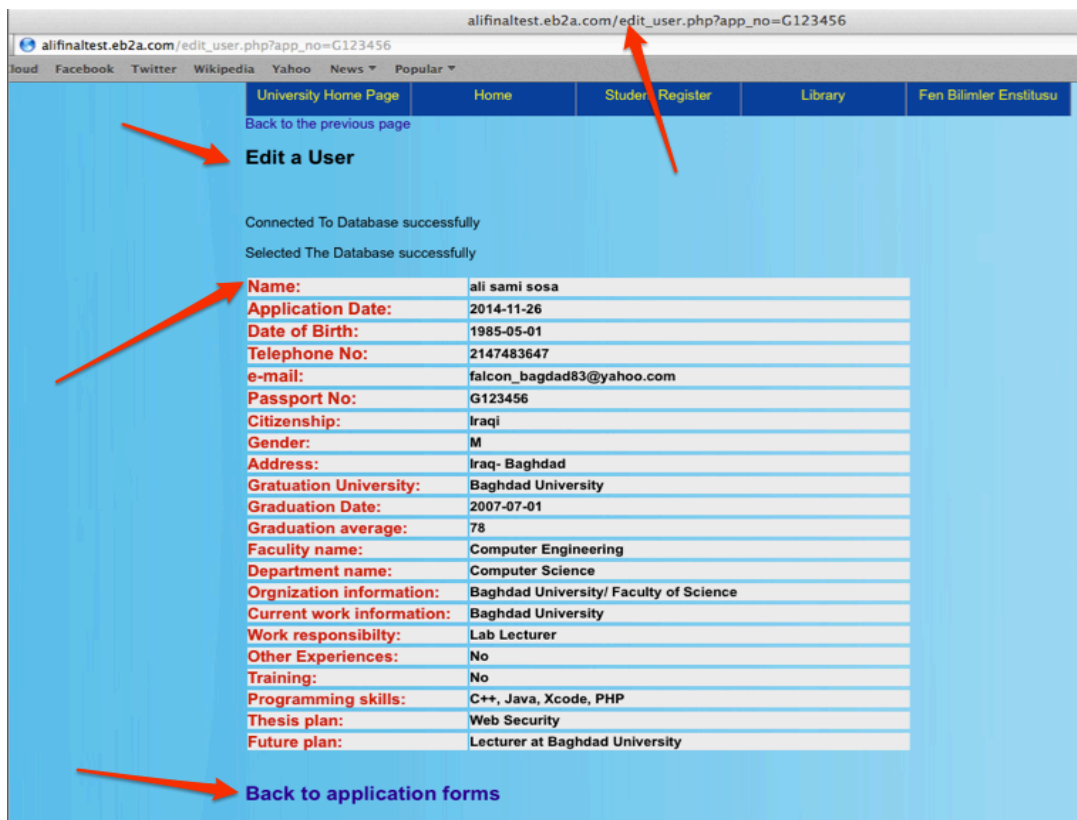


Figure 30 Employee edit user

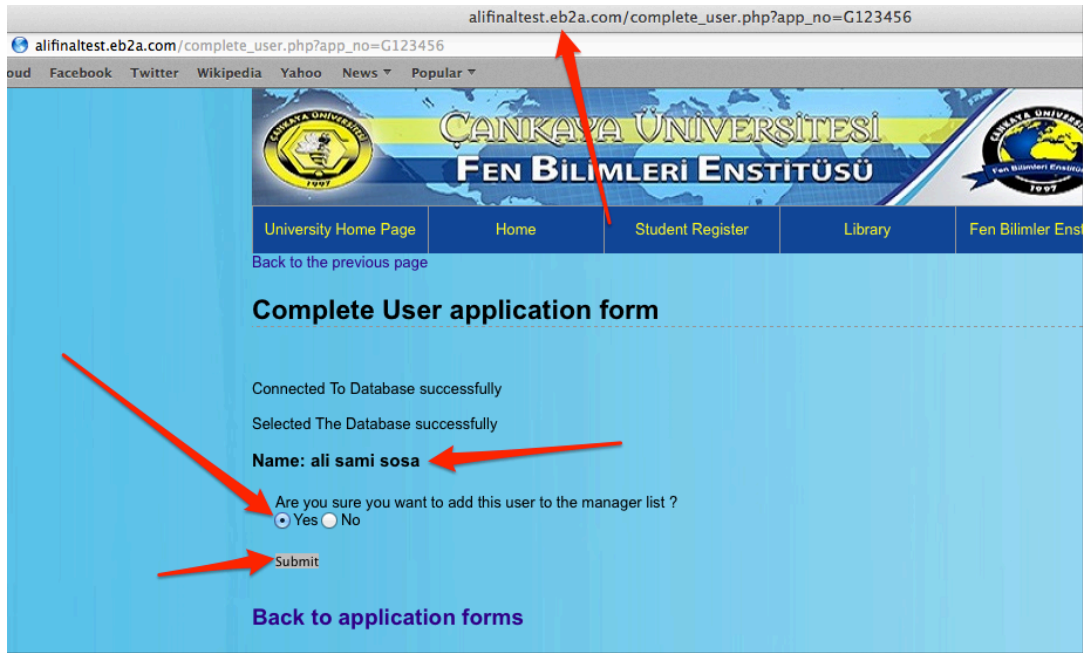


Figure 31 Employee convert to manager

3. Manager Page.

Manager can enter to page and make these functions as show in figure 32, 33, 34, 35, 36 and 37.

A. Show the completed application forms.

In this page can:

- Show the user application form.
- Make final Decision the user.

B. Search user application form.



Figure 32 Manager login

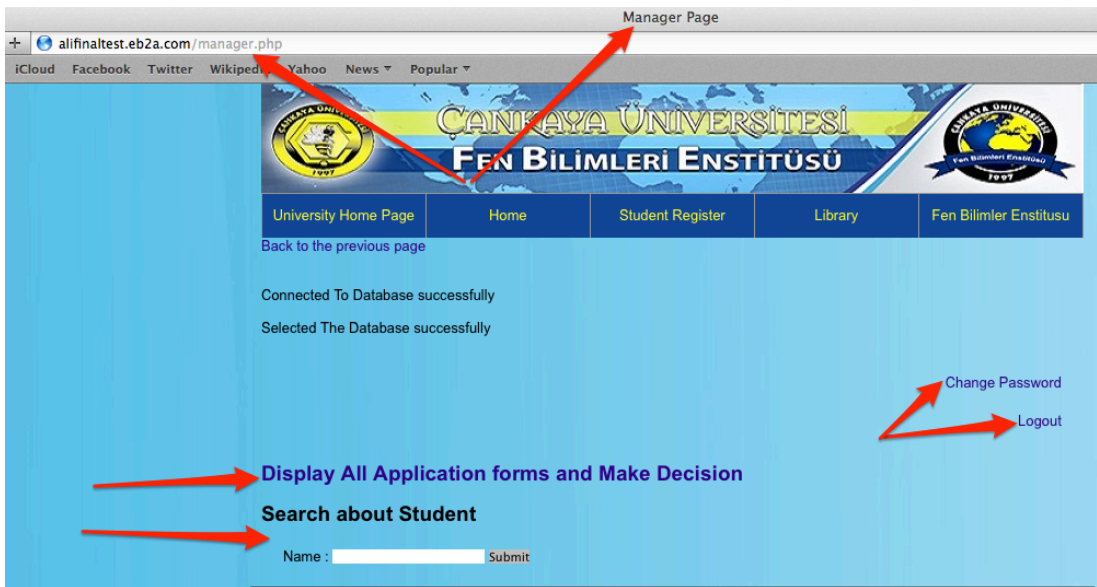


Figure 33 Manager page

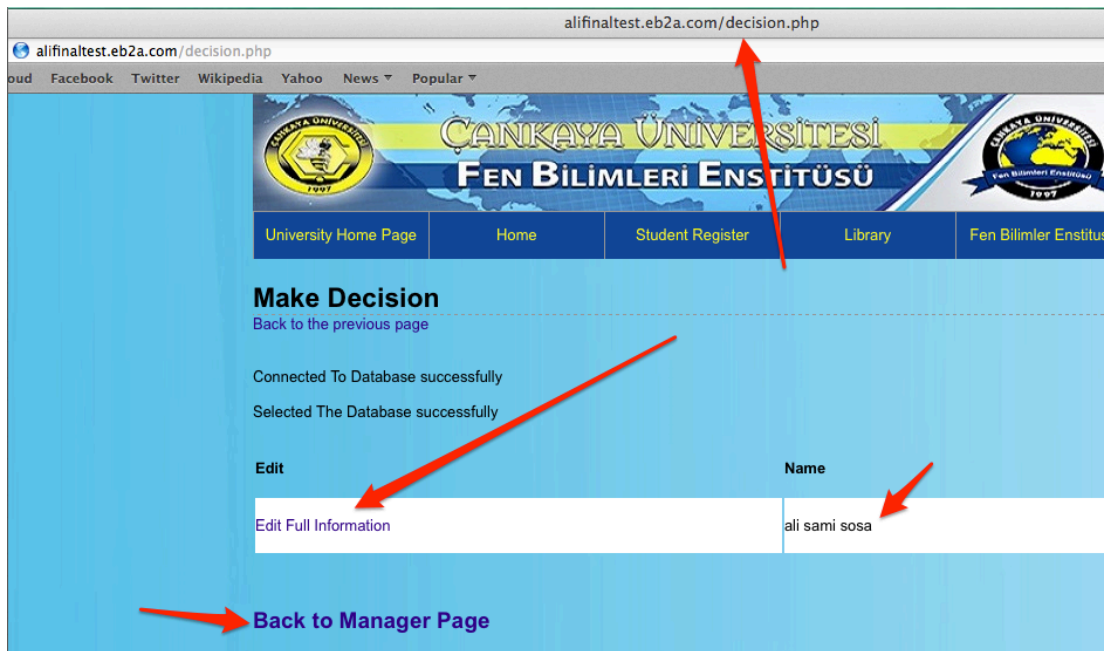


Figure 34 Edit users to manager

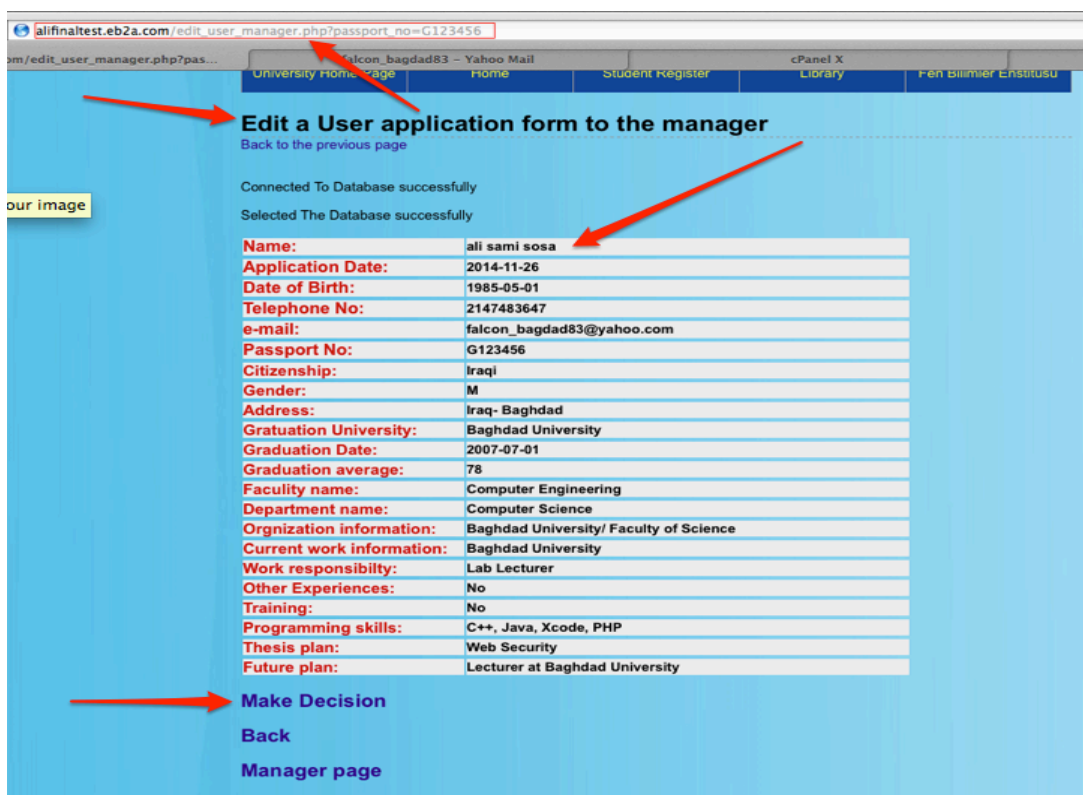


Figure 35 Edit full user information to manager

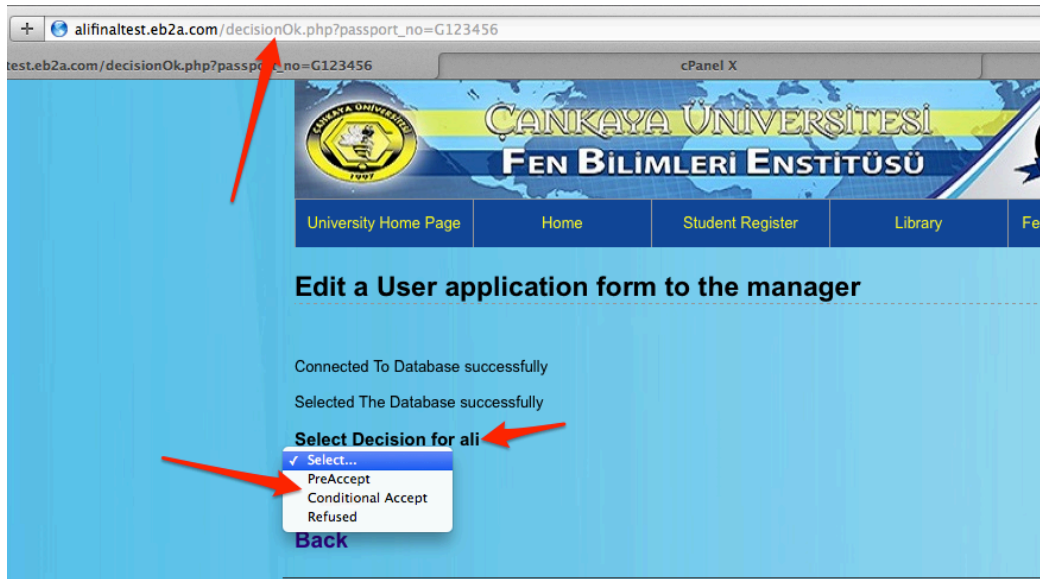


Figure 36 Manager decision

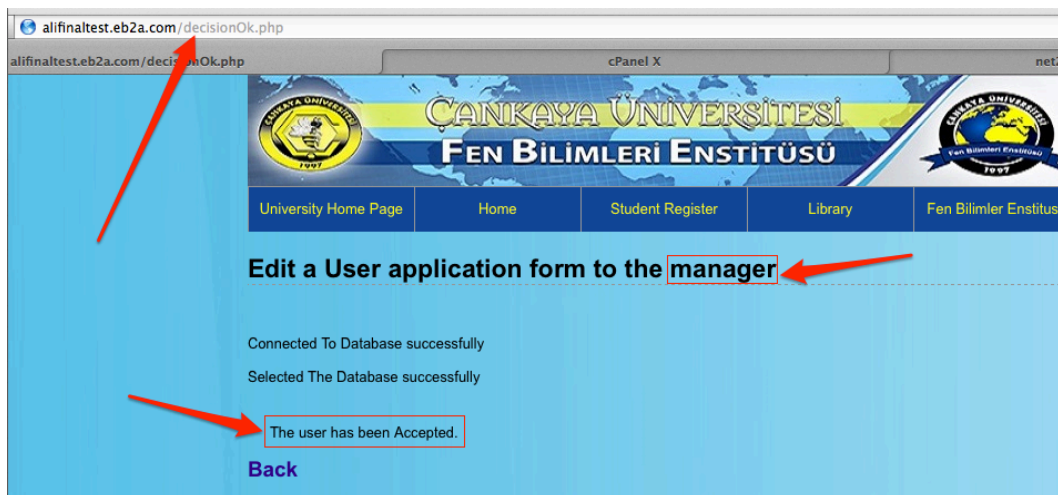


Figure 37 Decision done

4. Admin Page.

Administrator can enter to page and make these functions.

- A. Update, delete, insert and check the whole database.
- B. Backup the database.

5. Others.

Any other users attempt to login with invalid credential inputs or attempt to attack the database information it cannot be done as shown in figure 38, 39, 40, 41, 42, 43 and 44.

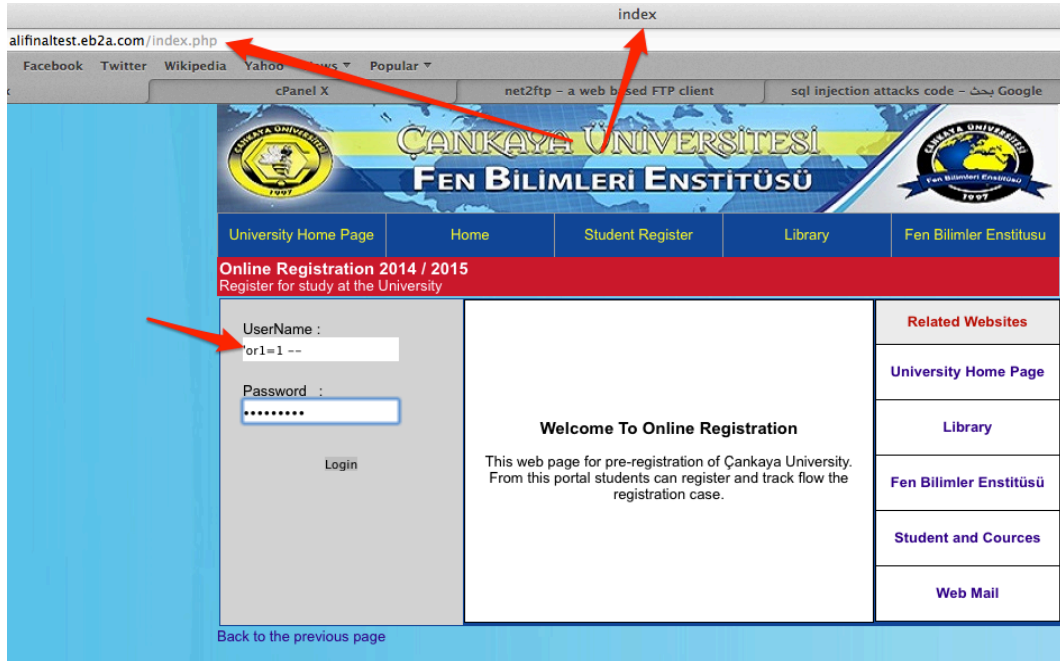


Figure 38 SQL injection attack

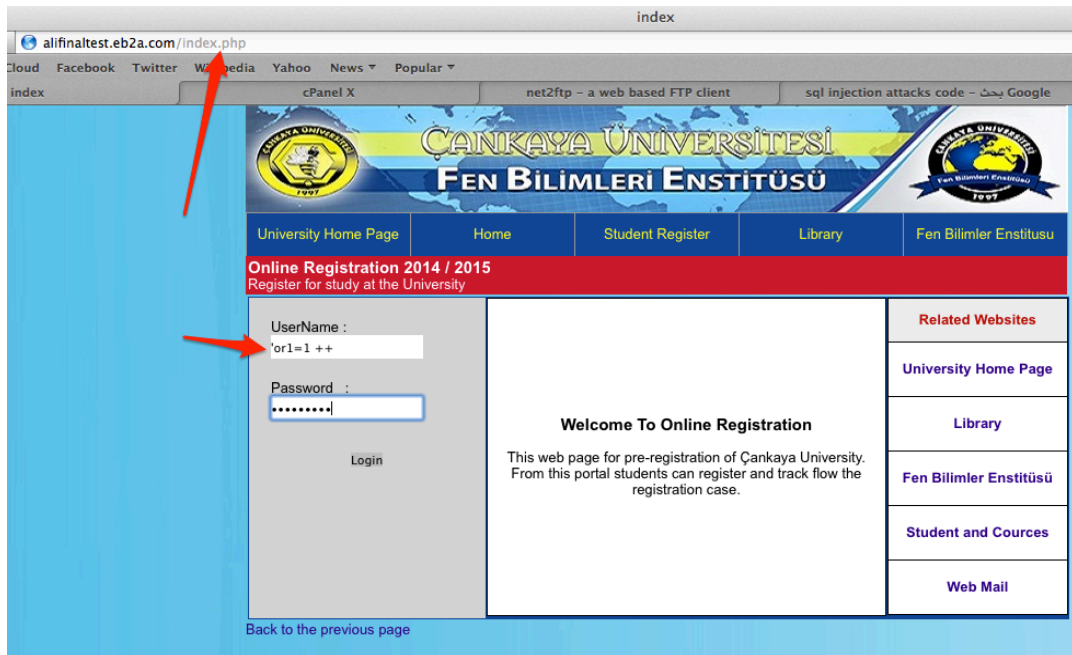


Figure 39 SQL injection attack

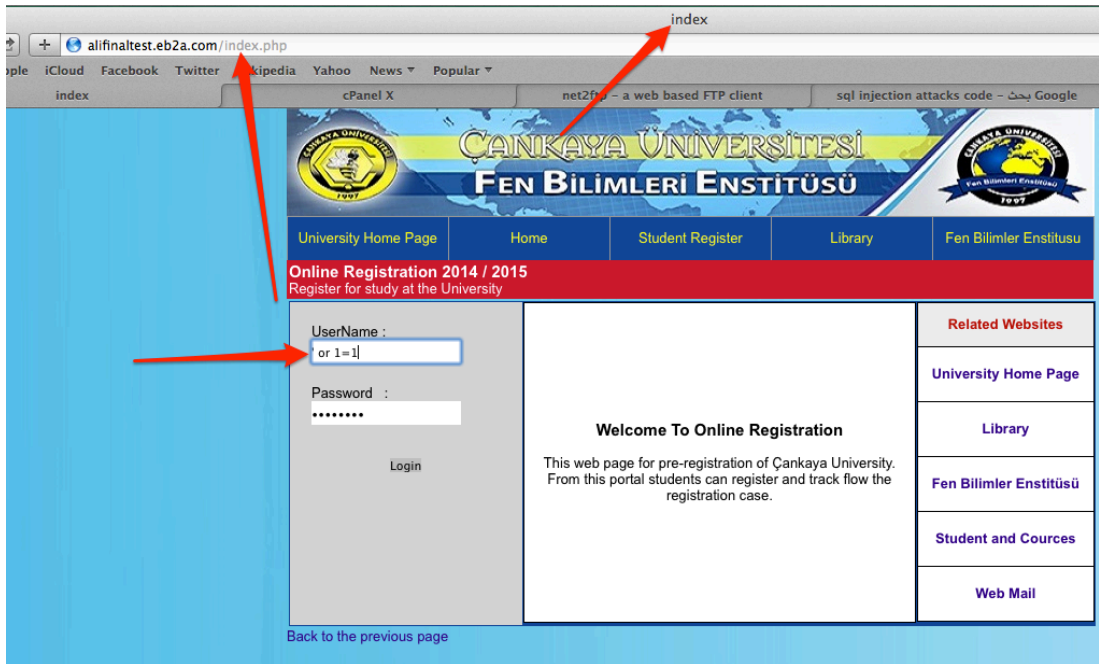


Figure 40 SQL injection attack



Figure 41 XSS injection attack

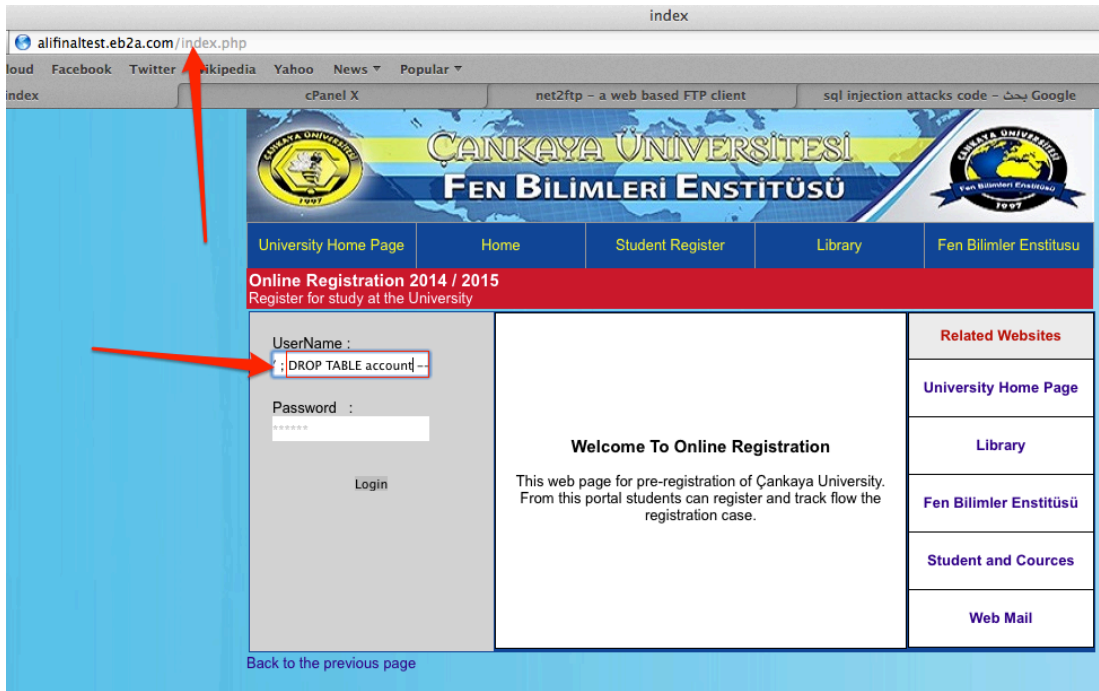


Figure 42 SQL injection attack

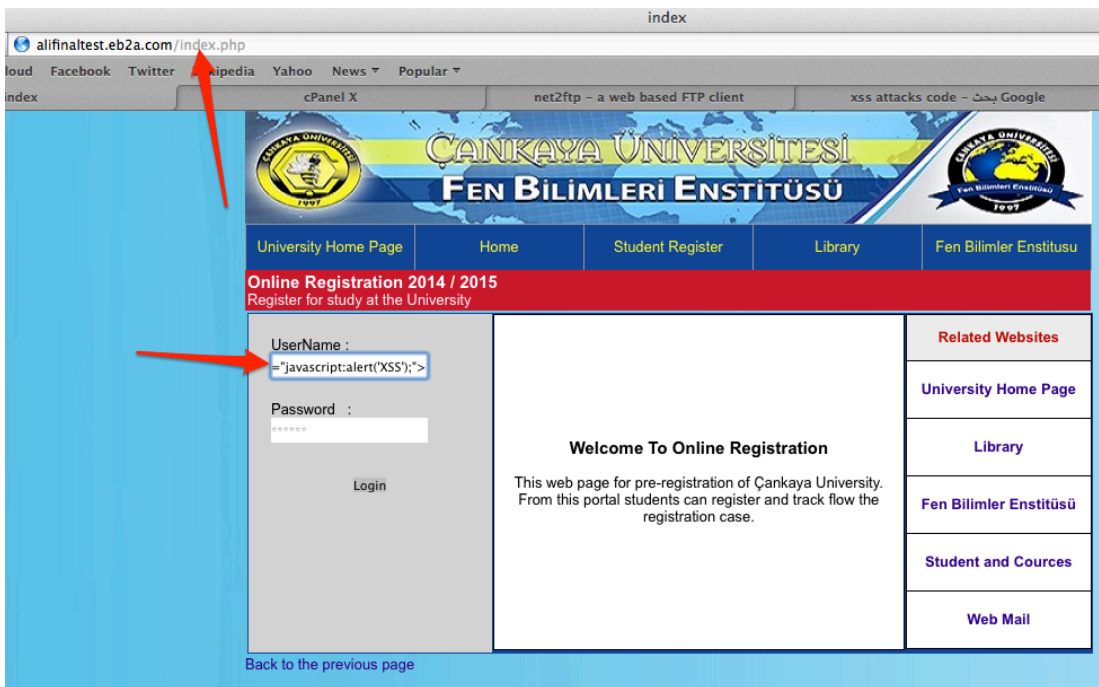


Figure 43 XSS injection attack

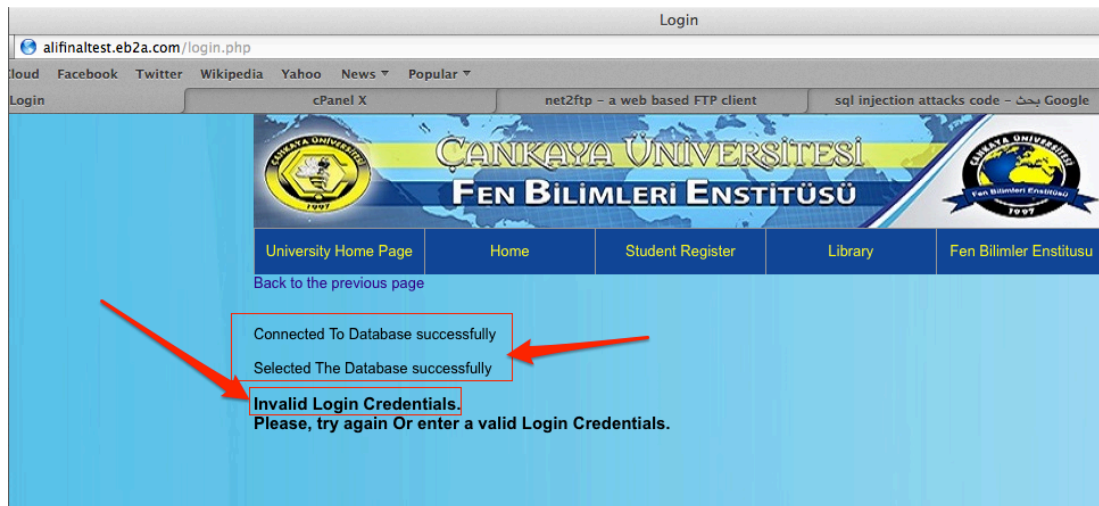


Figure 44 Invalid login or attack

4.2. Mobile Application

The sample of mobile application is shown in figure 45.



Figure 45 Mobile application

CHAPTER FIVE

CONCLUSIONS

5.1 Conclusions

The main concluding remark that is drawn from the work:

XSS and SQL injection are a related but fundamentally more difficult problem. Whereas database systems restrict command execution to a well-defined language, web browsers do not. Web browsers parse HTML permissively as a result of early software engineering decisions that seemed beneficial in the short term they enabled browsers to display poorly written HTML pages but have now made XSS more difficult to prevent. We examined browser input and discovered many subtle and undocumented ways for untrusted strings to invoke the JavaScript interpreter. We then constructed a policy that describes these ways using a regular language and employed our string-taint analysis to find XSS vulnerabilities. Our results are sobering: every manually written input validation routine that we analyzed and that allows any HTML fails to prevent XSS.

In the process of designing and implementing our static analysis, we found that many. Web applications use dynamic language features, and such features inhibit static analysis. We looked to testing as a complement to static analysis, but previous work on automated test input generation focuses on numeric values and pointer-based data structures, which languages like C and Java emphasize. Web scripting languages, like PHP, promote a style of programming that emphasizes strings and associative arrays. We designed and implemented an algorithm for automated test input generation for web applications. By incorporating values gathered from program executions, we model string operation semantics more precisely than static analyses have done, and we found vulnerabilities in real-world code that every available static analyzer failed to analyze.

1. System helps students who are planning to study at the university:
 - A. Registers by computer and mobile.

- B. Provide the ability of registration process.
- 2. System saves time and effort for institute's employee:
 - A. Track flow student's activities.
 - B. Archives student's information.
- 3. System gives reports for each user.
- 4. The system model can be used for prevention techniques against SQL injection and XSS injection attacks, which make the system more safe and strong.

5.2 Future Work

The following are some recommendations suggested for future works:

1. Developing the present system by performing additional features of registration system, such as mail box and add different language for the web application Interface (Turkish, Arabic and others).
2. Using another secure DB server, which can be used instead of MySQL database server such as Oracle database system.
3. Using distributed database.

REFERENCES

1. **Weinberger J., (2012)**, “*Analysis and Enforcement of Web Application Security Policies*”, vol. 1, pp. 489-495.
2. **Gary M. W., (2008)**, “*Techniques and Tools for Engineering Secure Web Applications*”, vol. 19, pp. 656-663.
3. **Nugroho L. E., (2001)**, “*A Context-Based Approach for Mobile Application Development*”, vol. 1, pp. 98-109.
4. **Uzi B. L. and Donald S., (2003)**, “*Web Application Security : A Survey of Prevention Techniques*”, vol. 1, pp. 568 – 574.
5. **Yoshihama M., Tanaka H., Hokao S., and Furukawa T., (2003)**, “*Design and Implementation of Web Based Registration Completion System at Universities using Postgre SQL*”, vol. 1, pp. 1687–1692.
6. **Merlo E., Letarte D., Antoniol G., & Montr C., (2007)**, “*Automated Protection of PHP Applications Against SQL-injection Attacks*”, vol. 5, pp. 754-762.
7. **Lin J. C., and Chen J. M., (2007)**, “*The Automatic Defense Mechanism for Malicious Injection Attack*”. *7th IEEE International Conference on Computer and Information Technology (CIT 2007)*, vol. 1, pp. 709–714.
8. **Sulaiman J., (2008)**, “*Electronic Student Academic System (E-SAS) For Secondary School*”, vol. 5, pp. 211–216.
9. **Michael G. W., (2008)**, “*Security Techniques and Tools for Engineering Secure Web Applications*”, vol. 19, pp. 656-663.
10. **Bonné B., (2011)**, “*Improving Session Security in Web Applications*”. Vol. 1, *Opt. Laser Tech.*, vol. 41, pp. 285-288.
11. **Liu Y., Gao F. and Liu Y., (2012)**, “*Design and Implementation of Student Registration System for Universities*”. *2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, vol. 2, pp. 1760–1763.
12. **Mohammed H. A., (2012)**, “*Client Side Action Against Cross Site Scripting Attacks*” vol. 1 pp. 121-127.

13. **Hauzar D. and Kofron J., (2012)**, “*On Security Analysis of PHP Web Applications*”, IEEE 36th Annual Computer Software and Applications Conference Workshops, vol. 36, pp. 577–582.
14. **Manchanda A., (2012)**, “*Expense Tracker Mobile Application*”, vol. 5, pp. 211–216.
15. **Pacio R. D., (2013)**, “*Online Student Information System of Benguet State University*”, vol. 4 pp. 39–47.
16. **Spångmyr M., (2014)**, “*Development of an Open-Source Mobile Application for Emergency Data Collection*”, vol. 1, pp. 294-302.
17. **Buss M. and Gruhn V., (2014)**, “*Process-Aware Continuation Management In Web Applications. Science of Computer Programming*”, vol. 1, pp. 3–17.
18. **Carafa M. M. C., Tarabusi, G., and Kastelic V., (2014)**, “*SHINE: Web Application for Determining the Horizontal Stress Orientation. Computers & Geosciences*”, vol. 1, pp. 39–49.
19. **Ferrara E., DeMeo P., Fiumara G., and Baumgartner R., (2014)**, “*Web Data Extraction, Applications and Techniques: A Survey. Knowledge-Based Systems*”, vol. 1, pp. 301–323.
20. **Doğan S., Betin-Can A., & Garousi, V., (2014)**, “*Web Application Testing: A Systematic Literature Review. Journal of Systems and Software*”, vol. 1, pp. 174–201.
21. **Metze F., Anguera X., Barnard E., Davel M., and Gravier G., (2014)**, “*Language Independent Search in Media Eval’s Spoken Web Search Task. Computer Speech & Language*”, vol. 28, pp. 1066–1082.
22. **Sadat-Mohtasham S. H., and Ghorbani A., (2008)**, “*A Language for High-Level Description of Adaptive Web Systems. Journal of Systems and Software*”, vol. 7, pp. 1196–1217.
23. **Walker J. D., and Chapra S. C., (2014)**, “*A Client-Side Web Application for Interactive Environmental Simulation Modeling. Environmental Modelling & Software*”, vol. 1, pp. 49–60.
24. **Thuraisingham B., Clifton C., Gupta a., Bertino E., and Ferrari E., (2001)**, “*Directions for Web and E-Commerce Applications Security*”. Proceedings Tenth IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises. WET ICE, vol. 1, pp. 200–204.
25. **Leff A., and Rayfield J. T., (2008)**, “*WebRB: A Different Way to Write Web Applications. IEEE Internet Computing*, vol. 12, pp. 52–61.
26. **Guide, S., & Yank, B. K. (2005)**, Book Reviews PHP and SQL Made Simple, vol. 8, pp. 6–20.

27. **Beja I. P., Superior E., Tecnologia D., and Iii R. A., (2008),** "*Instrumentation Remote Control through Internet with PHP*" vol. 1, pp. 14–16.
28. **Boomsma H., Hostnet B. V., and Gross, H. G., (2012),** "*Dead Code Elimination for Web Systems Written in PHP: Lessons learned from an Industry Case*". 2012 28th IEEE International Conference on Software Maintenance (ICSM), vol. 1, pp. 511–515.
29. **Cui W., Huang L., Liang L. and Li J., (2009),** "*The Research of PHP Development Framework Based on MVC Pattern*". Fourth International Conference on Computer Sciences and Convergence Information Technology, vol. 1, pp. 937–949.
30. **Gauthier F. and Merlo E., (2012),** "*Alias-Aware Propagation of Simple Pattern-Based Properties in PHP Applications*". IEEE 12th International Working Conference on Source Code Analysis and Manipulation, vol. 1, pp. 44–53.
31. **Curphey M. and Arawo R., (2006),** "*Web Application Security Assessment Tools*". IEEE Security & Privacy Magazine, vol. 4, pp. 32–41.
32. **Fonseca J., Vieira M., and Madeira H., (2009),** "*Vulnerability & Attack Injection for Web Applications*". IEEE/IFIP International Conference on Dependable Systems & Networks, vol. 8, pp. 93–102.
33. **Ludinard R., Totel E., Tronel F., Nicomette V., Kaaniche M., Alata E., and Bachy Y., (2012),** "*Detecting Attacks Against Data in Web Applications.*" 7th International Conference on Risks and Security of Internet and Systems (CRiSIS), vol. 2, pp. 1–8.
34. **Criscione C., Salvaneschi G., Maggi F., and Zanero S., (2009),** "*Integrated Detection of Attacks Against Browsers, Web Applications and Databases.*" European Conference on Computer Network Defense, vol. 1, pp. 37–45.
35. **Chris Snyder T., (2010),** "*Pro PHP Security From Application Security Principles to the Implementation of XSS Defenses*", vol. 1, pp. 201-213.
36. **Hydara I., Sultan A. B. M., Zulzalil H., and Admodisastro N., (2014),** "*Current State of Research on Cross-Site Scripting (XSS) – A systematic literature review. Information and Software Technology*", vol. 1, pp. 114–116
37. **Mason A., (2012),** "*Caught in the Cross-Site Scripting Fire. Network Security*", vol. 5, pp. 5–9.
38. **Sun Y., and He D., (2012),** "*Model Checking for the Defense Against Cross-Site Scripting Attacks*". International Conference on Computer Science and Service System, vol. 1, pp. 2161–2164.

39. **Lan D., and Shuting W., (2013)**, “*Analysis and Prevention for Cross-Site Scripting Attack Based on Encoding*”, IEEE 4th International Conference on Electronics Information and Emergency Communication, vol. 1, pp. 102–109.
40. **Louw M. T., and Venkatakrisnan V. N., (2009)**, “*Blueprint: Robust Prevention of Cross-Site Scripting Attacks for Existing Browsers*”. 30th IEEE Symposium on Security and Privacy, vol. 1, pp. 331–346.
41. **Wassermann G., and Su Z., (2008)**, “*Static Detection of Cross-Site Scripting Vulnerabilities.*” Proceedings of the 13th International Conference on Software Engineering - ICSE ’, vol. 12, pp. 171-181.
42. **Kie A., Guo P. J., and Ernst M. D., (2009)**, “*Automatic Creation of SQL Injection and Cross-Site Scripting Attacks*”, vol. 1, pp.199–209.
43. **Jang Y.S., and Choi J.Y., (2014)**, “*Detecting SQL Injection Attacks Using Query Result Size. Computers & Security*”, vol. 1, pp. 44, 104–118.
44. **Sadeghian A., Zamani M., and Manaf A, (2013)**, “*A Taxonomy of SQL Injection Detection and Prevention Techniques*”. International Conference on Informatics and Creative Multimedia, vol. 1, pp. 53–56.
45. **Sadeghian A., Zamani M., and Abd-Manaf A, (2014)**, “*SQL Injection Vulnerability General Patch Using Header Sanitization*”. 2014 International Conference on Computer, Communications, and Control Technology, vol. 14, pp. 239–242.
46. **Kiani M., Clark A., and Mohay G., (2008)**, “*Evaluation of Anomaly Based Character Distribution Models in the Detection of SQL Injection Attacks*”, third International Conference on Availability, Reliability and Security, vol. 1, pp. 47–55.
47. **Shanmuganeethi S. V., Shyni S. C., Swamynathan S., (2009)**, “*SBSQLID: Securing Web Applications with Service Based SQL Injection Detection*”. International conference on Advances in Computing, Control, and Telecommunication Technologies, vol. 1, pp. 702–708.
48. **Wu, H. (2011)**. “*Test SQL Injection Vulnerabilities in Web Applications Based on Structure Matching*”. Proceedings of International conference on computer science and network technology, vol. 1, pp. 935–940.
49. **Zhang K. X., Lin C. J., Chen S. J., Hwang Y., Huang H. L., and Hsu F. H., (2011)**, “*TransSQL: A Translation and Validation-Based Solution for SQL-Injection Attacks*”. First International Conference on Robot, Vision and Signal Processing, vol. 1, pp. 248–257.

- 50. Tian W., Yang J. F., Xu J., and Si G. N. (2012),** “*Attack Model Based Penetration Test for SQL Injection Vulnerability*”. IEEE 36th Annual Computer Software and Applications Conference Workshops, vol. 1, pp. 589–594.
- 51. Ntagwabira L., and Kang S. L., (2010),** “*Use of Query Tokenization to Detect and Prevent SQL Injection Attacks*”. 3rd International Conference on Computer Science and Information Technology, vol. 1, vol. 1, pp. 438–440.
- 52. Hannah Wong (2014),** “*Apple Brings Vibrant Colors & iSight Camera to Most Affordable iPod Touch Model*”(http://www.apple.com/pr/library/2014/06/26Apple-Brings-Vibrant-Colors-iSight-Camera-to-Most-Affordable-iPod-touch-Model.html). Apple Inc. June 26, 2014.
- 53. Bowman T .R., (2014),** “*Apple Announces Updates to iTunes U*”. Apple Inc. (http://www.apple.com/pr/library/2014/06/30Apple-Announces-Updates-to-iTunes-U.html). June 26, 2014.
- 54. Brateris D., Bedford D., Calhoun D., Johnson A., Kowalski N., Martino K., and Krchnavek R. R., (2011),** “*IOS Hardware as a Sensor Platform: DMM Case Study*”. IEEE Sensors Applications Symposium, vol. 1, pp. 308–317.
- 55. Hui N. M., Chieng L. B., Ting W. Y., Mohamed H. H., and Hj Mohd Arshad M. R., (2013),** ” *Cross-Platform Mobile Applications for Android and IOS*”. 6th Joint IFIP Wireless and Mobile Networking Conference (WMNC), vol. 1, pp. 1–9.
- 56. Singh C., Roy A. J., and Roy-Chowdhuri S., (2014),** “*Image-Based Cytopathology Reference App on IOS Platform*”. Journal of the American Society of Cytopathology, vol5, pp. S11–S18.
- 57. Naito K., Mori K., Kobayashi H., Kamienuo K., Suzuki H., and Watanabe A., (2014),** “*End-to-End IP Mobility Platform in Application Layer for AOS and Android OS*”. IEEE 11th Consumer Communications and Networking Conference (CCNC), vol. 1, pp. 92–97.

APPENDIX

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Sosa, Ali Sami Sosa

Nationality: Iraqi

Date and Place of Birth: 01 May 1985, Bagdad

Marital status: Married

Phone: 05313081512

Mail: falcon_bagdad83@yahoo.com



EDUCATION

Degree	Institution	Year of Graduation
M.Sc.	Çankaya University Mathematics and Computer Science	2014
B.Sc.	AL-Yemenia University Engineering and Computer Science	2008

FOREIGN LANGUAGES

Arabic, English.