1. INTRODUCTION

1.1 WEB PAGE, WEB SITE AND WEB APPLICATION

In the recent past, a factor that has been a major attraction among people is the Internet. File sharing mechanisms and message forum services provided increasingly powerful mechanisms for the information exchange. The credit for the exponential growth of the Internet can be given to personal publishing. The fundamental technology behind the web is relatively simple. A web server is a computer that is connected to the Internet and this serves the documents to the requesting clients. In earlier days, a little knowledge of HyperText Markup Language (HTML) was sufficient to create a web site. In the beginning, web site was just loosely connected sets of web pages, branching hierarchically from a home page. Now, web site is much more than just an assembly of web pages. Maintaining thematic consistency of content, having a common look and feel across all its pages, organization and interconnection between the contents make a website more complex issue. Earlier, most of the information shared by these web sites was mostly static and only a few of them were dynamic. The increase of dynamic information services changed this scenario and resulted in the dawn of the dynamic web. New services evolved and these services included search engines, scripts and packages. These services connected web sites to relational database. A client–server application that uses a web browser as its client program and delivers interactive services through web servers distributed over the internet is called a web application[1]. A web application can present dynamically tailored content based on request parameters, tracked user behaviors and security considerations.

The Internet has simplified the life style of people and has replaced the authentic administrative procedures that were otherwise in practice. Internet based software applications also called web applications have grown exponentially facilitating the rapid Internet growth. Web Applications are
developed using three tier web architecture, with the bottom tier being the database tier, the middle tier being the application tier, and the client tier at the top. Figure.1.1 shows the three tier architecture of the web application.

![Figure.1.1: Web Application Architecture](image)

Web browsers are the client tier, which functions as the graphical user interface. The web server administers the business logic and the database server administers the stored data. Business to customer data exchanges, like the online shopping, banking, gaming, ticket booking, travel booking and such others have been made feasible by the web applications. Web applications use the TCP/IP protocol for data communication. TCP/IP refers to two of the most important protocols within the suite, namely Transmission control Protocol (TCP) and Internet Protocol (IP).

### 1.2 TCP/IP SUITE

The protocol layers associated with TCP/IP are [1]:

1. The Network Interface layer
2. The Internet layer
3. The Transport layer
4. The Application layer
The Network Interface layer is responsible for the lowest level of data transmission within TCP/IP, facilitating communication with the underlying physical network.

The Internet layer provides mechanisms for intercommunication, controlling message routing, validity checking, composition and decomposition of message headers.

The transport layer provides message transport services between applications running on remote systems. This is the layer in which TCP operates. TCP provides reliable, connection–oriented message transport. Some services that do not require reliability associated with TCP make use of User Datagram Protocol (UDP).

The Application layer operates the services associated with the Internet. This is the layer in which HyperText Transfer Protocol (HTTP) operates. The HTTP uses the client-server paradigm. The HTTP client sends a request to the HTTP server and the request is received, processed and executed by the server. The Server returns an HTTP response message. Hence, communication is in the form of request–response interactions.

1.3 HYPERTEXT TRANSFER PROTOCOL

Transfer protocols can be of two types, namely stateless protocols like the HTTP and stateful protocols like File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP) and Post Office Protocol (POP) [2].

The stateful protocol treats the sequence of related commands as single interactions. The server must maintain a state of its interactions with the client throughout the transmission of successive commands, until the interaction is terminated. Sequences of transmitted and executed commands are called a session. They have a persistent connection, with multiple sessions. When the connection is terminated, all the sessions expire.
In contrast to the stateful protocol, HTTP is a stateless protocol, which means, each HTTP exchange consists of single request and single response. The client and the server are not required to maintain state between transmitted commands. HTTP uses a message based model in which a client sends a request message and the server returns a response message. This is as depicted in Figure 1.2.

**Figure 1.2: HTTP Request –Response Circuit**

- **HTTP REQUESTS**

An HTTP request message consists of a request method with Uniform Resource Identifier (URI) and version of the protocol, followed by one or more headers each in a separate line describing the client machine platform, then a mandatory blank line and a request message body represented in HTML code. An example of an HTTP request message is shown in Figure 1.3.

```
GET/ books/ search.asp?q=wahh HTTP/1.1
Accept: image/gif, image/xbitmap, image/jpeg, image/jpeg, application/vnd.ms-excel, application/msword, */*
Referer: http://wahh-app.com/books/default.asp
Accept-Language: en-gb, en-us; q=0.5
Accept-Encoding: gzip, deflate
User-Agent : Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)
Host: wahh-app.com
Cookie: lang=en; JSESSIONID=00000tt8rkJ7joMx44S2v_vsnlc 502
```

**Figure 1.3: Sample HTTP Request**
HTTP’s security mechanism uses basic authentication technique like Base64. They are not encrypted or hashed but just encoded.

- **HTTP RESPONSE**

  An HTTP response message consists of protocol name, with response status and description, followed by one or more response headers each in a separate line explaining the server machine platform, followed by a mandatory blank line, followed by response body represented in HTML code. An example of an HTTP request message is as shown in Figure 1.4.

```
HTTP/1.1 200 OK
Date: Mon, 25 Jan 2016 14:45:35 GMT
Server:IBM_HTTP_SERVER/1/3/26/2 Apache/1/3/26 (unix)
Set-Cookie: tracking=00000t8rk7joMx44S2v
Pragma: no-cache
Content-Type: text/html; charset = ISO-8859-1
Content-Language:en-US
Content-Length:245435

<!DOCTYPE html PUBLIC "-//w3c//DTD HTML 4.01 Transitional//EN">
<html lang="en">
<head>

</head>

....
```

**Figure 1.4: Sample HTTP Response**

**1.4 SECURITY ISSUES OF INTERNET**

The glory of the Internet and its merits are being highly masked by the drawback associated with it. For more than a decade, organizations have been dependent upon security measures at the boundary of the network, such as firewalls, in order to protect the web application. However, now more attacks are targeting security flaws in the design of web applications. These threats originate from non-trusted client access points, session-less protocols, the general complexity of web technologies, and network-layer insecurity. Usually, in web applications, client software cannot always be controlled by the application owner. Therefore, input from a client running the software cannot be completely trusted and processed directly. An attacker
can create an identity to look like a legitimate client, duplicate user’s identity, or create fraudulent messages and cookies. The prime issue is Internet vulnerability, leading to data modification and data thefts. Some of the key problem factors in web application field [2] are:

i. Awareness of the security flaws is less.

ii. Nature of defect is unique for each application.

iii. Possibility to create a powerful application from scratch in a short period by a novice programmer.

iv. During development phase stability and functionality of the application preferred over testing for security.

v. Old technologies adapted for new requirements.

1.5 RELATED TERMINOLOGIES

• THREAT

A Threat is a piece of code or an input from the attacker, which is capable of causing damage or danger for the user. In this scenario, the danger or damage could be caused using web application as a medium. Threat agent is a person who is capable of acting against an asset in a harmful manner. The asset referred to here is valuable user information. The threat agents could try to get an unauthorized access to these data and modify, delete, use or disclose sensitive data.

• MITIGATION

The Oxford dictionary defines ‘mitigation’ as “the action of reducing severity, seriousness or painfulness of something.” Removing or eradicating a threat completely is not possible, because, as a solution for existing attack techniques evolve, new techniques are being devised by the threat agents.
Studying the nature of attacks, a solution for reducing the severity of the attack can be implemented.

- **BROWSER**

  Oxford dictionary defines ‘browser’ as ‘a computer program with a graphical interface for displaying HTML files, used to navigate the World Wide Web (WWW). The browser is a window to the Internet; it acts as an interface between the client and the Internet.

- **PROXY SERVER**

  An intermediary between a client and a server is a proxy server. A client browser interacts with the web application server to request for web pages. The frequently requested pages are stored in an intermediate system, for faster retrieval, thereby enhancing the performance of the server. This intermediate system is called a proxy server.

- **VULNERABILITY**

  Vulnerability is a hidden flaw in any software, which could be exploited by a hacker to access a system. A threat agent makes use of these vulnerabilities in a web application to pose a threat to the user and the system as a whole.

- **BENIGN APPLICATIONS**

  The word ‘benign’ means having little or no detrimental effect or harmless. The web applications which do not harm the users or those with no loophole through which the underlying system will be compromised is a benign web application. Applications which implement just the actual intended functionality without probing into the sensitive information of the user are benign ones.

  Benign applications often have a complex structure and are mostly hosted in high speed servers. The speed depends on the business logic the
application performs. In smartphone scenario, applications, called apps, are downloaded from the server and installed in the local machine. This mobile application gets information from the user. This information is sent to the server for processing and the requested service is provided back to the user. There are different categories of mobile applications like games, books, utility, entertainment, booking, and such others.

- **MALICIOUS APPLICATIONS**

The word malicious means intending or indented to do harm. The web applications that have a code, which could cause harm to the user or the system are called malicious web applications. These applications allow an intruder to penetrate the system and retrieve unintended user data.

Malicious applications usually try to reproduce a genuine application and advertise its URL to potential users. When an application is found malicious, the server which hosts the application will be blacklisted. Hence, these applications change their servers often. They are mostly hosted on servers with poor performance [3]. The structure of the malicious application is much simpler than that of benign application. These applications are not bothered about the business logic, or providing any service.

An application has several vulnerable points which can compromise a benign application. A benign application’s code related information could be extracted by a hacker, and this information could be used to infect the application. There are a number of techniques to explore an application [2]. Some of them are:

- **Web spidering** - requesting for a web page, parsing it for links, and then requesting those links recursively till no further new content is found.

- **Discovering hidden content** - searching for those contents in an application that are not directly linked to the other pages (with broken links).
• Brute-force techniques - attempting to guess the name or to gain entry into a restricted application with many numbers of trials.

1.6 WEB APPLICATION VULNERABILITIES

Developing web applications, in general, does not need professional developer skills. Many tools for web applications are readily available in the market. Hence, an amateur coder can develop a web application with much ease. This has led to the increase in security threats when personal data are concerned. Internet security threat report [4] shows that the threats on web applications have been increasing tremendously over the past 5 years. Top five web application vulnerabilities [5] are:

• Structured Query Language Injection Attack (SQL injection)
• Cross site scripting attack (XSS attack)
• Broken authentication and session management
• Insecure direct object references
• Security misconfiguration

Earlier in the year 2009, the vulnerability research report [6] shows that the SQL injection attack tops the list of attacks followed by the cross-site scripting attack. Later, in the security report [4] of the year 2014, it could be seen that attacks on smartphone applications topped the list when compared with the desktop applications.

In the web application scenario, the web server receives input from the user through the Web browser and provides input and receives results from the database server. Failing to check the inputs from the user or input to the database will lead to the Input validation based attacks.
1.6.1 Input-Validation Based Attacks

Input validation attack occurs when the input provided by the user is not validated against the legitimate format. This can lead to retrieving of information from the application’s server in an unauthenticated manner. Input validation attacks are of two types, namely SQL injection attacks and Cross site scripting attacks.

- **SQL Injection Attacks**

Web applications work with a back-end database to store or retrieve data. Many Web applications store the data in the database and retrieve or update information as needed. These applications are highly vulnerable to many types of attacks, one of them being SQL injection Attacks (SQLIA).

Input data given by the user is dynamically embedded into the HTML code and sent to the server as a request. The server processes this request and provides the response back to the user. If this input data is malicious, then this input will in some way try to extract sensitive information from the database. This type of attack where the database is compromised using SQL statements is called the SQL injection attack (SQLIA).

An SQL injection attack occurs when an attacker causes the web application to generate SQL queries that are functionally different from what the user interface programmer intended. For example, consider an application dealing with author details. A typical SQL statement looks like:

\[
\text{select id, firstname, lastname from authors;}
\]

This statement will retrieve the 'id', 'forename' and 'surname' columns from the 'authors' table, returning all rows in the table. The result set could be restricted to a specific 'author' using 'where' clause.

\[
\text{select id, firstname, lastname from authors where firstname = 'kamala' and lastname='krithivasar'};
\]
An important point to note here is that the string literals 'Kamala' and 'Krithivasan' are delimited with single quotes. Here the literals are given by the user and hence they could be modified. They become the vulnerable part in the application. If the attacker wishes to drop the table called ‘authors’, a vulnerable literal can be injected into the statement as given below,

\[
\text{Firstname: Kam'};\text{ drop table authors--}
\]

\[
\text{lastname:}
\]

Now the statement becomes,

\[
\text{select id, firstname, lastname from authors where firstname = 'Kam' ; drop tableauthors-- and lastname = ' ';}
\]

and this is executed.

Since the first name ends with delimiter ' and -- is given at the end of the input, all other commands following the -- is neglected. The output of this command is the deletion of the table named ‘authors’, which is not the intended result from a server database.

The most common SQL Injection mechanism [7] are,

1. Injection through user input
   - The input provided by the user is modified to cause an attack

2. Injection through cookies
   - An attack could be embedded in a cookie and submitted to a web application that builds SQL queries from cookies.

3. Injection through server variables
   - If a server variable, which has information on browsing trends, user statistics, etc., is logged on to a database without validation, an attack could be injected.
4. Second order injection

- A malicious input is seeded into the database, which could be used to trigger an SQL injection attack in later stages.

- **Cross Site Scripting Attacks**

  Cross site scripting (XSS) attack is one among the top 10 web application threats. The vulnerability report Cenzic [4] shows that in the year 2014, XSS attacks tops the threat list with 25% occurrence. This type of attack injects a malicious script into a web site [8]. The website might actually be a benign site with vulnerable code.

  Cross cite scripting attack can occur when a user input is embedded into a webpage without validation or encoding. If the user submits a malicious script, the script gets embedded in the page. When an unsuspecting benign user views the page, the malicious script gets executed in his machine and brings the machine under the attacker’s control.

  Cross site scripting attacks can be mainly categorized as stored XSS and reflected XSS. Stored XSS or persistent XSS occurs when the malicious user input is stored in the target server. This is persistent in the server and is made available to the victim, when the page is requested. Reflected XSS or non-persistent attack is that which is reflected from a server in the form of response, error messages or search results. The attack payload is included in a parameter when the benign user follows the URL to the web site.

1.6.2 **Broken Authentication and Session Management**

When a user connects with the web application, the application authenticates the user and provides a session ID. This authentication credentials and the session ID could pose a threat, if they are not properly encrypted. After completion of a session if the user fails to logout, the credentials could be hijacked by an attacker. The attacker can then continue to use the session posing as the original user. This attack is made possible
using some of the vulnerabilities in the handling of the session id’s, namely they are exposed in the URL, they do not timeout and they are sent over unencrypted connections.

1.6.3 Insecure Direct Object References

Insecure direct object references occur when an application provides direct access based on user supplied input. Certain parameter values are created in such a way that the consecutive values can be easily guessed. When these parameters, along with their values appear in the Uniform Resource Locator (URL), an attacker can conveniently change the value and submit the URL. This will provide direct access to the object in the database, and modifications can be executed on the accessed object. The value of the parameter is used to directly retrieve a record from the database by submitting a request such as http://fbook.com/photos?uid=12345. Using the value of the user ID the photos of the user could be retrieved.

1.6.4 Insecure Configuration Management

The configuration of the server which hosts the web application plays a very important role in the security of the web application. The servers provide services, like data storage, mail, messaging and such others, for the web application. There are a variety of server configuration problems like security flaws that have not been rectified, improper file and directory permissions, use of default certificates and such others that can infect the security of the system.

1.7 SMARTPHONE EVOLUTION

The Oxford dictionary defines ‘Smartphone’ as “A mobile phone that performs many of the functions of a computer, typically having a touchscreen interface, Internet access, and an operating system capable of running downloaded apps”. Smartphone with its phenomenal growth has brought along security threat as a compliment. Smartphone have become their users’ identity. All the details of personal information like messages, credentials,
contacts, photos, videos and such others are stored in the phones. This makes security of mobile phone applications gain importance. Phones are used as a pocket camera, pocket video recorder, media players and such others. But the main driving forces behind the success of the smart phones are the applications that run on them. To keep the phone system alive, millions of applications are designed to provide various features such as web browsing, location mapping, gaming, photo editing and much more.

Similar to web applications, these applications are also being developed by the general public from around the world. They have no standards to be followed. Most of the available applications are developed by amateurs. Though each operating system has its own app stores, there are many other applications available with the third party vendors. Due to the above said reasons, the smart phone is vulnerable to major threats like Trojan horse, virus and worms. There are a number of malicious applications developed for Android platform. When these applications are downloaded, they cannot be identified as malicious ones, because they are designed and programmed to pretend as benign applications. But once they are initiated for the first time, these applications work in the background and they bundle up important data from the mobile phone and send to the remote server. A study shows that the number of new malicious attack samples in 2012 increased 8 times more than those available in 2011. Smartphone attack can be classified into 4 types as

1. Operating System attack, where the vulnerable points in the operating system are targeted.

2. Application Attack, where the vulnerable codes present in the mobile application are exploited.

3. Communication Network Attack, which exploits the Wi-Fi and bluetooth vulnerabilities

4. Malware Attack, in which the application is created to be malicious.
One of the most affected platforms is the Android operating system. Kaspersky Lab has reported to have found 99% of newly discovered mobile malicious program to target android platform.

As per the study on the usage share of operating systems, Mobile Operating System (OS) browsing statistics on Net Applications shows that android OS has the highest usage share among other operating systems. Hence the Android operating system is considered for study in this thesis. The open-source framework of the OS is also an additional facet.

Android Applications are secure, only until the permissions to access other applications are limited. Interprocess communication between two applications in the Android operating system happens through permissions. These permissions are presented to the user in two ways.

a. As a part of the agreement.

b. As a list of permissions.

An agreement is presented to the user when an app is downloaded from the play store. The permissions that a mobile application requests, are listed to the user. The user can either agree for all the permissions and install the mobile application or disagree, thereby not installing the mobile application. There is no in between choice of partially agreeing and giving selective permissions. In the user interface, not all the permissions are listed. Certain permissions are defined such that they need not have the approval of the user.

For example, an application that is used to switch on the flashlight in a smart phone, may have requested permission to access the Secure Digital (SD) card, to access the contacts and to turn on the Wi-Fi. This application may be malicious. Once this app is initiated, the application can turn on the Wi-Fi, access the SD card and send data to its server. The damage caused is sometimes very serious.
There are applications that place call or send message to premium numbers. To implement this function these applications use the permission “COST_MONEY” that can be used to deduct money from the user account without the knowledge of the user. The permission “START_ON_INSTALL” will let the application start and run in the background as soon as it is installed. An application downloaded from the Google play store may turn out to be malicious. Every application is tested by Google using its service called Bouncer [9]. When an application is given to Google play store, it is first analyzed by a Bouncer service, which checks for vulnerable points in the code and flow of data. However, only the core part of the application is available for download from Google play store. After the application gets downloaded to the mobile device, it will connect to its remote server to download the remaining data. There is a possibility of malicious code being downloaded from the server. There is also a possibility of getting infected while downloading patch codes or while updating the applications. These patch codes and updating codes are not checked by Bouncer, hence there is every chance of a threat.

Apart from the permissions in Android system, another vulnerable point in a smartphone is a WebView. A WebView is a web-based interface that is used to retrieve and display web content on a smartphone from web servers. The WebView is application specific, since it is exclusively designed for displaying a specific application. A WebView may itself be benign, but the application presented using the WebView may have a script that exploits a vulnerable code in the WebView, thus gaining control over it.

1.8 OBJECTIVES AND SCOPE

The scope of the research work is to provide simple, yet effective solution for various security issues in Web applications. This research work focuses on four aspects which are briefly given below.
• Algorithms for detection and mitigation of input validation based attacks:

The primary part of the research work focuses on detecting and mitigating the SQL injection attacks and cross site scripting attacks on Web Application. To materialize this, the challenges and the issues concerning the attacks were reviewed and studied. A strategy was proposed to identify the vulnerable points in the code and to detect and mitigate an attack. The experiment results showed significant improvement.

• Algorithm to detect an abnormality in the System:

A system under attack has to be recognized and the mobile application causing the attack has to be identified and reported. Initially the behavior of the system under a normal condition was studied and recorded. The change in behavior when malicious activity occurs was identified and reported to the user. Amendment capture service algorithm was framed for identifying malicious activities in the system.

• Algorithm to detect and mitigate permission based vulnerabilities:

The access to a mobile application by a malware could be possible only through permissions. To avoid unrelated communication access, essential permissions for specific categories were segregated and the defaulters were monitored. Permission enforcer algorithm was designed and implemented to detect such malicious applications.

• Algorithm to detect and mitigate WebView based vulnerabilities:

WebView's overlying mobile application could alter its functionality, thus exposing the system to vulnerabilities. The WebView code is statically analyzed before installation or distribution by a third party market. WebView vulnerability detection rules were constructed and applied on these WebViews to expose the vulnerable holes.
1.9 ORGANIZATION OF THESIS

The contributions made in this dissertation are presented in eight chapters as follows:

Chapter 1 provides an introduction on web applications, protocols used in web applications, mobile applications in smartphone and their security issues.

Chapter 2 presents a detailed literature survey on the vulnerabilities in the web application and techniques that have been employed to curb them. Different solutions provided by authors have been discussed and compared.

Chapter 3 discusses the overall contribution of this dissertation. An overview of techniques employed to mitigate vulnerability is explained.

Chapter 4 presents the architecture, algorithm for detection and mitigation of input validation based attacks. The evaluation and analysis of the results are also presented.

Chapter 5 explains the architecture used to implement the algorithm for detection of abnormality in the smartphone system. System evaluations and results are also discussed.

Chapter 6 elucidates the system architecture and algorithm for detecting permission based vulnerabilities in Smartphones. A detailed performance analysis using case study is given.

Chapter 7 presents the architecture used to detect WebView based vulnerability in Smartphones besides the evaluation and result analysis.

Chapter 8 concludes the dissertation and provides scope for further work.