ANALYZING AND MITIGATION CRITICAL VULNERABILITIES OF WEB APPLICATIONS

THE SUPERIOR UNIVERSITY LAHORE

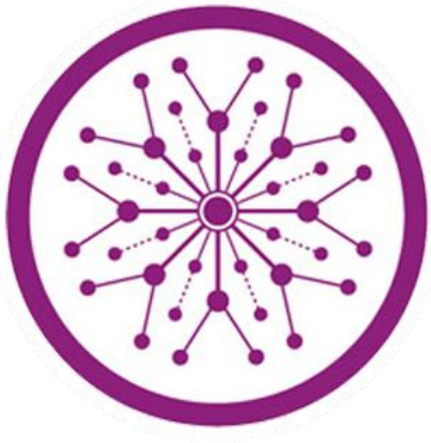
MS INFORMATION TECHNOLOGY

HASEEB AKBAR
MSIT-S19-004
Session: 2019 – 2021

DEPARTMENT OF COMPUTER SCIENCE &

INFORMATION TECHNOLOGY THE SUPERIOR UNIVERSITY

LAHORE, PAKISTAN



THE SUPERIOR UNIVERSITY LAHORE

ANALYZING AND MITIGATION CRITICAL VULNERABILITIES OF WEB APPLICATIONS

A thesis submitted in partial fulfillment of the requirements for the
Degree of Master of Science in
Information Technology

Student Name: HASEEB AKBAR

Supervisor Name: DR IMRAN KHAN

DECLARATION

This thesis is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions. I also declare that this work is the result of my own investigations, except were identified by references and free from plagiarism of the work of others.

Signature: …….

Haseeb Akbar

Date: …….

APPROVAL

The undersigned hereby certify that they have read and recommend the thesis entitled "Analyzing and Mitigation Critical Vulnerabilities of Web Applications" by "Haseeb Akbar" for the degree of Master of Science in Information Technology.

_____

Dr Imran Khan, Thesis Supervisor

_____

Thesis Committee Member

_____

Thesis Committee Member

_____

Head of Department

DEDICATION

I would like to dedicate this thesis research to my beloved family, teachers, and friends.

ABSTRACT

Use of Web Applications increase day by day, billions of users use web applications for different purposes. Many businesses and consumers are conducting businesses over the internet due to vast availability of facilities. As we know a successful organization is that who provide secure environment to their customers as well as employees, an organization is responsible for the security of their customers. Internet has become the fastest growing technology used in these days, but it also generates the new security challenges to web applications in internet due to the internet easily availability and freely available different types of web services. Security testing becomes an essential and crucial part of the web application development due to the growing complexity of web systems. Security testing helps to protect data confidentiality, prevent information leakage, and ensure that the system functions as intended. When web applications are exposed to malicious input data, it tests if they meet the security requirements. Due to the and number of security vulnerabilities, it is essential to comprehend their specific problems and concerns, which will ultimately serve as valuable feedback for security testing tool developers and test managers working on their respective projects. A vulnerability in web application can cause major to lose of data, and privacy issues. A simple vulnerability can allow the unauthorized access and perform conflict actions. Hence, the security of web applications is also as important as the other aspects on the internet. Web applications contain vulnerabilities, which may lead towards serious security issues such as authentication, confidentiality etc. Building a Secure and safe website is very important, costly, and challenging task for web designers. In this research, I literally discussed the most commonly faced security flaws/vulnerabilities in web apps: These are SQL Injection, XSS, Cross-Site Request Forgery, Broken Authentication and Session Management, Insecure Direct Object References, Security Misconfiguration, Insecure Cryptographic Storage, Failure to restrict URL Access, etc. After literally discussing I discussed how to prevent from most common security vulnerabilities in web applications. This research enables the developers for choosing the best technique to mitigate or prevent against the security vulnerabilities in Web applications.

Keywords: Web Application Vulnerabilities, Mitigation, Web Security, OWSAP

Table of Contents

INTRODUCTION AND BACKGROUND

This research presents the previous work of the Web Applications & their use in terms of scientific research. Applications Program means set of instructions given to the computer to solve some particular or specific problems. These instructions normally known as programs. All the applications are not programs, program with the user interface called application. There are many applications working today but according to usage following are the categories of applications:

1. Applications used in Desktop Computers

2. Web-based Applications

3. Applications used in Mobile devices

Software applications began with Desktop Applications. A desktop application defined as software that can installed on laptop or computer and used to perform some specific tasks. Many desktop applications versions also working on mobile phones. There are different desktop applications for data analysis & diagrams like MS Excel, for communication MS Outlook and for presentations and graphics MS PowerPoint etc. are used. After extreme success of Desktop apps, use of Web applications increases day by day. Today, many developers working on desktop applications, many desktop applications developed according to customers' requirements. Secondly, mobile applications commonly known as an app, that specially created to run on a mobile phone, such as Smart phones, Tablet computer and Handheld devices etc. Web applications are used almost in every organizations. However, with the advancement of internet and E-commerce, web application development gained importance. This research focused on web applications and its importance with vulnerabilities and mitigations in Web Applications. As billions of users uses Web applications these days. Following are the categories of web applications:

1. Informational (Newsletters, E-books, Online Newspapers)
2. Online Communities (Chat groups, auctions)
3. Web Portals (E-Shopping malls)
4. Web services (Enterprise applications)
5. It is collaborative (User-Provided Information, Customized Access)
6. Business transaction (Electronic Shopping, banking etc)
7. The Process (Status monitoring, Planning, and scheduling systems)
8. Workplaces with a wide range of positions (Collaborative design tools)

In comparison to software development, web application development has a very short history. But, within very short period, many web-based systems and software have been created and are being used frequently. Because of its cross-browser availability, developers have selected the web as the preferred platform for developing and deploying applications. Several changes have been made to web applications. Permission is given without charge to make digital or hardcopy notes of complete or partially of this work for personal use or classroom use. With the advent of technology, attacks on web apps are also increasing at a very high rate. As a

result, the conventional software development life cycle approach may not be appropriate when it comes to web application protection. It is difficult to write stable code and/or evaluate it to implementation is attack-proof. The use of context-specific security design patterns in web applications will help to reduce the risk of an attack. Attempts have been made to describe some security patterns that developers may use.

What is Web App Security?

A procedure of authenticate, validation and checking the efficiency of application security controls to assess the security and protection of a system or network is known as security. A web application security evaluation is totally focused on the security of a web apps. The method includes analyzing the competition and innovation for any bugs, technical defects, or weaknesses. Any security flaws found will be communicated to the application or device's owner, along with a mitigation plan, risk assessment or technical solution.

Vulnerability Means?

A vulnerability is a mistake or deficiency in the design, implementation, function, or management of a system that could be exploited to compromise the system's security objectives.

Threat Means?

Threat is something that exploits vulnerabilities to harm an application's assets (a malicious external user, an internal user, infrastructure instability). (resources of value etc).

What is Test?

Test is a procedure for demonstrating that application satisfies the security needs of an organization or user.

The OWASP technique is collaborative and open:

Open: Any security expert may contribute to the project by sharing his or her knowledge. All is there for free.

Collaborative: Before writing the posts, the team holds a brainstorming session to exchange ideas and create a shared vision for the project. This entails a broad consensus, a larger audience, and more engagement.

This method aims to produce a well-defined Testing Methodology that is:

• Reliable

• Reproducible

• Strong

• Consistency Controlled

The issues that need to be resolved have been well identified and reviewed. It's important to follow a system for testing all known vulnerabilities and documenting all security testing activities.

Software development can be thought of as a mixture of individuals, method, and technology. If these are the factors that "make" apps, it stands to reason that they should also be evaluated. Many people nowadays test the devices or applications before purchase.

Components of a successful research programmed should include:

People: This component used to ensure that proper education and information are given.

Process: This component used to ensure that required policies and guidelines are in place, and that citizens are aware of how to implement them.

Technology: This component used to ensure that the performed processes has been successfully implemented.

Systematic Technique is used, testing only the technological execution of an app that will not reveal managerial or processing weaknesses that may exist. Through checking staff, procedures, and systems, eradicating glitches as soon as possible, and finding the major causes of errors, a company may detect problems that will eventually appear as technology defects. Similarly, reviewing just a subset of a system's technological issues would result in an insufficient and ineffective security posture measurement. “If cars were constructed like applications, security measures would only consider sideways impact.” Cars would not be checked for roll stability, emergency maneuverability, brake effectiveness, side effects, or theft resistance,” said Denis Verdon, Head of Information Security.

PROBLEM STATEMENT

Vulnerabilities and Their Causes

- **Mistakes in design and development:** Both hardware and software may have flaws. These vulnerabilities have the potential to expose your company's sensitive data.
- **lack of appropriate device design is another source of weakness:** If the system is improperly designed, it may introduce flaws/vulnerabilities that allow attackers to get access to the system & steal data.
- **Human errors:** Security breaches can be caused by human factors such as inappropriate document disposal, leaving documents unattended, coding errors, insider attacks, exchanging passwords over phishing sites, and so on.
- **Connectivity:** If a device is connected to an unsecured network (open connections), hackers can access it.
- **Complexity:** As a system's complexity increases, so does its security vulnerability. The greater the number of features a machine has, the more likely it is to be targeted.
- **Passwords:** To avoid unauthorized entry, passwords are used. They should be complex enough that no one will guess them. Passwords should never be exchanged with others and should be updated on a regular basis. Under these instructions, people reveal their passwords by writing them down and use passwords that are very easy to guess.
- **User Input:** You have probably heard about SQL injection, buffer overflows, and other security issues. These techniques may be used to target the receiving device with data collected electronically.
- **Management:** Managing security is difficult and costly. As a result of a lack of appropriate risk control in certain companies, insecurity is introduced into the system.
- **Lack of training to staff:** Due to human errors and other security flaws, the security of the system becomes at risk.
- **Communication:** Communication channels/path such as the internet, mobile networks and telephones increase the risk of security theft.

TOP 10 OWSAP

SQL INJECTION:

SQL injections can be used by many hackers and try to access the database. This occurs when an attacker injects malicious SQL statements into a database through various fields and other injection points to collect information from the database and gain control over it. They may use this information to get access to, change, or even remove data, as well as destroy the entire system.

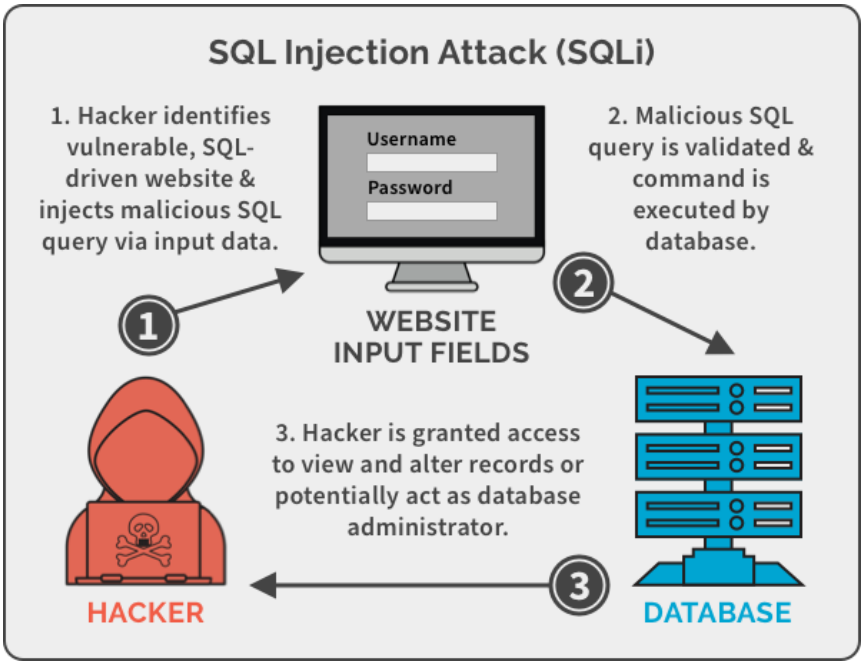


Figure. 1 SQL Injection Attack

CROSS SITE SCRIPTING:

Another common web application deficiency that can compromise the clients' protection is known as Cross-Site Scripting, or XSS. These attacks are responsible for introducing harmful code into an app that already working and executes it on the part of client.

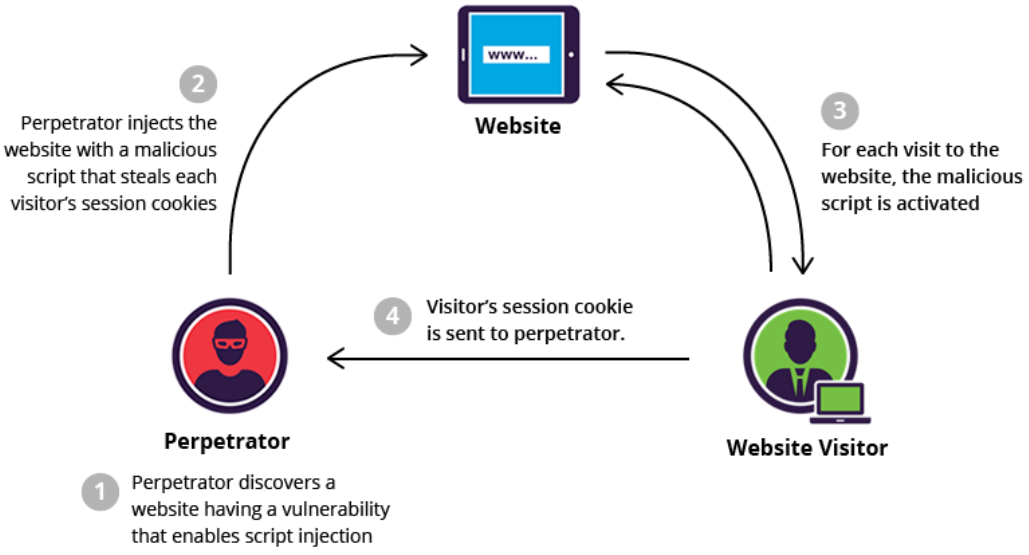


Figure. 2 Cross Site Scripting

BROKEN AUTHENTICATION AND SESSION MANAGEMENT:

Authentication-related web application vulnerabilities arise when appropriate user authentication controls are not implemented properly. Users' accounts are at risk of being

hacked as a result of this. Attackers may take advantage of these web security flaws to take control of any user account or even the entire system.

Credential Stuffing is one of these vulnerabilities, in which an attacker tests a list of valid passwords and usernames obtained from another hack or attack before they find a valid combination and gain access.

INSECURE DIRECT OBJECT REFERENCES:

It occurs when keys of database or files are accessible to the user. Since internal artefacts are exposed, attackers may use a variety of attacks to gain access to them and obtain data or access to critical databases. Authentication is often absent.

AFFECTED OBJECTS:

Database objects are normally vulnerable due to Uniform Resource Locator parameters that expose serialized data keys that a hacker may use to gain access to data. An intruder can also modify and alter static files to gain access to sensitive information or other users' data.

CROSS SITE REQUEST FORGERY:

Server-side input validation can be used to prevent access to confidential files and databases. Malicious users can manipulate URL parameters and file names if input is tested server-side. Furthermore, access control mechanisms may assist in determining whether a user has permission to access or modify files and databases.

SECURITY MISCONFIGURATION:

Web-based applications are often misconfigured, exposing plenty of vulnerabilities for attackers to exploit. Unpatched bugs, inactive pages, unsecured files or folders, obsolete applications, and executing software in debug mode are examples of security misconfigured vulnerabilities.

Affected objects:

Security misconfigurations can influence and attack all aspects of web applications. It is extremely important to run a security audit to check for attacker’s attacks or breaks whenever there is a misconfiguration.

INSECURE CRYPTOGRAPHIC

It is a common problem that arises when confidential data is not saved safely.

On a website, confidential data information includes user passwords, user profile details, user’s health details, user’s credit card information, and so on.

The database of the application will have all the information of the use which is at risk and is exposed to attackers if not stored properly or stored without encryption or hashing*.

FAILURE TO RESTRICT URL ACCESS:

Web apps search URL and access right by making these links and buttons secure. The applications must also perform similar access control tests whenever these pages are accessed.

Privilege pages, places, and services are not shown to privileged users of most applications.

By making an educated guess, an attacker can acquire access to privilege pages and all the sensitive information that they contain and by gaining access to the information, the intruder can view and invoke changes in the confidential information available on sensitive websites and also replace the functions.

MISSING FUNCTION LEVEL ACCESS CONTROL

Vulnerabilities may occur when server-side authorization is misconfigured, disabled, or missing, leaving your back end vulnerable to attacks.

Affected objects:

Front-end UIs designed with elements to allow administrators access to data or other critical app elements are often the target of these attacks. Most users won't be able to see the administrator feature in this situation, but those searching for flaws will be able to discover and exploit this bug with malicious requests.

USING COMPONENTS WITH KNOWN VULNERABILITIES

When considering using third-party code or components in your web application, do your homework. Using unrestricted code from unknown sources can lead to a slew of security issues.

AFFECTED OBJECTS AND HOW TO FIND THEM

The National Vulnerability Database has a comprehensive description of the various third-party vulnerabilities to assist in evaluating which components are vulnerable.

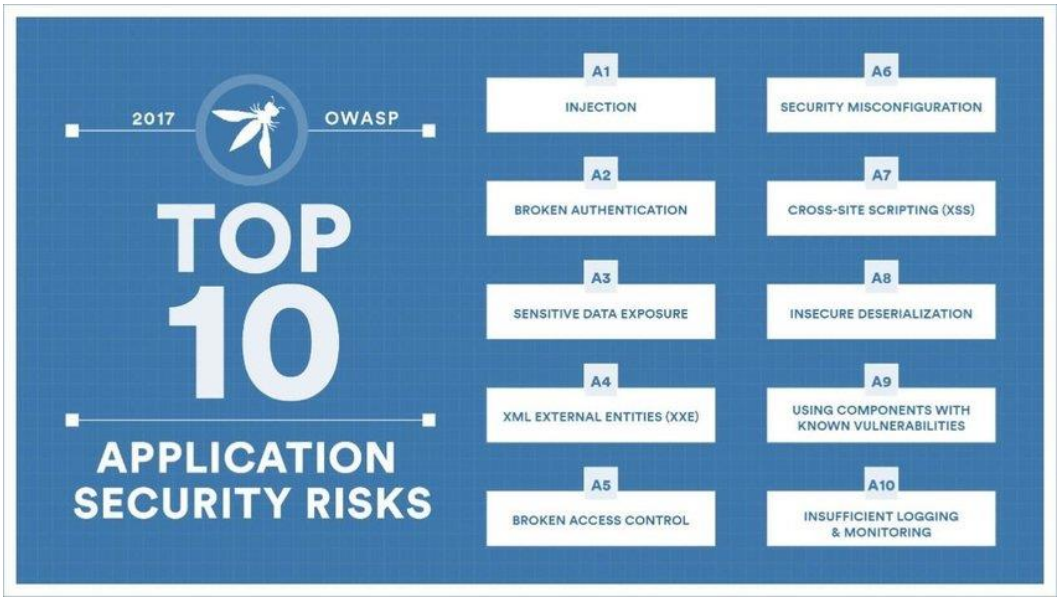


Figure. 3 Top 10 OWASP

Vulnerabilities in third-party code will affect any aspect of your app. Backdoors, for example, may be applied to financial services code, giving attackers access to confidential information.

OWASP Top 10 - 2013		OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

Figure. 4 Notable Changes the OWASP Top 10 from 2013 to 2017

Breakdown of Changes Between 2013 and 2017 Part 1

A1:

- Unchanged, Injection remains at the top spot.

A2:

- Broken Authentication and Session Management remains in the second spot.
- Name has been shorted to simply “Broken Authentication”.

A3:

- Previously occupied by Cross-Site Scripting (XSS)
- Now Sensitive Data Exposure

A4:

- The previous A4 category was Insecure Direct Object References
- A new category, XML External Entities (XEE) was placed here in 2017.

Breakdown of Changes Between 2013 and 2017 Part 2

A5:

- In 2013, A5 went to Security Misconfiguration.

- 2013's A4 (Insecure Direct Object Reference) and A7 (Missing Function Level Access Control) categories have combined to become Broken Access Control

A6:

- 2013's A5, Security Misconfiguration has been placed at A6 in 2017

A7:

- Previously, this spot went to Missing Function Access Control Level
- Now occupied by XSS (Cross-Site Scripting)

Breakdown of Changes Between 2013 and 2017 Part 3

A8:

- The now removed category, Cross-Site Request Forgery, sat here in 2013.
- Insecure Deserialization (a new category) has been placed at A8.

A9:

- Unchanged, remains as Using different Components with Known weaknesses/Vulnerabilities

A10:

- 2013's A10, Not validated Redirects and Forwards, was removed from the Top 10
- Now occupied by the new category, Insufficient Logging and Monitoring

There are many methods are used to analyzing the vulnerabilities following methods I cover in my research:

1. Manually Inspections & Reviews
2. Threat Modeling
3. Reviewing Code
4. Testing using Penetration Technique.
5. Using testing framework of OWASP

1. Manual Inceptions & Reviews

Manual inspections can be defined as human observations of users, procedures, and machines while for technology decisions inspection, such as architectural designs, can be review manually. Reviewing paper documents or getting interviews with the creators or higher level management of the device are the components of manual inspections. Manual inspections and human evaluations, though basic in nature, can be among the most efficient and reliable methods available. The tester will easily decide whether any security issues are likely to be apparent by asking others how something operates and why it was introduced in a particular way. Manual checks and assessments are one of the methods for testing the SDLC (software development life-cycle) process and ensuring that appropriate policies and expertise sets are in place.

When performing manual inspections and evaluations, it must be ensuring that a trusted but verify model be used, as it is for many other items in life. Not everything appears in front of the tester or told to the tester will be correct. Manual reviews are extremely important and helpful for ensuring that people knows the security process, are known of policy, and have the important skills to develop or execute a safe app.

Following are the advantages of Manual Inceptions & Reviews:

• Doesn't need any additional technology

• Can be used in several circumstances

• Early in the SDLC

• Flexible

• Promotes teamwork

Following are the Disadvantages of Manual Inceptions & Reviews:

• Requires extensive human thinking and ability to be successful

• Time consuming

• Helping material always not accessible

2. Threat Modelling

Threat modelling one of the most common tool for system designers to check the security and their threats that their computers and apps can face. This model can be viewed as a tool which can be helpful as an application risk assessment. After analyzing we get the result, threat modelling can be viewed as an application risk assessment. It allows the designer to devise mitigating plans for possible flaws, allowing them to concentrate their extremely limited materials and effort on the parts of the system that need it the most. It is important that all apps create and document a threat model. Threat models should be introduced as soon as possible in the System Development Life Cycle, and they should be reconsidered as the app develops.

To create this model, we suggest using a straightforward approach based on the NIST 800-30 [11] risk assessment standard.

• Decomposing the app

By using manual inspection method to figure out how the app functions, as well as its properties, features, and connectivity.

• Classifying & Defining assets

Divide Data into critical and Non-critical assets and order them in sequence of value to the company.

• **Examining possible weaknesses**

Whether technological, organizational, or management related.

• **Investigating possible threats**

Using threat simulations or attack trees, provide a reasonable view of harmful attack vectors from the attacker's points of view.

• **Developing mitigation strategies**

For each of the risks considered reasonable, create mitigating controls.

This model's performance can be large, but it is usually a series of lists and figures. According to the OWASP Reviewing Code Guides an web App Threat Modeling approach that is be used to test software for possible security bugs in the design. There is no limited approach that used to developing hazard models and performing application knowledge risk assessments. (12).

Following are the advantages of Threat Modelling:

• Practically check the attacker's perspective of the device

• Flexibility

• Used in the Early years of the SDLC

Following are the Limitations **of Threat Modelling:**

• This is a relatively modern technique.

• Good threat models do not always indicate good software.

3. Reviewing Code

The process in which manually reviewing the code of a web app for security problems is called source code analysis. There are many significant problems that may leads towards major security problems are most of them are difficult to detectable using different method of inspection or checking. "If you want to know what's really going on, go straight to the source,". Almost every security experts believe that looking at the code is the only way to learn about it. All the details needed to find security flaws is somewhere in the code. When testing web applications (especially if they were built in-house), unlike testing the source code by third-party closed software such as OS, the source code should be made available for testing.

Other types of analysis or monitoring, such as penetration testing, are highly difficult to detect many unintended yet important security issues, making source code analysis the technique of choice for technological testing. A tester can reliably assess what is happening (or supposed to be happening) with source code, eliminating the guesswork of black box testing.

Concurrency issues, flawed business logic, access control issues, and cryptographic vulnerabilities, as well as backdoors, time bombs, and other types of malicious code, are

examples of issues that are particularly conducive to being discovered through source code reviews. These problems often manifest as the most dangerous flaws in websites. Source code review may also be very useful for identifying implementation problems, such as areas where input validation was not done or where fail open control procedures were used. However, remember that operating procedures must be checked as well, as the source code deployed can differ from the one studied here [13].

Following are the advantages of Source Code Review:

• Comprehensiveness and effectiveness

• Fast

Contrary to popular belief, there are several disadvantages to using this method.

• Needs security developers with a high level of expertise.

• In collected repositories, it is possible to skip problems.

• Run-time errors are difficult to detect.

4. Penetration Testing

For several years, penetration testing has been a popular method of assessing network security. Black box research or ethical hacking are other terms for it. This testing is the "method" of remotely testing an app who running and doing his process check for security vulnerabilities without understanding the application's internal workings. Typically, the penetration testing team will have user-level access to an application. The tester takes on the role of an intruder, looking for and manipulating weaknesses. A valid account on the system will be given to the tester in many cases.

Though penetration testing has been shown to be useful in network protection, it is not naturally applicable to applications. When it comes to network and operating system penetration testing, the bulk of the job entails locating and exploiting identified weaknesses in particular technologies. Penetration testing for the web apps arena is like a pure analysis since web apps are almost entirely wearable. This testing has been created to automate the process, but their efficacy is generally low due to the design of web applications.

Web apps penetration testing is now widely used as a key security testing method for many people. Although we believe it has a place in a testing program, we do not believe it should be the main or only testing method. When Gary McGraw said in [14], "If you fail a penetration test, you know you have a really bad problem indeed," he summed up penetration testing perfectly. If you pass a penetration test, you have no way of knowing whether or not you have a big threat.”

Principles of Testing

1. No Silver Spoon

Although it is believed that a scanner for security or device firewall can have a variety of defenses against attack or detect a wide range of issues, there is no magic bullet for vulnerable apps. Application security evaluation software is typically inexperienced and unsuccessful at deeply tests or giving the sufficient test coverage, though useful as a initial pass to finding the low-hanging fruit. Keep in mind that security is a method, not a product.

2. Think Strategically, Not Tactically

Most of the Security experts have recognized the fallacy of the patch-and-penetrate model, which was common in information security during the 1990s, over the last few years. The patch-and-penetrate strategy describes addressing a problem that has been found, but without conducting a thorough investigation into the root cause. This model is often correlated with the vulnerability window depicted in the diagram below. The ineffectiveness of this paradigm has been shown by the evolution of vulnerabilities in widely used applications around the world. Please see [6] for more detail on the vulnerability window.

The patch-and-penetrate model makes some assumptions that are wrong. Many users assume that updates disrupt normal operations and can cause existing applications to malfunction. It's also a mistake to presume that all users are aware of new updates that have been issued. As a result, every user of a product cannot submit patches, either because they believe patching will interfere with the software's functionality or because they are unaware of the patch's existence. To avoid issues that is related to the security issues in a web application, it is critical to the security that is incorporate security into the development of software method name called Software Development Life Cycle (SDLC). By creating specifications, rules, and guidance that match and function within the methodology of development, developers can incorporate protection into the SDLC. To help allocate sufficient resources to the different areas of a system that are high at risk, threat modelling and other techniques can be used.

3. SDLC

The SDLC is an advanced method among developers. Through incorporating security into every step of the SDLC, organizations may take a comprehensive approach to application security that takes advantage of existing procedures. As the names of the different steps/phases which vary depending on the SDLC model used by a company, each conceptual step of the archetype SDLC will be used to create the application. To ensure a cost-effective and robust security programmed, each step has security requirements that should be integrated into the current process. There are several stable SDLC frameworks available that provide both descriptive and prescriptive guidance. The sophistication of the SDLC process determines whether an individual follows descriptive or prescriptive advice. Prescriptive advice explains how the safe SDLC should operate, while descriptive advice explains how it is used in practice. Both have their own merits. If you are unsure where to begin, a prescriptive approach will include a list of possible security controls that can be implemented in the SDLC.

4. Test Early Test often

When an error is discovered at the staring phase in the SDLC, it can be fixed very quickly and at a very lower cost. In this respect, a security error is no different than a practical or performance-based bug. Educating the development and Quality Assurance teams about most common security problems and how to spot and avoid them is a critical step in making this possible. New libraries, tools, and languages can contribute to the development of better programmed (with less security bugs), but new threats emerge on a regular basis, and developers must be aware of the threats that impact the applications they are creating. Security testing education often aids developers in developing the necessary mentality to assess an application from the viewpoint of an intruder.

5. Understanding the Security limitation

It is necessary to understand that how much protection an app would necessitate. Different Information and properties that must be covered should be classified in a way that specifies how they will be treated. Legal counsel should be consulted to ensure that all relevant security standards are met. Federal legislation, such as the Gramm-Leach-Bliley Act [8], or state laws, such as California SB-1386 [9], can impose requirements in the United States. Both country-specific legislation and EU Directives can apply to organizations based in EU countries.

6. Development of the right mindset

For evaluating an application against the security flaws, you must think "outside the box." Normal use cases can evaluate the application's normal behavior when it is used in the intended way by a user. Going above and beyond what is anticipated in security testing necessitates thinking like an attacker attempting to crack the application. It can also assist in determining which assumptions made by web developers are not always correct and how these assumptions can be challenged.

7. Knowing the subject

The need for correct application documentation should be one of the first big initiatives in any successful security programmed. The structure, data-flow diagrams, use cases, and other details should all be recorded and made available for analysis. The technical specification and implementation documentation should contain information that not only mentions the desired use cases, but also any use cases that are expressly prohibited.

8. Right tools Right Place

Although there is no such thing as a silver bullet tool, tools do play an important role in the overall protection programmed. Many routine security activities can be automated using a variety of open source and commercial software. By assisting security personnel in their duties, these tools will simplify and speed up the security process. However, it is important to understand precisely what these methods can and cannot do in order to avoid overselling or improper use.

9. Develop Metrics

The ability to decide whether things are getting better is an integral aspect of a successful protection programmed. It's critical to keep track of the outcomes of testing engagements and create indicators that expose the organization's application security patterns.

Consistent metrics that can be created automatically from publicly accessible source code will also aid the institution in evaluating the efficacy of security bug prevention mechanisms. Since metrics are difficult to create, standard metrics such as those given by the OWASP Metrics project and other organizations are a good place to start.

It is very necessary to create a tool for monitoring that what testing activities were using, by whom, when they were done, and specifics of the test results at the end of the testing phase. It's a good idea to settle on a report format that would be beneficial to all parties involved, including developers, project managers, company owners, IT departments, audits, and enforcement.

The result should be transparent to the owner of the business in terms of defining risks of material and obtaining their approval for subsequent mitigation measures. The result should also be transparent to the application developer in terms of identifying the specific feature that is affected by the flaws/weaknesses and providing suggestions for addressing these issues in a language that the developer understands.

Another security tester should be able to replicate the findings using the analysis. Security testers aren't known for their creative writing abilities, so deciding on a complicated report may lead to test results that aren't properly recorded. Using a security test report prototype will save time and ensure that reports are reported correctly and reliably, as well as in an audience-appropriate format. Many businesses have begun to employ automated web application scanners. Although they certainly have a place in a testing programmed, some fundamental issues about why automated black box testing is not (or will never be) successful must be addressed. However, bringing these problems to attention does not prevent companies from using web application scanners. Instead, the aim is to ensure that the constraints are acknowledged and that research frameworks are properly designed. To run a good testing programmed, one must first figure out what the testing goals are. These goals are determined by the security criteria. This portion explains how to document the detail security testing specifications using relevant guidelines and legislation, as well as positive and negative implementation requirements.

OBJECTIVES

This thesis has the following nine objectives:

1) To help ensure safe web browsing by introducing diverse machine learning approaches.
2) To secure information available on web databases
3) This study will also help in providing confidentiality, authentication, and authorization of websites.

4) Our aim is to help in designing and realize a perfect technique to resolve the security problems in web applications.
5) To achieve the correct prediction about security in web applications.
6) To achieve the high-rate accuracy in web applications by using efficient and secure techniques.
7) To get a successful feedback that has been acquired for secured web applications.
8) To create a best performance on these features in learning model and evaluation measures.
9) Lastly, we try our best to get maximum result in minimum budget.

Significance of the study

This significance of study is to introducing security testing in web-based application quality check phase. The aim of the study is too aware about the vulnerabilities and security problems in web applications and aware about how it destroys the personal security and gets confidential information through different attacks. The main goal of the study is to achieve early detection of vulnerability in initial stage and prevent vulnerability to destroy the public confidential information. Daily, approximately 30 000 - 50 000 websites are hacked, according to latest figures. The numbers are increasing day by the day, and the numbers of website security is increasing at a breakneck pace. Security is becoming particularly crucial in the online world and protecting the website and the data it holds is crucial right now. There are more than 1,5 billion websites on the internet today, and people use search engines to find information on those pages. As a result, search engine optimization is more critical than ever, and every webmaster must comprehend both the true meaning of SEO and the opportunity it can provide for any company. Customers are warned and prevented from accessing the website by Google and other search engines (for which you don't want to be on the naughty list). Google, for example, has finally stepped up its game. Starting in July 2018, any website that does not use SSL (HTTPS) will be marked as vulnerable, resulting in an SEO penalty that will make it more difficult for your business to reach new customers. Google also revealed new information about its spam-fighting activities, showing that over 80% of compromised pages have been identified and excluded from search results. (Image courtesy of Search Engine Journal). However, the fact is that a compromised website causes a customer to lose confidence, which leads to a loss of company credibility, which in the case of e-commerce can also mean the end of the business.

Viruses are also on the rise when it comes to website protection and CMS security. WordPress, for example, remains the most contaminated website CMS. Every day, approximately 50 000 websites are hacked, and the bulk of these 50,000 pages are legitimate small businesses unintentionally spreading malicious code for cyber criminals. When, your website is compromised and attached to various blacklists, potential customers are unable to access the goods or services you provide. In any case, if a potential customer visits your site and is alerted or poisoned, the chances of the customer returning are extremely slim. According to Acunetix's latest study, approximately 84 percent of websites have vulnerabilities, which means that all of them may be compromised at any time. The method of removing malware from a website is all about understanding the vulnerabilities and understanding how a hacker think. This is why we still suggest manual clean-up service providers. Malware is always hidden from the original

files and databases, and attackers go to great lengths to ensure that you won't be able to quickly delete their backdoors. It is, without a doubt, costly. Not only the malware removal service, but even the lost sales and reputational harm will take a long time and cost a lot of money to repair. If your site isn't safe, hackers will use it to infect visitors with malware and steal the information it contains. In, this study we are using different techniques that how to prevent hacker to gain the access of someone's personal data. We discussed most used hacking tools and techniques and how to prevent against those hacking tools and techniques. The truth is that a compromised website causes a customer to lose confidence, which leads to a loss of company credibility, which in the case of e-commerce may also mean the end of the business.

Research question and Hypothesis

Following question discussed in this Research.

1. Have vulnerabilities been reduced to a point that they can be released?
2. How does this product's protection compare to that of related software products?
3. Is it true that all security test standards have been met?
4. What are the primary causes of security problems?
5. How many security vulnerabilities are there in comparison to security bugs?
6. Which security operation is the most efficient at detecting flaws?
7. Which team is more efficient at resolving security flaws and vulnerabilities?
8. What proportion of all vulnerabilities are considered high-risk?
9. What mechanisms are the most powerful at identifying vulnerabilities?
10. Can types of security tests (e.g., white box testing vs. black box testing) are most successful in detecting vulnerabilities?
11. How many vulnerabilities are discovered during reliable code reviews?
12. During safe design tests, how many vulnerabilities are discovered?

It is necessary to know deeply understanding the testing methods also the testing methods to make an informed decision based on the testing results. To determine which protection tools to use, a tool taxonomy should be used. Security devices may be classified as effective at detecting commonly known security flaws in a variety of objects.

The problem is that unknown security vulnerabilities are not checked. The absence of flaws in a security test does not imply that the program or application is secure. According to some studies [22], tools can only detect 45 percent of total vulnerabilities at most. And the most advanced automation software cannot compete with a professional security tester. Depends solely on good test results from automatic detection tools can lead to a wrong sense of security/vulnerability. The higher the performance of the security evaluation and review, the more familiar security testers are with security testing methodology and testing procedures, the better. It is critical that managers who invest in security testing software often consider recruiting professional human resources and investing in security testing training.

The following details should be used when disclosing security test results:

1. The classification of each weakness according to its type.

2. The risk of the problem solving a security threat.
3. The source of security problems (e.g., security errors, security vulnerabilities)
4. The method for identifying the problem that was used to locate it.
5. The vulnerability's remediation
6. The Density of the vulnerability

It would be known to understand whether and why the detection and control of security flaws is unsuccessful in mitigating the security threat by explaining what the security threat is. Reporting the problem's root cause will facilitate in determining what needs to be done. In the situation white box inspection, the violating source code would be the software protection root cause of the vulnerability.

It is also critical to give feedback to the software developer that helps the tester how to recheck and detect the vulnerability after problems have been identified. To determine whether the code is vulnerable, a white box testing technique (e.g., security code analysis with a static code analyser) can be used. If a vulnerability is discovered using a black box testing technique, the test report must be give the instructions on what is the procedure to verify the vulnerability's disclosure to the front end.

Security test data were examined by software developers to demonstrate that application is designed more safely and effectively. Project managers are looking for information that can help them handle and use security testing activities and tools in accordance with the project plan. Security test results will inform project managers that projects are on budget, on track for completion dates, and improving during tests.

LITERATURE REVIEW

Web based applications are increasing day by day, with every little upgrade in web systems, data security is becoming the biggest challenge [1] web complexity is increasing very fast, data security concerns are discussed in every forum. As web-based systems receive requests from user in form of hypertext transfer protocol request, receiving all such requests become difficult to scan when in a large-scale web environment, hence the initial phase of scanning is to be done before the deployment of project to test and verify its safe deployment as well as security [1].

To access the security of web application, the need to conquer all challenges of handling complexity is core requirement [1] mitigating all vulnerabilities, using modern web scanning tools, some are automatic in nature some are manual, by using those tools, web applications could be secured [1].

Various difficulties that web application security testers are facing is because of insecure data storage facility, the initial stage where the website or web application is launched is itself insecure and not ready to defend against certain attacks [9].

Following are some of the major possibilities how web applications could be attacked or penetrated:

- Insecure login methods
- Weak or common passwords
- Insecure connection
- MITM
- Social engineering
- Unauthentic validation service
- Cookie stealing method.
- Hidden field mismanagement
- Unencrypted communication
- Parameter's handling

As web applications are scanned through automated tools, the scan is required to be thorough which is one of the biggest challenges yet [4] with every little advancement and upgrade in technology, previous tools are becoming useless or out of scope.

When an application start it's functioning, a lot of its activities are generated, hence, web application first check for requirement, then continue forward with three available options, first option is to understand the required connected peripheral devices and important component, simultaneously cooperate with software resources as well, when software resources are configured[5], all tools required for its setup are to be installed and ready to use, configuration phase is a bit technical phase, hence a technical staff is required to establish a secure an configured system for the DECISION TREE team to begin with DECISION TREE security testing environment[3] next is to test the security of connection, internet connection is often declined from this phase because of stable connectivity but that's not the reason, a stable

internet connection is not the guarantee of secure connection, security is not dependent on stability, so internet security is also to be noticed, if possible establish a private network connection to continue with processing[5].

SQLIVDT, a tool created by Zoran Djuric [6], is intended for efficient SQLI vulnerability detection. The tool's main purpose of this tool produce inputs and evaluate the results. Web applications flaws allow hackers to take advantage of unauthorized CCOUNT ACCESS to carry out malicious acts. Web application vulnerabilities have increased over the last decade. The black box strategy is focused on simulating SQL attacks on web apps.

Jose Fonseca, Nuno Seixas, Marco Vieira, and Henrique Madeira [7] analyzed 715 vulnerabilities and more than 100 exploits in 17 web apps. Some web apps were developed in a weakly typed language, while others were developed in a solid typed language. The executed goals, according to the paper, are weakly typed. The majority of web applications contain critical flaws that compromise their security. It is critical to comprehend their standard software flaws in order to avoid security issues. SQL Injection and XSS are the most common and dangerous web application flaws. This research examines the source code of the script used to determine how these vulnerabilities are exploited by hackers.

Mukesh Kumar Gupta, M.e.Govil, and Girdhari Singh [9] suggested a categories of information security techniques for developing stable software at different phases of the software development life cycle. Static analysis methods can identify the source of a security issue and detect bugs. Error detection not only lowers the cost of error, but it also increases the coding method. FN (False Negative) and FP (False Positive) outcomes plague the stable analysis method. The reliance on web applications has risen dramatically in recent years, posing a challenge as well as being used for other purposes. The most destructive security weaknesses exploited in web applications, such as eBay, and Facebook, Google are sql injection and XSS. Since they do not meet security guidelines, most developers make the same programming mistakes in their code.

Adnan Masood and Jim Java [10] addresses shortcomings of current situations and review emerging strategies and tools for improving service protection by finding vulnerabilities. A service name RESTFUL services now established themselves as the new technique for service growth. The OWASP top 10 vulnerabilities for web applications are summarized here. The possibility of using static code analysis to find vulnerabilities is addressed. Web services application layer vulnerabilities are a microcosm of the exploits open to attackers of web applications. Recursive threat modelling methods scan better prepare us to face and mitigate such attacks, as well as improve the stability of service-oriented architecture.

Iberia Medeiros and Nuno Neves [11] have provided a method for detecting and fixing bugs in websites, as well as a device for implementing the methods in PHP programs. Two methods are used: static source code analysis and data mining. Web application protection remains a difficult problem to solve. In this research, a variety of approaches are used to find bugs in source code with fewer false positives. For experimental evaluation, WAP tools are used, as

well as a wide number of PHP applications. They discovered 388 flaws approximately in 1.4 million LOC.

J. Pradeep Kumar, Dr. T. Ravi, and K. V. Nagendra proposed a system for modelling SOLIAs that uses the Augmented Attack TREE and regular expressions to find suitable SQL statements created by SQLIA adversaries [12]. It also identifies several security vulnerabilities in web applications and provides analysis to aid in the prevention of security flaws. It depicts various SOL Injection and Cross-Site Scripting attacks.

Wang Chunlei, Liu Li, and Liu Qiang [13] suggested a fuzz testing-based approach for identifying and analyzing web services vulnerabilities. which included identifying inputs, generating fuzz testing data, conducting fuzz testing, monitoring, and identifying normal fragility, among other things. Using WSFuzzer to perform web services vulnerability fuzz testing, many vulnerabilities such as SQL injection, knowledge leakage, and Xpath injection were discovered. It demonstrates that the approach introduced in this paper can effectively and efficiently assess the weaknesses of web applications. When conducting tests over the internet, firewalls and intrusion detection systems can interrupt web service requests, which has a significant impact on fuzz testing.

The use of a static code analysis method to identify weaknesses in plugins has been investigated by Jose Fonseca and Marco Vieira [8]. Many plugins currently in use around the world have dangerous CSS and SQL Injection flaws that can be abused, according to the findings. Using RIPS and PHPSAFE, this research examined the security flaws more than 30 WordPress plugins. More than 340 XSS and SOL unknown security flaws have been discovered, implying that plugins may be a source of security issues. The analysis of Static tools need to be more efficient, in terms of coverage and false positives. Many websites enable the incorporation of third-party (SSP) server-side plugins, which provide additional working while also opening the door to new vulnerabilities.

Sidra Shahbaz, Muddassar Masood, Ali Hur, Abdul Razzaq, and H Farooq Ahmad [14] conduct side by side comparison web apps firewall solutions with very important features required for layer of application protection. Due to their emphasis on the network layer and shortcomings in their core technology design, the available security solutions are ineffective. The various open source and commercial solutions available are making it difficult to choose the best solution for the protection of different systems. Users may use critical/sensitive analysis of WAF solutions to help them choose the best solution for their situation. As the number of web applications grows, protection becomes more flaw to a kind of threats. The success of these attacks is primarily due to application developers' ignorance when developing web applications and weaknesses in built in technologies.

Web applications have faced new vulnerabilities in security, as shown by Marcelo Invert Palma Salas, Paulo Licio de Geus, and Eliane Martins [15]. According to the technology is vulnerable to XML Injection flaw, which enable an hacker to gain access and manipulate information to add malicious code in either the client or server side, and is the most widely used attacks against web apps. This technique increases the robustness of web apps by allowing for more flexibility

in changing test cases and locating software bugs. 82 percent of web services tested were facing the vulnerablility to XML Injection attacks, according to the findings.

Theodoor Scholte, William Robertson, Davide Balzarotti, and Engin Kirda [16] presented a novel technique for avoiding the manipulation of XSS and SQL injection vulnerabilities based on automatic input parameter data type detection. The author developed IPAAS for PHP and tested it on five real-world web applications that had documented XSS and SQL injection flaws. According to the findings, IPAAS would have avoided 83 percent of SQL injection vulnerabilities and 65 percent of XSS vulnerabilities. Millions of users have come to rely on web applications in their everyday lives. However, attackers are still targeting these applications, and critical flaws are still popular.

The Web application introduced by Chandershekhar Sharma and Dr. S.C. Jain [17] communicates with the back end database to access data as and when requested by the user. SQL injection attacks are one of the most dangerous forms of vulnerabilities in database-driven web applications, and SQL injection vulnerabilities are the most severe. The description of SQL injection attacks is discussed in this paper, as well as an overview of the risk associated with each attack. The outcome of the study emphasises the effect of attacks on web application databases. This research can be extremely useful in the development of detection tools.

According to Seung-Jae Yoo and Jeong-Mo Yang [18], cyber hacking incidents and injuries have recently increased dramatically, causing serious harm to companies, society, and the national level. Web protection has emerged as a key topic in security research, as it has emerged as the most critical security. In this paper, the author develops a method for detecting unencrypted packets using real-time packet monitoring, as well as a proposal for implementing an encrypted system. More than 3000 pages, including subpages of the homepage, are gathered and analysed in this study. Around 2100 pages were not encrypted among them. This corresponds to an average likelihood of 80%.

To detect RCE vulnerabilities, Yunhui Zheng and Xiangyu Zhang [19] proposed a path and context sensitive inter procedural analysis. This paper develops a prototype framework and tests it on ten real-world PHP applications, identifying 21 true RCE vulnerabilities, including eight that have never been published before. To overcome the two sets of constraints simultaneously, a novel algorithm is created. The results show that the technique is very efficient at detecting RCE vulnerabilities in real-world PHP applications, with much less false positives than other methods. Remote code execution attacks are one of the most common web application security threats. It's a kind of cross-site scripting attack in which client inputs are stored and executed as server-side scripts.

Jan-Min Chen and Chia-Lun Wu [20] developed an automated vulnerability scanner to scan injection attack vulnerabilities. They suggested SQL injection and cross-site scripting attacks as mechanisms. They can easily see where the bug is, reduce debug time, and increase performance since they detect based on injection point.

A method for evaluating and comparing web application vulnerability scanners has been proposed by José Fonseca, Marco Vieira, and Henrique Madeira [22]. They demonstrated how

to use software fault injection techniques to test and benchmark automated site vulnerability scanners. The number of false positives is extremely high, varying from 20% to 77% in the studies conducted, and the findings indicate that coverage is generally poor.

Mattia Monga, Roberto Paleari, and Emanuele Passerini [23] have proposed a hybrid analysis technique for detecting injection vulnerabilities in web applications. They explained how Phan, a hybrid analyzer for PHP applications that operates directly at the Zendbytecode level, was designed and implemented. Their proposal combines the advantages of static and dynamic methods, with the preliminary static analysis process assisting in reducing the run-time overhead associated with dynamic monitoring. The preliminary results show that the change over a fully dynamic taint analysis is important.

A SQL injection vulnerability detection method based on secret web crawling has been proposed by Xin Wang, Luhua Wang, Gengyu Wei, Dongmei Zhang, and Yixian Yang [24]. They combine authentication with the crawler model and find SQL injection vulnerabilities by simulating web attacks and analyzing response data. They also conducted two experiments, the first of which was to compare the coverage of our tool with that of other three traditional scanners [10, l3, 24] by detecting three popular public web sites, and the result showed that the system we implemented can retrieve hidden wp-content.

The use of dataflow-based approaches to interpret the data has been proposed by Gang Zhao and Hua Chen [25]. In bytecode, there is a flaw in a Java program. A technique for exposing the global impact of data in a software is data-flow analysis. They spoke about the features of the program flaw. The connection between vulnerability and dataflow analysis is investigated. A structure for a dataflow-based analysis method has been suggested. Its implementation aims to produce Java bytecode.

The various methods used in phishing have been proposed in detail by Weider D., Yu Shruti Nargundkar, and Nagapriya Tiruthani [26]. They conduct a root-cause study of the phishing methods used, as well as the motivation for phishing, and create a fishbone diagram detailing the causes and methodologies used in phishing in the process. This research would aid developers in designing and developing better anti-phishing solutions.

Tânia Basso, Plnio César Simes Fernandes, Mario Jino, and Regina Moraes [28] presented an experimental study in which they investigated the impact of Java software faults inserted into Web application source code on security vulnerabilities. They also looked at how these flaws affected the identification of security vulnerabilities by a well-known commercial web security vulnerability scanner tool. Manual attacks based on attack trees were used to verify the scanner tool's performance.

Nuno Antunes and Marco Vieira [29] published an experimental study comparing a number of web vulnerability detection methods that use either penetration testing or static code. They demonstrated the effectiveness of these two methods in detecting SQL Injection vulnerabilities in web services code. The findings indicate that static code analyzers can detect more SQL Injection vulnerabilities than penetration testing tools in general, and that static code analysis tool coverage is usually much higher than penetration testing tool coverage.

METHODOLOGY

Vulnerability testing is the step in which the application is examined for security problems, technical defects, or vulnerabilities. This would necessitate the use of a penetration testing skill set. The issues that related to security discovered during this step must be written up and provided to Best Bank, along with remedial options. Various areas must be checked depending on the application. The following areas that should be checked for testing for a specific web app.

INJECTIN FLAWS

When a web application receives some data that we can't trust and delivers it to the compiler or interpreter, it is vulnerable to injection issues. Injection faults are one of the most prevalent and dangerous problems identified in many online applications. There are several injection issues depending on the underlying interpreter, such as SQL injection, SQLI, LDAP injection, command injection, and many more. From here on out, we'll focus on SQL injection, which is the most prevalent online application vulnerability.

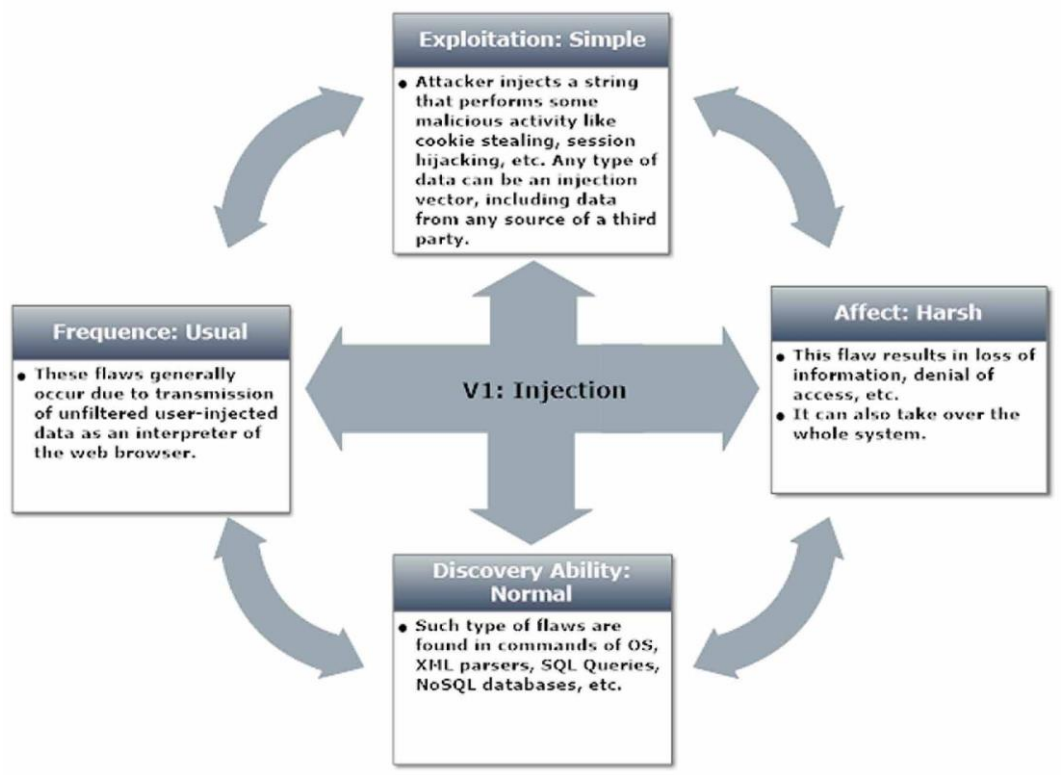


Figure. 5 INJECTIN FLAWS

1. Consider the case of a Best Bank customer wants to visit and looking to see his remaining amount in his account. When a person visit the account balance URL the user will be directed to the page where he/she can see the details of the amount.

The SQL query that is run in response to this is:

1. SELECT username, balance FROM userinformation WHERE userid = 001

2. SELECT username, balance FROM userinformaton WHERE userid = 001

As a result, if the value is tampered with, the interpreter will parse any value the attacker provides.

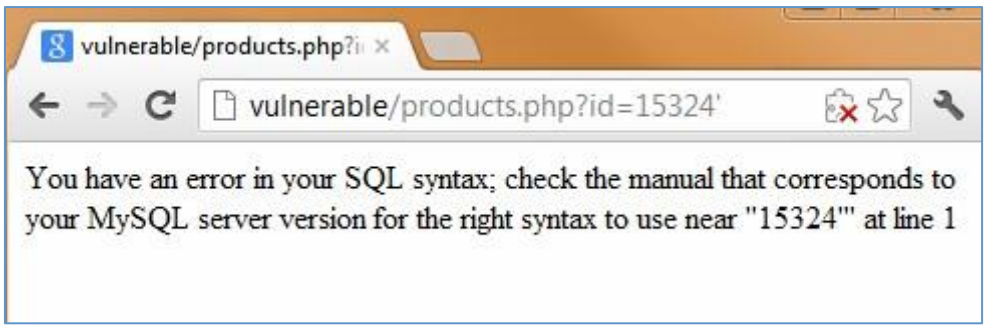
2. Consider the scenario in which an attacker injects something dangerous into your system hide or unavailable for the userif=001 by using ‘DROP TABLES’ and absence of separation b/w data and code.

Because the interpreter has no way of knowing whether or not the above command is malicious, it would execute it and dump the tables. As a result of the absence of separation between "data" and "code," injection vulnerabilities arise.

IDENTIFICATION

The main concern is if every interpreter usage clearly differentiates untrustworthy data from the entered command/query. This can be accomplished during pen testing by delivering input to the application and watching what’s the reactions. Look for places where the database can be accessed.

Inserting a single quote, to discover the presence of a SQL injection flaw, may result in an bug due to syntax difficulties, as demonstrated in the below picture:



Sometimes, even if the application does not react and show an error message and instead displays a custom page, SQL injection could occur. This situation is known as "Blind SQL Injection," and it must be discovered using yes or no criteria. SQL injection issues can be classified as Union-based or error-based, depending on the type of exploitation.

IMPACT

Injection vulnerability extremely dangerous since they allow an hacker to:

Access all database details modify/delete data in the database complete host takeover (depending on the situation)

A hacker/attacker can harvest passwords and other critical details from the DB by using a simple, well-crafted payload.

SUGGESTION:

SQLi problems avoided by keeping untrustworthy data separate from instructions.

By using an API that is safe, which skips the language processor completely or provides a partitioned interface, is the recommended approach. Be cautious when using APIs, it can still introduce injection behind the scenes.

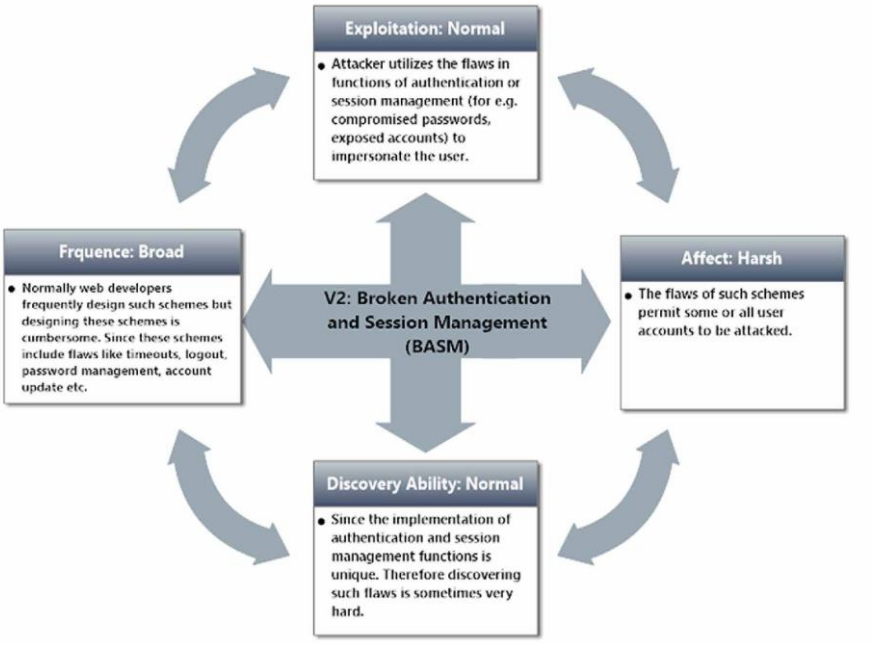
If you don't have access to a parameterized API, you need carefully skip the special symbols or characters using the interpreter's unique syntax. Many of these escape routines are available through OWASP's ESAPI.

Positive or "whitelist" input validation is also advised, although it isn't a comprehensive defense because most of the applications needed special characters. Whitelist input validation routines can be found in OWASP's ESAPI library, which is expandable.

BROKEN AUTHENTICATION AND SESSION MANAGEMENT

Authentication-related web application vulnerabilities arise when appropriate user authentication controls are not implemented properly. Users' accounts are at risk of being hacked as a result of this. Attackers may take advantage of these web security flaws to take control of any user account or even the entire system.

Credential Stuffing is one of these vulnerabilities, in which an attacker tests a list of valid passwords and usernames obtained from another hack or attack before they find a valid combination and gain access.



Example:

Because this vulnerability affects so many different locations, and it might be impossible to present a single sample that covers all possible situations. The following is a simplified illustration of how the session management problem in the password reset process might be explained.

When an attacker accesses the Best Bank app & selects the option from the options 'Forgot Password'.

Then he inserts the ID of a different user (for example, admin).

Following that, the attacker just queries an internal page, and log in as administrator. This happens because the app incorrectly sets the sessional attribute when the lost or forgot password procedure is started. The attacker gets the advantage of this and gain access by requesting it in a specific order. This is referred to as a "Session Puzzle" attack.

HOW TO IDENTIFY?

These problems can be found throughout the application. The following are some examples that can assist the user in spotting the many critical issues:

This occur when you are using the very weak account manageable functions (e.g., account creation, password change, password protection, and many more) that might lead to account accessible.

MADE SESSION IDS PUBLIC VIA THE URL.

If an app using the same session cookies before when he/ she authenticate himself and after authentication, it's subject to session fixation attacks. This would allow an attacker to repair the session before the user logs in and then hijack it. As a result, see if the program resets or creates the session cookie data following login.

SESSION NO TIMED OUT

Sometimes session is not terminated on main server after logout:

When a user logs out, the session should ideally be terminated on the server. Nonetheless, some developers make the mistake of deleting the cookies on the client side while leaving the session on the server active. This may be determined by logging out of the application and using a proxy tool such as Burp to re--send an earlier captured request. It ensures that the session has not been ended if the program still returns a valid response after logging out.

IMPACT

A hacker/attacker can get the access some or all of the users' accounts by exploiting session management weaknesses. An attacker can do anything a regular user can do after the account is compromised. Depending on the vulnerability in question, the possibility of these problems occurring varies substantially. An attacker might, for example, take advantage of a bad implementation of forgot password capability to completely compromise a victim's account.

SUGGESTION:

It is critical to have suitable authentication and session management measures in place to combat this issue. The following are some of the most essential considerations that developers should make in order to address this problem: Establishing a session: It's critical to determine when and how the application establishes and maintains a valid session. Account breach can occur if session tokens are created before authentication and session attributes are created in the improper places.

Creating session tokens: Tokens should be random and unpredictable because they are used to identify users. An attacker can use any guessable pattern inside the tokens to get access to sensitive information.

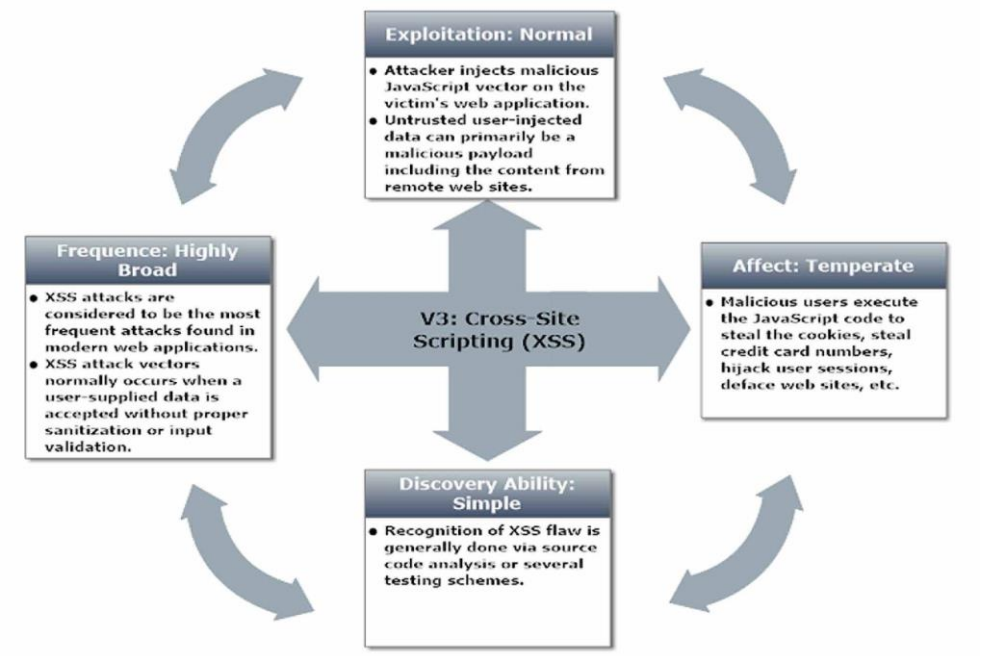
Session tokens are not disclosed in URLs or presented in responses. If they are looking to sent in the response of any body, other attacks like XSS can access and steal them.

Session termination: Most developers handle session ending incorrectly by simply deleting cookies on the side of the client. If an attacker gets the tokens, he can connect to the server. As a result, it's critical that the server session be correctly terminated.

Cookie Properties: Cookie attributes' increase the security of session tokens. The HTTPOnly attribute, for example, ensures that a cookie cannot be accessible by a client-side script used in cross-site scripting (XSS) attacks. Similarly, the secure attribute prevents cookies from being sent via an insecure connection.

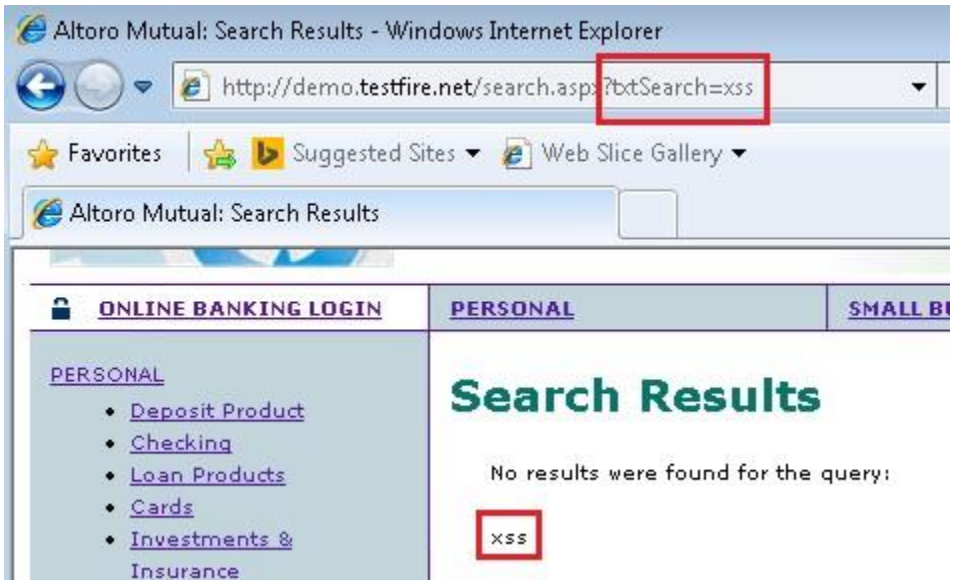
CROSS SITE SCRIPTING

In online application security testing, cross-site scripting (XSS) is the most common concerns. XSS issues occur when a web app accepts the input entered by the user and then send this back to the user without properly validating it. In short, by supplying scripts as input to the application, an unauthorized person will get the access and able to execute scripts in the browser that used by the end user. The following example can assist you in better understanding the problem.



Example:

Consider a Best Bank search page that accepts input after those returns it to the user. Enter the text "xss" as an example and notice how the text mirrors the same page as displayed in the image.



This value now resides in the result, sandwiched b/w certain tag of html as shown in the given screenshot:

```
<div class="fl" style="width: 99%;">

<h1>Search Results</h1>

<p>No results were found for the query:<br /><br />
<span id="_ctl0__ctl0_Content_Main_lblSearch">xss</span></p>

</div>
```

As a result, introducing specific scripts result, shown in their execution, as browser access the html normally ignorant that it was sent by an unauthorized user.

IDENTIFICATION

Before we go into how to spot XSS flaws, it is crucial to understand that there are many types of XSS but most common three different types of XSS flaws/vulnerabilities given below:

Reflected:

The data entered is reflected on the page that is same.

Saved:

The data that is inserted is saved in the repository of database and return when the user returns to the page later. (For instance, user comments in discussion boards.)

DOM:

By using this example, we can observe that the value entered does not reach the destination i.e server; the data source is in the DOM, as is the sink, and the data flow never leaves the browser. Look for places where the user's input is reflected in the answer during pen testing. Check to see if the program verifies the input once the reflection scenario has been found. Try to send the special symbols or characters like, >, <, # e.t.c. to see if the application replies differently. If this is not the case, frame a payload to test for XSS based on the response location.

IMPACT:

XSS vulnerabilities take advantage of the user's trust in a website. Because he trusts the domain, the user clicks on the given link supplied to him by the attacker. An attacker can use XSS to steal session cookies, record keystrokes, redirect visitors to malicious sites, and deface websites by exploiting it.

SUGGESTIONS:

XSS can be remedied by using one of the two methods listed below. :

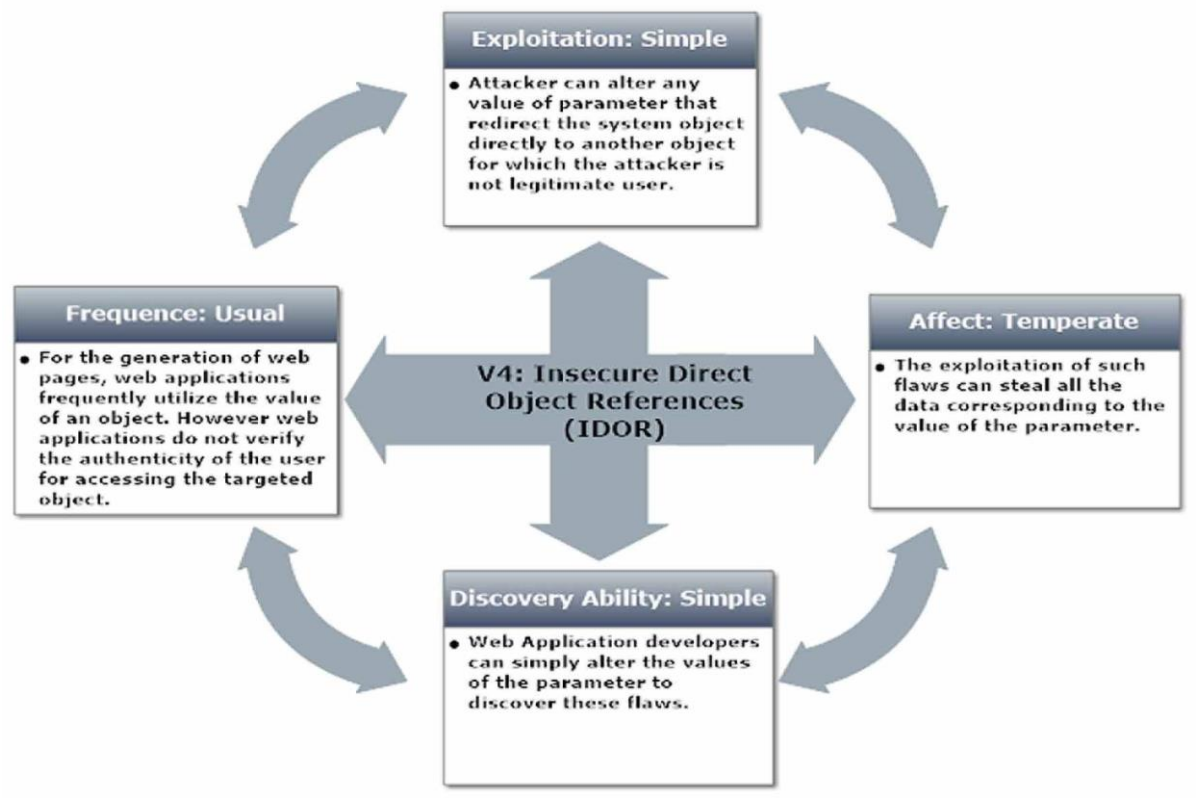
Only allow Validate the input and only allow those characters that are relevant to the situation. If you're validating, don't use blacklisting because hackers will always find a way around the filters. Whitelisting, which allows only a small number of known values, is suggested instead.

Output Encoding:

Before returning variable output to the end user, applications must guarantee that all variable output on a page is encoded. Encoding variable output replaces HTML markup with entities, which are different representations of HTML markup. The entities are seen in the browser, but they are not run. For example the given tag of <script> gets converted to <script>.

INSECURE DIRECT OBJECT REFERENCES

When creating web pages, applications frequently refer to their resources by their real names. Before allowing access to any of the internal resources, developers must implement permission checks in their application. If no validation checks are in place, an Hacker/Attacker can simply tamper with different parametric values to gain the access to unapproved websites that may contain critical information.



EXAMPLE:

According to the following scenario: A user logon into his personal Best Bank account and clicks on the ‘Account Balance Check' link:

ViewBalance.jsp?user=tom http://www.mybestbank.com

If the developed application does not have any authorization checks in place, Hacker can tamper with the data of the ‘user' and access the parameter and see the balance of another user (say John) by requesting the URL below:

http://www.mybestbank.com/ViewBalance.jsp?user=John

IDENTIFICATION

If the program has proper authorization checks in place, insecure direct object references can be recognized. Try changing the values of the parameters (GET/POST) as described in the example above to see how the program reacts.

What are the consequences?

The consequences of such flaws can range from mild to severe. It is dependent on the type of data that the object reference refers to. Any sensitive personal information that is exposed because of these flaws might be extremely damaging.

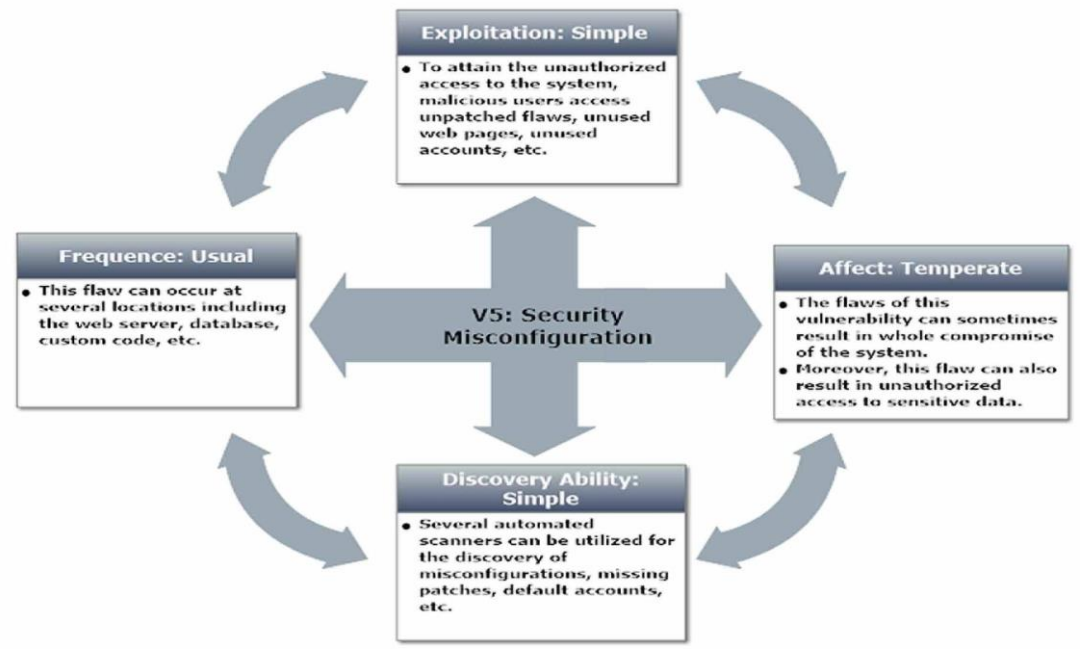
SUGGESTIONS:

The following steps can be taken to avoid insecure direct object references:

1. Instead of using the references of object directly, the developed application can use IDs (numbers ranging from 1 to any number) for each user, which correspond to a database key on the server. As a result, Hacker are unable to directly access his target by using illegal resources in this situation.
2. Before granting access to the application's internal resources, make sure it has sufficient access control or authorization checks in place.

SECURITY MISCONFIGURATION

This is due to misconfigured platform, web server, database, application server, and other settings, as well as missing security patches, default accounts, and other issues.



Example:

Consider the circumstance where directory listing on the BestBank server is not deactivated. If an attacker learns this flaw, all he has to do is look through the folders to identify any file on the Best Bank Server. After that, the Hacker/Attacker can access the critical data and download any confidential documents or source code files on the database server, and then trying to proceed to identify a significant access control hole in your program.

IDENTIFICATION:

Pen testers should investigate the application's security hardening posture by asking the following questions:

1. Check for information leakage in the application, such as information on the program being used and its versions. For example, the version of a Server response header could be leaked. Similarly, we can determine the current version of the database in use by exploiting SQL injection vulnerabilities.
2. Use of superfluous features: Unnecessary features (such as ports, services, pages, accounts, and privileges) could be activated. Run tools like nmap to look for them.
3. Default accounts and passwords: Does the program allow you to utilize default accounts and passwords? Check for default usernames and passwords for SAP accounts, for example, if the application uses SAP.
4. Error messages and stack traces may reveal information if the application is not correctly setup. This can be problematic since they could aid an attacker in exploiting a different vulnerability.
5. Verify that the security settings in the server, development frameworks, and libraries are not set to secure values. Some application servers, for example, include sample pages that have been reported as having vulnerabilities. As a result, check to see if the settings have been modified to prevent access to certain pages.

The preceding is not an exhaustive list, but it serves as a good starting point. Depending on the underlying application and technology, the pen tester must determine whether there are any alternative methods that may be investigated in order to acquire information about misconfigurations.

IMPACT:

The consequences can be severe, as some of the flaws can lead to entire system compromise without the user's knowledge. For example, if a server has a heart bleed vulnerability, an attacker could gain access to sensitive data stored in the server's memory

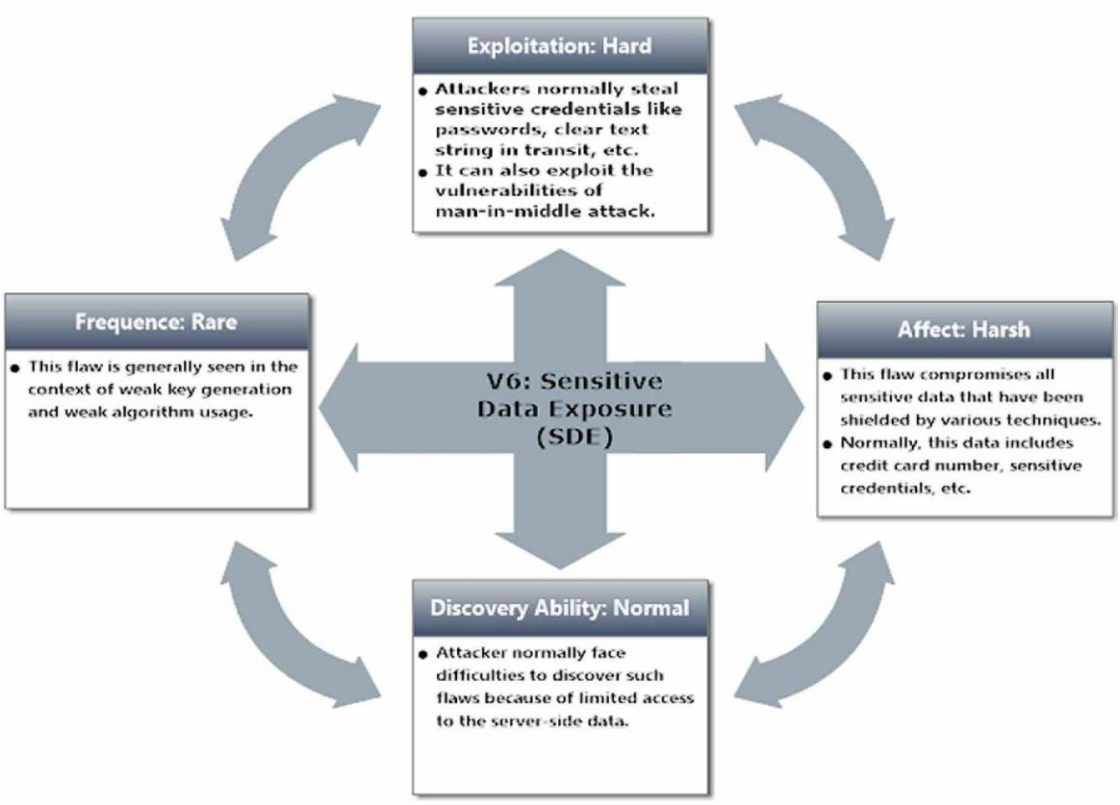
SUGGESTIONS:

The process of security hardening should be ongoing in order to keep systems up to date with the most recent updates. Also, while customizing, make sure that the default accounts and

passwords are modified. Running code scans and audits on a regular basis can also assist in finding new threats that develop daily.

SENSITIVE DATA EXPOSURE

This is among the most frequent vulnerabilities that may be identified on most websites. This vulnerability describes a situation in which sensitive data is not encrypted and is exposed. Even if the data is encrypted, there is a potential that the algorithm is flawed or that the key generation and administration are ineffective.



Example:

Assume that the BestBank application uses automatic database encryption to encrypt credit card numbers in a database. This does, however, mean that when the data is retrieved, it is also automatically decrypted. An attacker may get credit card numbers in clear text if there was a SQL injection problem in this application. Instead, BestBank should have used a public key to encrypt the credit card data and only permitted back-end applications to decrypt them with the private key.

IDENTIFY

It's crucial for a pen tester to determine which data is sensitive for a certain application (for example, passwords, credit card numbers, and salary information). The next step is to determine the level of security required in each situation. Passwords, for example, should be hashed and saved in the database (rather than encrypted).

As a result, after identifying sensitive data, try to answer the following questions:

1. Is the sensitive information on the server stored in clear text? Database administrators and attackers would be able to access the data as a result of this. Unless you use a vulnerability like SQL injection to acquire information from the database, this may be difficult to detect.
2. Is there a place in the program where sensitive data is logged?
3. What algorithms does the program employee if sensitive data is encrypted? Are they safe or do they have flaws? Is the key size long enough?
4. Is there a lack of security directives or headers in the application that handle security vulnerabilities at the browser level? The pages are cached in the browser when there are no correct caching headers in the response, for example. The response header acquired by the proxy tool can be used to confirm this. Any sensitive information on these pages could be accessed by an attacker in the future.

IMPACT:

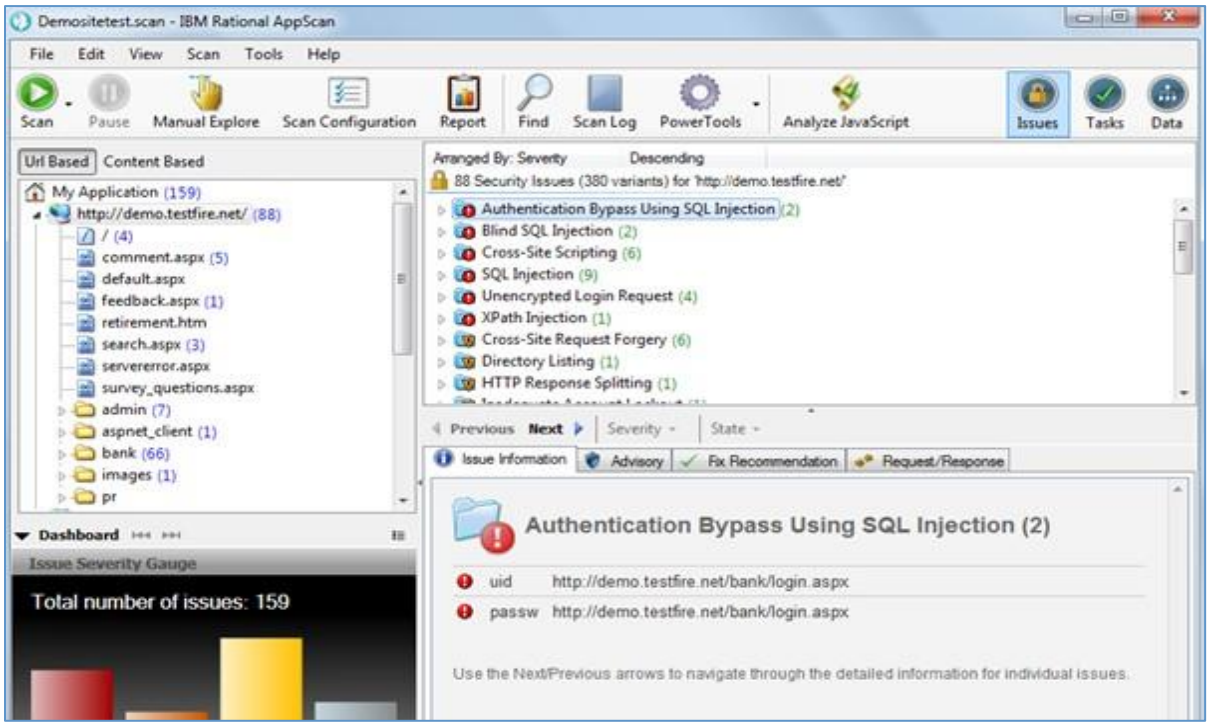
Because this is a matter of sensitive information, the consequences are always significant. However, it varies widely depending on the application and the sort of organization.

SUGGESTIONS:

The following suggestions may be useful in resolving some issues:

1. Determine which areas deal with sensitive data and safeguard it: The first step is to determine which areas deal with sensitive data. It could contain personal information such as Social Security numbers, credit card numbers, and passwords, among other things. The next step is to verify that this data is encrypted both during transmission and when it is stored.
2. Remove needless storage: Sensitive data should only be used when absolutely essential. Don't keep critical information around unnecessarily. It should be thrown away as quickly as feasible.
3. Strong algorithms and keys should be used with sensitive data. Broken methods like MD5 should not be utilized. When encrypting values associated with sensitive data, only robust algorithms should be utilized. The length of the keys should be sufficient to prevent brute force attacks.
4. Password Hashing: Passwords must be hashed before being stored. Even if an attacker obtains access to their values, he will not be able to reverse them to obtain the original plain text values. Consider employing the 'salt' mechanism, which adds an additional degree of security to the password hashes.

The following is a screenshot of the IBM Appscan program, which reports on numerous vulnerabilities:



Limitations and scope

Although the number and types of attacks increases with the passage of time. It is very difficult to conclude all the attacks because research on web security is still very low. Many methods used to scan weaknesses in web apps for example Automated web apps vulnerability Scanners and Manual way detect vulnerabilities in web applications.

Web apps vulnerability detectors are automatic tools that search web apps for signatures of well-known security flaws like XSS, SQL injection, and many more. There are a number of paid and free vulnerability/flaws scanners available in the market; here is a list of the most popular:

- AppScan Automated scanner
- Burp Suite Automated scanner
- Nessus Automated scanner
- NeXpose Automated scanner
- WebInspect Automated scanner
- Websecurify Suite Automated scanner
- Zed Attack Proxy Automated scanner

Some of these scanners only need a connection to the destination websites, and others need some changes before they can be used. Before running the test, many of them have latest configuration options that enable the clients to use the scanner (disable unnecessary parts, set the highest number of vulnerabilities, customize test and critical, non-critical levels, and so on).

Scope and Limitations of automated vulnerability scanners are:

1. The majority of the XSS vulnerabilities where malicious for JAVA (JavaScript) code can be inserted into a client request and provided in the answer.
2. Any of the other XSS vulnerabilities where malicious JavaScript is saved on the server and shown if anyone (the attacker or an unwitting user) requests a particular page.
3. There are very few DOM-based XSS vulnerabilities – where client-side JavaScript uses user-controllable data.
4. An known as the Path traversal flaws/weaknesses arbitrary read of files on the insecure website server.
5. Vulnerabilities in file inclusion any file that access from through internet may be included in the answer.
6. By inserting a command that will either provide the answer too late or give the unique output in the client's request, command injection vulnerabilities can be exploited (i.e. ipconfig)
7. Pages and directories that are hidden.
8. Webserver vulnerabilities are listed in this directory.
9. Other vulnerabilities can be identified by examining the web application's requests and answers.

Aside from the drawbacks, some scanners/detectors are not intelligent enough to search for complex bugs in application logic. Vulnerability scanners for web applications are unable to search for:

1. Authentication weaknesses such as username evaluation, password guessing avoidance, any account recovery features, credential predictability, or any other logic flaws in the authentication process.
2. Vulnerabilities in access controls where a user with no admin rights may access other users' data or admin features.
3. Ordering a negative number of objects, missing a stage in a multistage process (i.e. heading straight to the delivery webpage, skipping the payment page), and other application logic flaws.
4. Vulnerabilities in shared hosting check for Errors in shared infrastructure and between ASP-hosted apps.
5. Leakage of confidential data such as user logs in secret areas.

To summaries, automatic web app scanners are critical devices for any flaws/vulnerability evaluation or penetration test, but their strengths and limitations should be understood. However, a competent penetration tester must be able to interpret the data, conduct advanced manual testing, and comprehend the organization's risks.

RESULT & DISCUSSION

There are a variety of explanations for these systems' inaccessibility, including power outages and a lack of or restricted public interface. We tried to gather as much information as possible about the remaining 413 websites. However, there were some highly sensitive systems for which we lacked approvals, and therefore could not be grouped into any of the categories using the proposed method. Thirty-one percent of the overall systems were classified as "Not Classified." The remaining systems were thoroughly examined, with all relevant data collected for the categories identified in the proposed framework.

All of the necessary information was gathered, and the criteria for the systems' security classification were determined. Based on our estimates, 82 percent of the systems were assigned a Medium Security rating, as shown in Fig. 5. Static sites make up a large portion of these networks, with integrity and availability as the primary criteria. In our study, however, (6 percent) assets (applications) were classified as Large. This particularly includes that communicate with citizens' private information and information, as well as systems that are shared among departments. As a result, applications in the High category have higher security standards and can get more attention when it comes to testing and integrating security features. Just 12% of the systems scored in the Low category. This group includes applications that are freely accessible to the general public and do not have any security specifications.

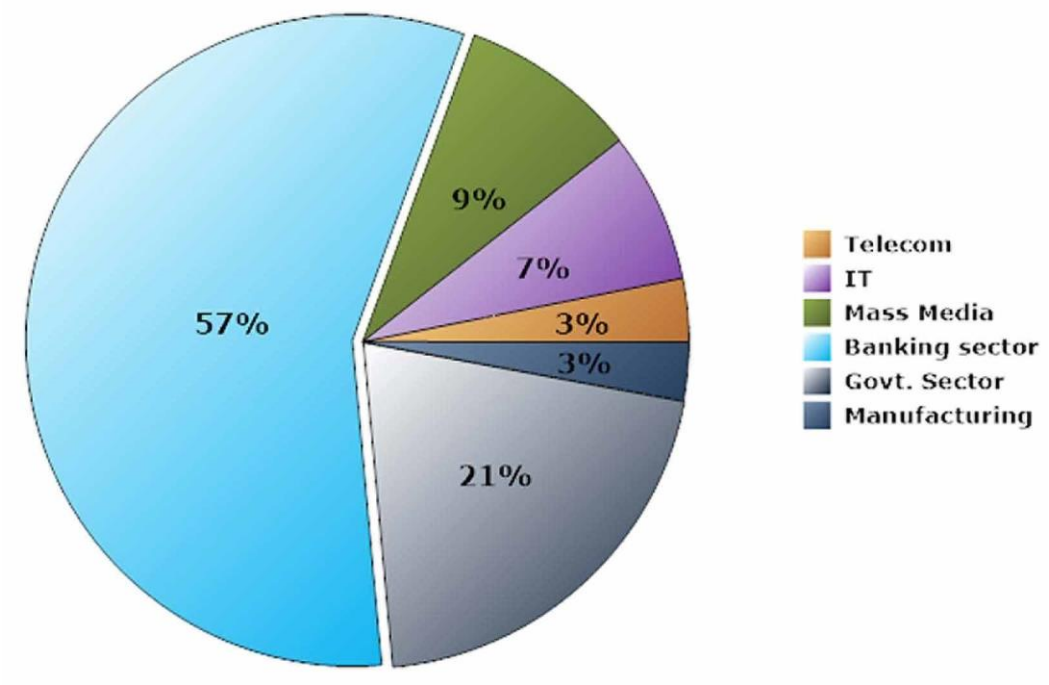


Figure. 6 %age of Vulnerability in different organizations

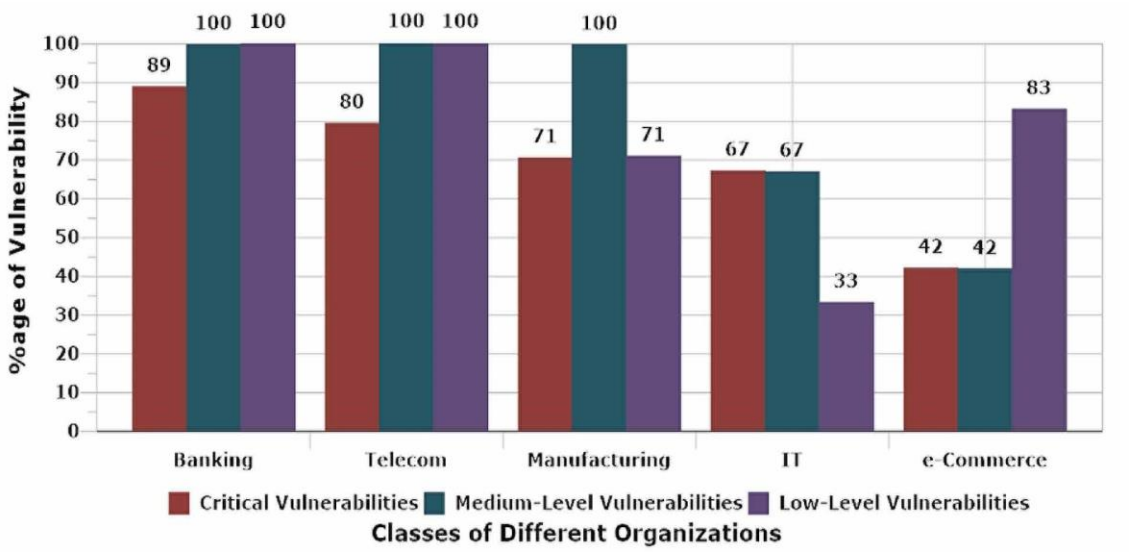


Figure. 6.1 Vulnerability in different organizations

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

Figure. 6.2 Overall Risk Severity in different organizations

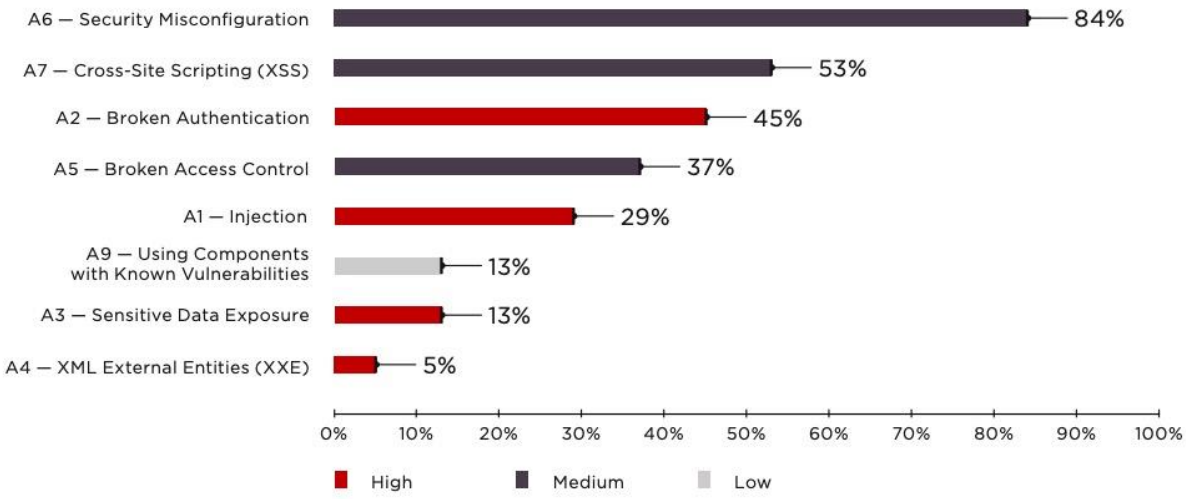


Figure. 7 Top 10 OWASP Vulnerability level

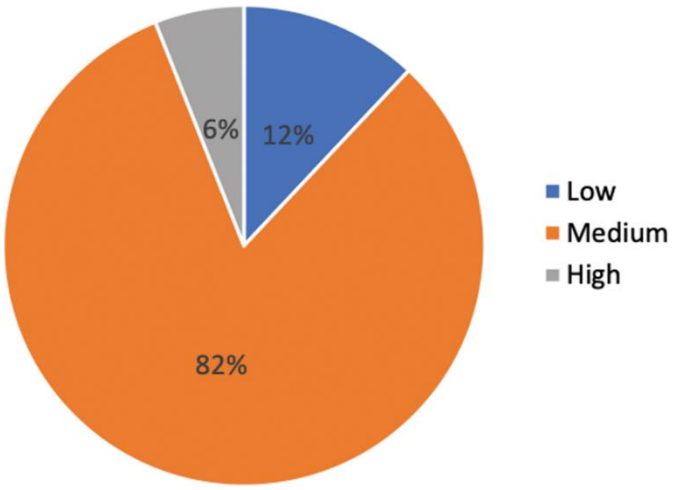


Figure. 8 Security level classification of the studied systems

Testing of Study Cases

The methodology we use it also involves web-based information system security research. For vulnerability checking, we selected a selection of 20 systems from various organizations and categories. We also made sure to choose systems from both the High level and medium level Asset Categories, as determined by our proposed asset classification method.

It is worth noting that the different flaws/vulnerability testing was done with manually methods, without the use of any automated testing software, to ensure that these systems were not harmed or disrupted. Table 1 present some basic details about the dummy applications that were checked for security flaws. The actual Uniform Resource Locators of the pages are anonymized for confidentiality purposes, as can be seen in the chart. The output of our vulnerability checking on the sampled applications are summarized in Figure 6. Broken access control is seen in many applications, followed by sensitive data leakage, using elements with known flaws/vulnerabilities, and security misconfiguration, as shown in the diagram. Cross-site scripting (XSS), SQL injection, and broken authentication, on the other hand, are only present in a few applications.

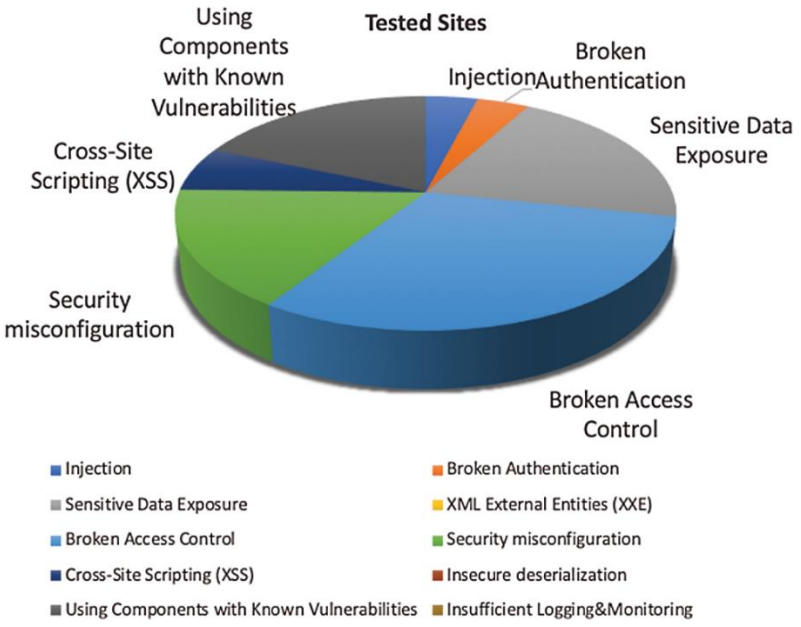


Table. 1 contains the fully detailed results of flaws/vulnerability testing, i.e., for every domain, with the site name removed. Broken authentication, SQL injection, and XSS (cross-site scripting) were found to have the least number of vulnerabilities in 1, 2, and 3 applications, respectively, according to the chart. More than half of the web applications analyzed had vulnerabilities such as critical data leakage, broken access control, and the use of important components with known flaws/vulnerabilities.

Site	Injection	Broken authentication	Sensitive data exposure	XML external entity	Broken access control	Security misconfiguration	Cross-site scripting	Insecure deserialization	Using components with known vulnerabilities	Insufficient logging & monitoring
Site 1	No	No	Yes	–	Yes	Yes	No	–	Yes	–
Site 2	No	No	Yes	–	Yes	Yes	No	–	Yes	–
Site 3	No	No	Yes	–	Yes	Yes	No	–	Yes	–
Site 4	No	No	Yes	–	Yes	No	No	–	No	–
Site 5	Yes	No	Yes	–	Yes	Yes	Yes	–	Yes	–
Site 11	No	No	Yes	–	Yes	No	No	–	No	–
Site 12	Yes	No	Yes	–	Yes	Yes	Yes	–	Yes	–
Site 13	No	Yes	No	–	Yes	Yes	Yes	–	Yes	–
Site 14	No	No	Yes	–	Yes	No	No	–	Yes	–
Site 15	No	No	Yes	–	Yes	No	No	–	Yes	–
Site 16	No	No	No	–	Yes	No	No	–	No	–
Site 17	No	No	Yes	–	Yes	Yes	No	–	No	–
Site 18	No	No	Yes	–	Yes	No	No	–	Yes	–
Site 19	No	No	Yes	–	Yes	Yes	No	–	Yes	–
Site 20	No	No	No	–	Yes	No	No	–	No	–

CONCLUSION & FUTURE WORK

A structure for Asset Security Classification of Information Systems is introduced in this research. The technique is focused on the compilation and review of a large amount of data about the System Under Evaluation, as well as the assignment of a security measures (High, Medium, Low) to the SUT in order to priorities security measuring. The system uses mathematical equations to measure the SUT's confidentiality, credibility, availability, and exposure level before automatically assigning a protection level to it. We then ran security tests on a subset of the prioritized systems to see whether any of the OWASP (Open Web Application Security Project) Top 10 vulnerabilities is present. Our findings showed that these systems have a number of weaknesses. We talked about different ways to exploit web applications and webservers. As you may have noticed, most of the attacks we carried out were successful due to a lack of input validation, whether it was a SQL injection, RFI, LFI, or XSS. Most of these weaknesses arise due to the programmer being not able to successfully sanitize/filter the consumer input.

The proposed approach is advantageous because it gives organizations a comprehensive description of the security standards for their assets and allows them to priorities which assets should be tested for security. Even though the focus of this research is on prioritizing and testing web-based systems, the proposed methodology is general and can be applied to other types of systems with minor changes. Furthermore, the proposed architecture can be automated in the future to gather all data from web-based applications automatically, analyze it, and allocate an acceptable security level.

Reference and Bibliography

[1] http://www.acunetix.com/websitesecurity/webapplications/

[2] Nadya ElBachir El Moussaid, Ahmed Toumanari, (2014), "Web Application Attacks Detection: A Survey and Classification", International Journal of Computer Applications Volume 103, No.12.

[3] Http://phpsecurity.readthedocs.org/en/latest/Cross-SiteScripting-(XSS).html. [Accessed on February 2016]

[4] Katkar Anjali S., Kulkarni Raj B, (2012), "Web Vulnerability Detection and Security Mechanism", International Journal of Soft Computing and Engineering, Volume-2, Issue-4.

[5] Gopal R. Chaudhari, Prof. Madhav V. Vaidya, (2014), "A Survey on Security and Vulnerabilities of Web Application", International Journal of Computer Science and Information Technologies, Vol. 5 (2).

[6] Zoran Djuric, (2013), "A Black-box Testing Tool for Detecting SQL Injection Vulnerabilities", IEEE Second International Conference on Informatics and Applications, pp. 216- 221.

[7] Jose´ Fonseca, Nuno Seixas, Marco Vieira, and Henrique Madeira, (April 2014), "Analysis of Field Data on Web Security Vulnerabilities", IEEE transactions on dependable and secure computing, Vol. 11, No. 2, pp. 89- 100.

[8] José Fonseca and Marco Vieira, (2014), "A Practical Experience on the Impact of Plugins in Web Security", IEEE 33rd International Symposium on Reliable Distributed Systems, pp. 21-30.

[9] Mukesh Kumar Gupta, M.E. Govil and Girdhari Singh, (May 2014), "Static Analysis Approaches to Detect SQL Int. J. of Comp. & Info. Tech., (2016) 4(2): 35-42.

[10] Adnan Masood, Jim Java, (2015), "Static Analysis for Web Service Security – Tools & Techniques for a Secure Development Life Cycle", International Symposium on Technologies for Homeland Security, pp. 1-6.

[11] Ibéria Medeiros, Nuno Neves, (2015), "Detecting and Removing Web Application Vulnerabilities with Static Analysis and Data Mining", IEEE TRANSACTIONS ON RELIABILITY, pp.1-16.

[12] J. Pradeep Kumar, Dr. T. Ravi and K. V. Nagendra, (2012), "Analysis of security vulnerabilities for web based application" IEEE, pp. 233- 236.

[13] Wang Chunlei, Liu Li, Liu Qiang, (2014), "Automatic fuzz testing of web service vulnerability" International Conference on Information and Communications Technologies, pp. 1-6.

[14] Abdul Razzaq, Ali Hur, Sidra Shahbaz, MuddassarMasood, H Farooq Ahmad, (2013), "Critical Analysis on Web Application Firewall Solutions", IEEE Eleventh International Symposiumon Autonomous Decentralized Systems, pp. 1-6.

[15] Marcelo Invert Palma Salas, Paulo Lício de Geus, Eliane Martins, (2015), "Security Testing Methodology for Evaluation of Web Services Robustness - Case: XML Injection", IEEE World Congress on Services, pp. 303- 310.

[16] Theodoor Scholte, William Robertson, Davide Balzarotti, EnginKirda, (2012), "Preventing Input Validation Vulnerabilities in Web Applications through Automated Type Analysis", IEEE 36th International Conference on Computer Software and Applications, pp. 233- 243.

[17] Chandershekhar, Dr. S.c. Jain, (2014), "Analysis and Classification of SQL Injection Vulnerabilities and Attacks on Web Applications", IEEE International Conference on Advances in Engineering & Technology Research, pp. 1-6.

[18] Seung-Jae Yoo, Jeong-Mo Yang, (2014), "Web login Vulnerability Analysis and Countermeasures", International Conference on IT Convergence and Security, pp. 1-4.

[19] Yunhui Zheng and Xiangyu Zhang, (2013), "Path Sensitive Static Analysis of Web Applications for Remote Code Execution Vulnerability Detection", IEEE 35th International Conference on Software Engineering, pp. 652- 661.

[20] Jan-Min Chen, and Chia-Lun Wu, (2010), "An Automated Vulnerability Scanner for Injection Attack Based on Injection Point", International Computer Symposium (ICS), pp. 113-118.

[21] Ha Thanh Le and Peter Kok Keong Loh, (2008), "Evaluating AVDL Descriptions for Web Application Vulnerability Analysis", ISI 2008, June 17-20, 2008, Taipei, Taiwan, IEEE, pp. 279-281.

[22] José Fonseca and Marco Vieira, Henrique Madeira, (2007), "Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks", 13th IEEE International Symposium on Pacific Rim Dependable Computing, pp. 365-372.

[23] Mattia Monga, Roberto Paleari, Emanuele Passerini, (2009), "A hybrid analysis framework for detecting web application vulnerabilities", ICSE'09 Workshop, Vancouver, Canada, IEEE, pp. 25- 32.

[24] Xin Wang, Luhua Wang, Gengyu Wei, Dongmei Zhang, Yixian Yang, (2010), "Hidden web crawling for SQL injection detection", 3rd IEEE International Conference on Broadband Network and Multimedia Technology, pp. 14- 18.

[25] Gang Zhao and Hua Chen, (2008), "Data-flow Based Analysis of Java Bytecode Vulnerability", The Ninth International Conference on Web-Age Information Management, pp. 647- 653.

[26] Weider D., Yu Shruti Nargundkar and Nagapriya Tiruthani, (2008), "A Phishing Vulnerability Analysis of Web Based Systems", IEEE Symposium on Computers and Communications, 2008. Pp. 326- 331.

[27] Lwin Khin Shar, Lionel C. Briand and Hee Beng Kuan Tan, (2013), "Web Application Vulnerability Prediction using Hybrid Program Analysis and Machine Learning", IEEE 36th International Conference on Computer Software and Applications, pp. 1-35.

[28] Tânia Basso, Plínio César Simões Fernandes, Mario Jinoand Regina Moraes, (2010), "Analysis of the Effect of Java Software Faults on Security Vulnerabilities and Their Detection by Commercial Web Vulnerability Scanner Tool", International Conference on Dependable Systems and Networks Workshops (DSN-W), pp. 150- 155.

[29] Nuno Antunes, Marco Vieira, (2009), "Comparing the Effectiveness of Penetration Testing and Static Code Analysis on the Detection of SQL Injection Vulnerabilities in Web Services", 15th IEEE Pacific Rim International Symposium on Dependable Computing, pp. 301- 306.

[30]. Clarke-Salt. (2012). *SQL Injection Attacks and Defense*. Burlington, MA, USA: Elsevier.

[31]. D. Huluka and O. Popov. (2012). "Root cause analysis of session management and broken authentication vulnerabilities," in Proc. of WorldCIS-2012, Guelph, ON, pp. 82–86.

[32]. A. Andreu. (2006). *Professional Pen Testing for Web Applications*. New York, USA: John Wiley & Sons Inc., . [Online]. Available: https://www.amazon.com/Professional-Pen-Testing-Web-Applications/dp/0471789666.

[33]. KPCERC Cyber Emergency & Response Center, "KP ITBoard," 2018. [Online]. Available: https://kpcerc.com/.

[34]. Common Vulnerability Scoring System (CVSS). (n. d.). Retrieved from https://www.first.org/cvss