Information Security Criticality Testing and Remediation

Submitted By

Rajvi Contractor

16MCEI24

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2018

Information Security Criticality Testing and Remediation

Major Project

Submitted in fulfillment of the requirements

for the degree of

Master of Technology in Computer Science & Engineering

(Information & Network Security)

Submitted By

Rajvi Contractor

(16MCEI24)

Guided By

Asst. Prof. Sapan H Mankad

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2018

Certificate

This is to certify that the Major Project entitled **"Information Security Criticality Testing and Remediation"** submitted by **Rajvi Contractor (Roll No: 16MCEI24)**, towards the fulfillment of the requirements for the award of degree of Master of Technology in Computer Science & Engineering (Information & Network Security) of Nirma University, Ahmedabad, is the record of work carried out by her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project part-I and part-II, to the best of my knowledge, havent been submitted to any other university or institution for award of any degree or diploma.

Prof. Sapan H Mankad
Guide & Assistant Professor,
IT Department,
Institute of Technology,
Nirma University, Ahmedabad

Dr. Sharada Valiveti
Coordinator M.Tech - INS,
CE Department
Institute of Technology,
Nirma University, Ahmedabad

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

Dr. Alka Mahajan
Director,
Institute of Technology,
Nirma University, Ahmedabad

Statement of Originality

I, **Rajvi Contractor, 16MCEI24**, give undertaking that the Major Project entitled ”**Information security Criticality Testing and Remediation**” submitted by me, towards the fulfillment of the requirements for the degree of Master of Technology in **Information & Network Security** of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made.It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

_____________-

Signature of Student

Date: 18 May, 2018

Place: Ahmedabad

Endorsed by

Asst. Prof. Sapan H Mankad

(Signature of Guide)

Acknowledgements

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

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

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

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

- Rajvi Contractor

16MCEI24

Abstract

Application security has turned into an essential piece of Information security, it is never again discretionary.Now a days organizations have welcome building up their own application, so as the information related with the application expands, security has turned into the significant worry for application security.The Application Security helps you secure applications throughout the development and maintenance of the code.Securing Applications helps you to secure the code from vulnerabilities. Now a days, attacker perform Heart-bleed, SQL injection, Cross site scripting, Cross Site Request Forgery, and many other attacks on the application, which causes a loss of sensitive data of that application.For protecting this kind of sensitive information source code review should be done. Source code review is a process to identify and remove security risks of the applications which contains sensitive information related to company. When developer develops an application, who don't have knowledge of how to code securely then this will create a big security hole for the application and attacker can take advantage of this security weakness. Some of the attacks are considers false negatives which your system cannot Identify. These are the most dangerous attacks nowadays. This can result in to big damage to the company and its reputation.So the idea is to develop the system which will help the AppSec team to for reducing their daily manual work and protect the application from the attackers. So I am developing the web Portal for the Application security team which will reduce the manual work for the Bug Bounty program, where external researcher can report the vulnerabilities which they have found during their research work.

Abbreviations

SAST	Static Application Security Testing
DAST	Dynamic Application Security Testing
XSS	Cross Site Scripting
OWASP	Open Web Application Security Project

–

Contents

List of Figures

Chapter 1

Introduction

1.1 Application Security

Application security has turned into an essential piece of Information security, it is never again discretionary.Now a days organizations have welcome building up their own application, so as the information related with the application expands, security has turned into the significant worry for application security.The Application Security helps you secure applications throughout the development and maintenance of the code.It helps to secure the code from vulnerabilities.

Security of an application is becoming important concern as more and more applications are uploading on the network.Now a days, attacker perform Heart-bleed, SQL injection, Cross site scripting, Cross Site Request Forgery, and many other attacks on the application, which causes a loss of sensitive data of that application. So while developing an application it is mandatory to ensure that it is not vulnerable to any kind of web based attacks. Now a days, organizations have started developing their own applications which contains sensitive data [1]. For protecting this kind of sensitive information source code review should be done. Source code review is a process to identify and remove security risks of the applications which contains sensitive information related to company. When developer develops an application, who don't have knowledge of how to code securely then this will create a big security hole for the application and attacker can taker advantage of this security weakness.

For the code review process there are multiple ways, SAST and DAST are one of them. SAST (Static Application Security Testing),which is also known as white-box testing methodology where application is tested from inside to outside by reviewing the source code which may shows that whether there is presence of security vulnerability or not.DAST (Dynamic Application Security Testing),which is also known as black-box security testing methodology where an application is tested from the outside to inside by reviewing an application while it is in it's running state and try to perform some attacks as attacker perform [2]. As shown in fig.1 there are various stages of web application security testing methodology like information gathering, planning analysis, Vulnerability Detection, Penetration Testing And Reporting.

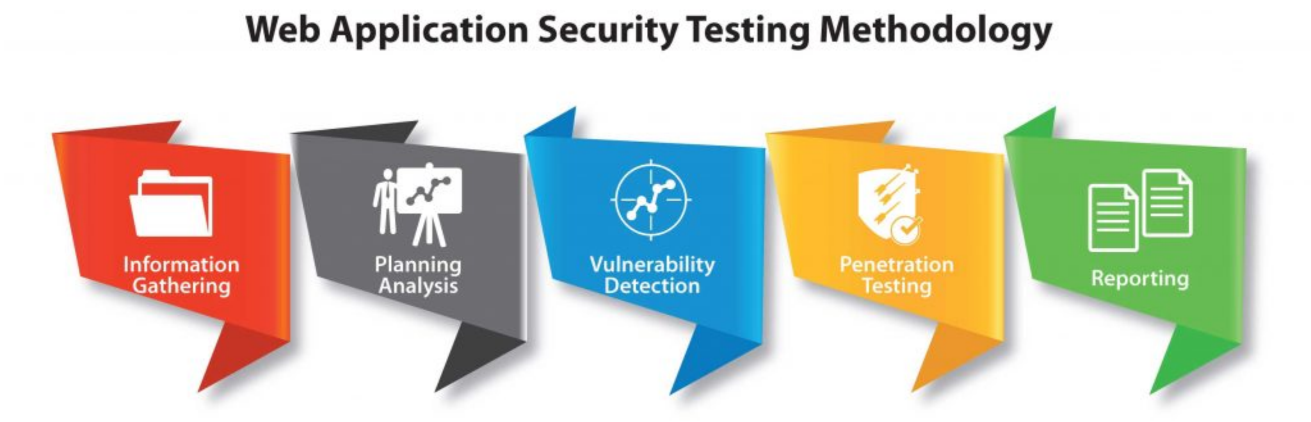
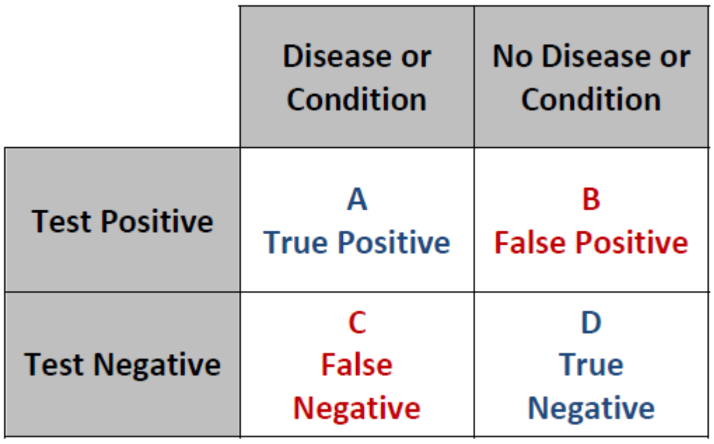


Figure 1.1: Application Security Testing Methodology

1.1.1 Identification Categories

When application goes through various process of code review, there are some tools and also manually vulnerabilities are found. When tool raise an alert for the vulnerability there are chances that actually vulnerability is not present. Here are some of those scenarios.

- **True Positive:** Which means Condition is detected actually when condition is present. Basically it will classify intrusion in the system as an intrusion.
- **True Negative:** This means it does not detect the condition when the condition is absent. So it will correctly classify the normal data as an normal.
- **False Positives:** Which means Condition is detected actually when condition is absent. Basically it will incorrectly classify normal data as intrusion in the system.



	Disease or Condition	No Disease or Condition
Test Positive	A True Positive	B False Positive
Test Negative	C False Negative	D True Negative

Figure 1.2: Test Statistics

- **False Negatives:** This means it does not detect the condition when the condition is present. So it will incorrectly classify the intrusion as an normal[3].

1.2 Identity and Access Management

It empowers the correct people to get to the correct assets at the correct circumstances and for the correct reasons.Organizations keep on adding service for both inside clients and their customers.Numerous services may need identity management to provides such services to their clients.Identity alliance involves at least one frameworks that unify client get to and enable clients to sign in view of authenticating against one of the framework partaking in that federation. The SAML (Security Assertion Markup Language) is a globally accepted XML based language for getting security by protecting integrity and authentication of SAML assertions.

1.2.1 Authentication And Authorization

Authentication: Server uses authentication to check that who is using their services,etc. In that user have to show it's identity to server or client. It does not specifies which type of tasks a user can perform or which kind of files a user can see.Authentication only specifies and check who is the person or system accessing service provided by them. List of possible issues related to authentication

- Disclosing sensitive information
- No user lock down policy in place

- Authentication Bypass
- Single Sign on can be abused

Figure 1.3: Identity Management

Authorization: It is a process by which server checks whether client has any kind of permission to use resources or not and he can access some files or not. So in other words it is a function which specifies who have access rights to which resources and that is related to information security. Here are some possible weak area in application for the authorization.

- Insecure Session tokens
- Authorization mechanism is weak
- Bypass the single place authorization
- Bypass the role-based weakness and exploitation

1.3 Motivation

When developers develop an application sometimes he has not taken care of secure coding practices. So there are many vulnerabilities in the applications which needs to be fixed before it goes for the users to use it. If such things is not taken care then outside threat will always be their because many application may contain sensitive data related to it's organization or their user which is not supposed to be exposed.For example, if developer have not used parameterized query for the SQL statements than there will always threat of attacks like SQL injection. There some other kind of attacks like Cross Site Scripting(XSS),Heart-bleed,XML signature wrapping attacks,Zero days, Broken authentication,etc can be possible. And when application goes through the tools for code review some times even tools can't detect some of those vulnerabilities.So there should be some proper system which can detect most of the vulnerabilities in the application, reduce manual task of the AppSec team, and also able to do pen-testing on the application before it goes live.

1.4 Objective

The objective of the project is to design a system that will reduce the huge manual work of AppSec team for application code review. By developing this kind of system it will be easy for the team to go through the code and find vulnerabilities. This system will reduce the rate of false negatives up to certain extent and also reduce some known attacks. And also it will automate the Application security portal, which also allows the external user to report the suspicious behaviour of an application. Which will add the values for the organization because as the number of attacks/vulnerabilities decreases value/reputation of that organization increases gradually.

1.5 Scope of Work

For the code review process every application needs to be register in the Application Security Portal. In review process there are four steps: 1. Register: where the owner of the application have to register their application on the AppSec portal. 2. Plan: where they need to answer some of the questions for the security start process. 3. Scan: where all the review process has to be done, one report is generated and 4. Remediate: where the security vulnerabilities get solved and the app owner is notified if their application have any vulnerabilities. To reduce some manual work which appsec team is doing currently,we are developing one more portal for Appsec where users can report vulnerabilities in the applications which they are using related to our organization. This portal is developed using Visual Studio 2017 and we are planning to automate this portal using selenium framework. [4]

1.6 Tools and Technology

- Visual Studio 2017 - Provides Integrated development environment, which allows you to develop web sites, web services, mobile apps, etc.
- NetSparker - Web application security Scanner.
- Selenium - Software testing framework for web application.
- Windows Services

Chapter 2

Literature Survey

2.1 Code Review

Main task is to review the applications and provide the security stars.App security is like if your code has not maintain the proper standards or some basic rules etc,then there are chances of attacks on that app.So code scan review is important.Coding standard should be maintain if it is not then there vulnerabilities or open issues.Review process has divided into three parts: **Static, Dynamic and Manual**.Following image shows the process for the source code review:

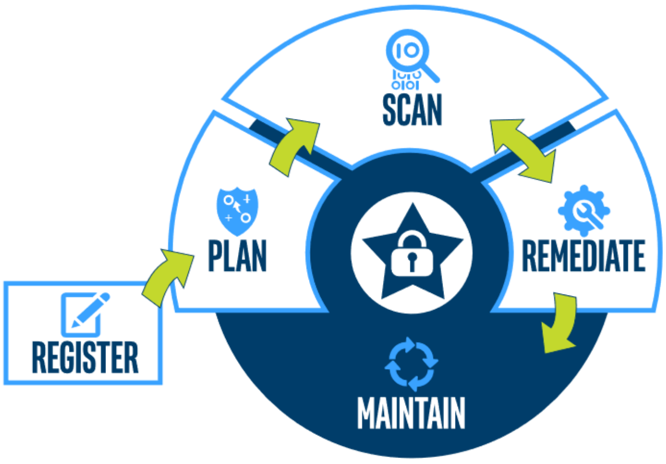


Figure 2.1: Code Review Process

Code scan review process is divided into three steps. Register, Plan and scan.After all this done AppSec team will start remediation process.[5]

- **Register:** This is the initial step for Code Review process. In this step Owner of the application has to register his app with the AppSec Portal. That AppOwner has to follow certain steps and after that they are provided with unique AppId.

- **Plan:** In this step, AppOwner have to answer some questions related to their application. These questions are related to the some of the security concerns which an AppOwner should answer.Each and evry application which is registered with the AppSec portal have to have PKI certificate only after that, an application can be registered with the portal. The answers are recorded and it is transfer to the reviewer for the review along with the for the review.

- **Scan:** This step means after doing all this stuff by AppOwner the code of the application will be given to the Intel Standard Tool for code scan process. This code is provided by the ApOwner only. This scan can be done manually and also by the tool. If some of the language is not supported by the tool then AppSec team have to manually review that application for the vulnerability check.After doing this one report is generated which includes detailed description about vulnerabilities. Then AppSec team will review this After scan one report has been generated by the tool with the detailed description about the vulnerabilities.Using this report remediation process is done. [6]

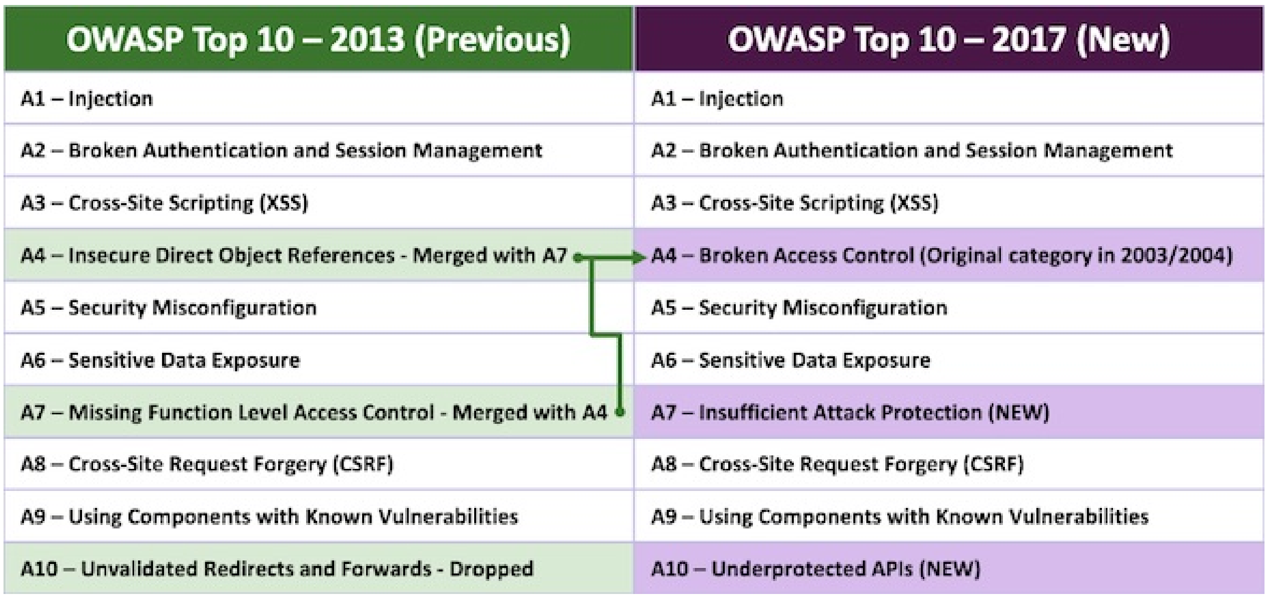
2.2 Common vulnerabilities during Code Review

As per the OWASP top 10 standards their are list of vulnerabilities which is very common in the application. And those vulnerabilities needs to be taken care for the security purpose.Attackers may use different paths through the application which will results in to harm the company.According to OWASP top 10 Application is exposed means vulnerable to attacker when:

- The data which is supplied by the user is not validated properly,if that data is not filtered and sanitized properly.

- Non-parameterized queries are used.

- Coding is not done as per the secure coding standards.



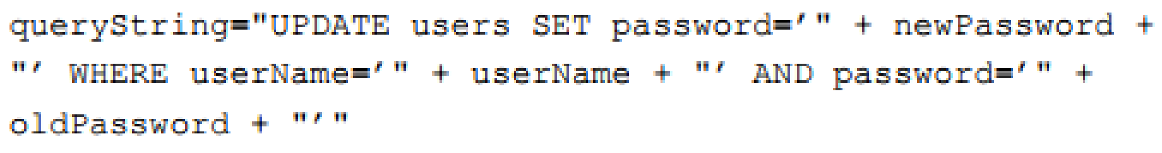
OWASP Top 10 – 2013 (Previous)	OWASP Top 10 – 2017 (New)
A1 – Injection	A1 – Injection
A2 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References - Merged with A7	A4 – Broken Access Control (Original category in 2003/2004)
A5 – Security Misconfiguration	A5 – Security Misconfiguration
A6 – Sensitive Data Exposure	A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control - Merged with A4	A7 – Insufficient Attack Protection (NEW)
A8 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Components with Known Vulnerabilities	A9 – Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards - Dropped	A10 – Underprotected APIs (NEW)

Figure 2.2: OWASP Top 10

2.2.1 SQL Injection

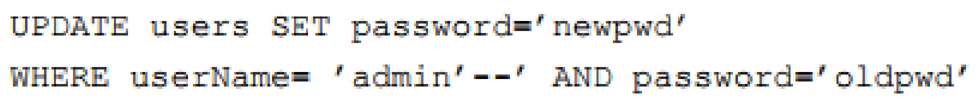
SQL Injection is a vulnerability which is described as one of the most serious vulnerabilities. If there is SQL Injection Vulnerability in the web application then it may allow an outsider means attacker to get into the application and allows the attacker to get complete access to the database of the application. This database contains some sensitive information related to organization.

SQL Injection occurs when an attacker tries to change the effect of it by simply giving the SQL keywords inside the query.This injection can be done in multiple ways like Injection using cookies,user inputs,server variables. Here is one example of this attack.Let's say there is one web application which have command:



```
queryString="UPDATE users SET password='" + newPassword +
"' WHERE userName='" + userName + "' AND password='" +
oldPassword + "'"
```

Figure 2.3: SQLInjection Vulnerability



```
UPDATE users SET password='newpwd'
WHERE userName= 'admin'--' AND password='oldpwd'
```

Figure 2.4: SQLInjection Vector Input

Here In fig. 2.3 user is modifying passwords by,first he or shes checks that current password is known and then will change the passwords if the previous check is successful.Then the query string is sent to the database. SQLInjection is all about compromising your database with the help of these kinds of Injection Vectors. Because "–" is the SQL Injection vector in fig. 2.4, everything which is after that vector is considered as comment in the database.

2.2.2 Broken Authentication and Session Management

The commonness of broken authentication is across the board because of the plan and usage of most personality and access controls. All the stateful applications have Session management and its s very important for authentication. There are many automated tools which is used by the attackers to catch the broken authentication with the help of list of passwords and attacks like dictionary attacks. There are some things which shold be taken care like secure channel should be used to transport cookies, if it is possible then all traffic should be conducted over the HTTPS, password should be complex and strong.

2.2.3 Cross Site Scripting (XSS)

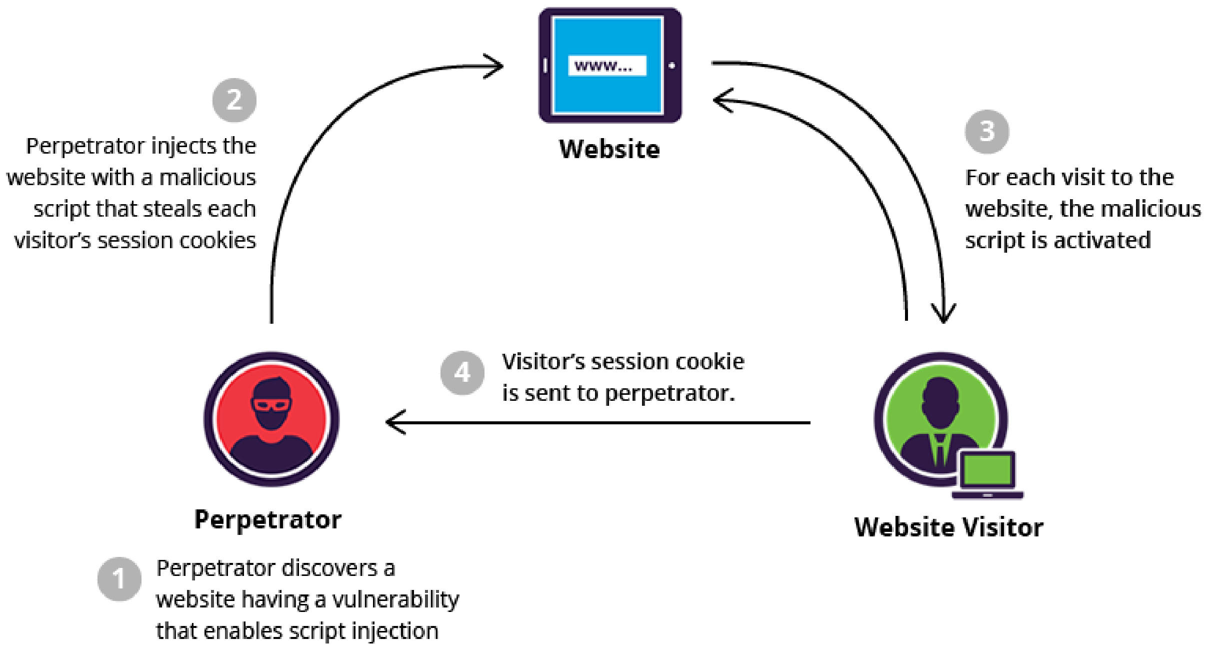

www...
Website
2
Perpetrator injects the website with a malicious script that steals each visitor's session cookies
3
For each visit to the website, the malicious script is activated
4
Visitor's session cookie is sent to perpetrator.
Perpetrator
Website Visitor
1
Perpetrator discovers a website having a vulnerability that enables script injection

Figure 2.5: Cross Site Scripting attack

This type of attack uses scripting codes in the web application in which those codes are added in to the output of that application then it will be sent directly to the user's browser who is accessing that web application.It happens when progressively created site pages show input that isn't appropriately approved.This permits an attacker to implant pernicious JavaScript code into the produced page and execute the content on the machine of any client that perspectives that site. There are three types of XSS:Reflected XSS, Stored XSS and DOM XSS.

2.3 Problem with existing tool

The tool which is used for code review is capable of searching most of the vulnerabilities. But there are some cases where tool is not able to identify some of the attacks. And also some languages are not supported by the existing tool[7]. So there are chances of attacks like identity theft, Zero days, Heart bled, etc. So for all this things manual review is needed.

2.3.1 Advantages

- Find weaknesses at the exact location in the code.
- If automated tools are used then it is relatively fast.
- Used only by the trained developer who can fully understand the code written.

2.3.2 Disadvantages

- If manually conducted then it is time consuming.
- Tool do not support all the programming languages.
- Produce False Positives and False Negatives.
- tool is only good as the rules which are statically defined to find vulnerabilities.

Chapter 3

SAST And DAST

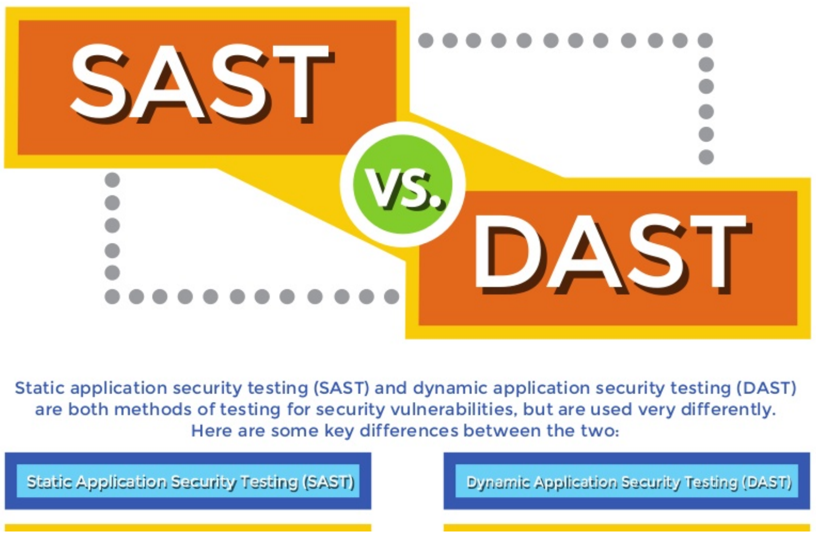

SAST
VS.
DAST
Static application security testing (SAST) and dynamic application security testing (DAST) are both methods of testing for security vulnerabilities, but are used very differently. Here are some key differences between the two:
Static Application Security Testing (SAST)
Dynamic Application Security Testing (DAST)

Figure 3.1: SAST Process

3.1 What is SAST(Static Application Security Testing)?

Application security have been raised exponentially due to increasing huge amount of cybercrime and some malicious activities which made each and every organization to think over their sensitive data which is under their applications. SAST stands for Static Application Security Testing. We can also refereed to as 'White-Box' testing. This is the static security analysis method that can statically scan the source code and look for the vulnerabilities. There is no need to compile the code. SAST should be the mandatory requirements for the organizations because are are large amount of attacks which is taking place at application layer. [8]

3.2 How does it work?

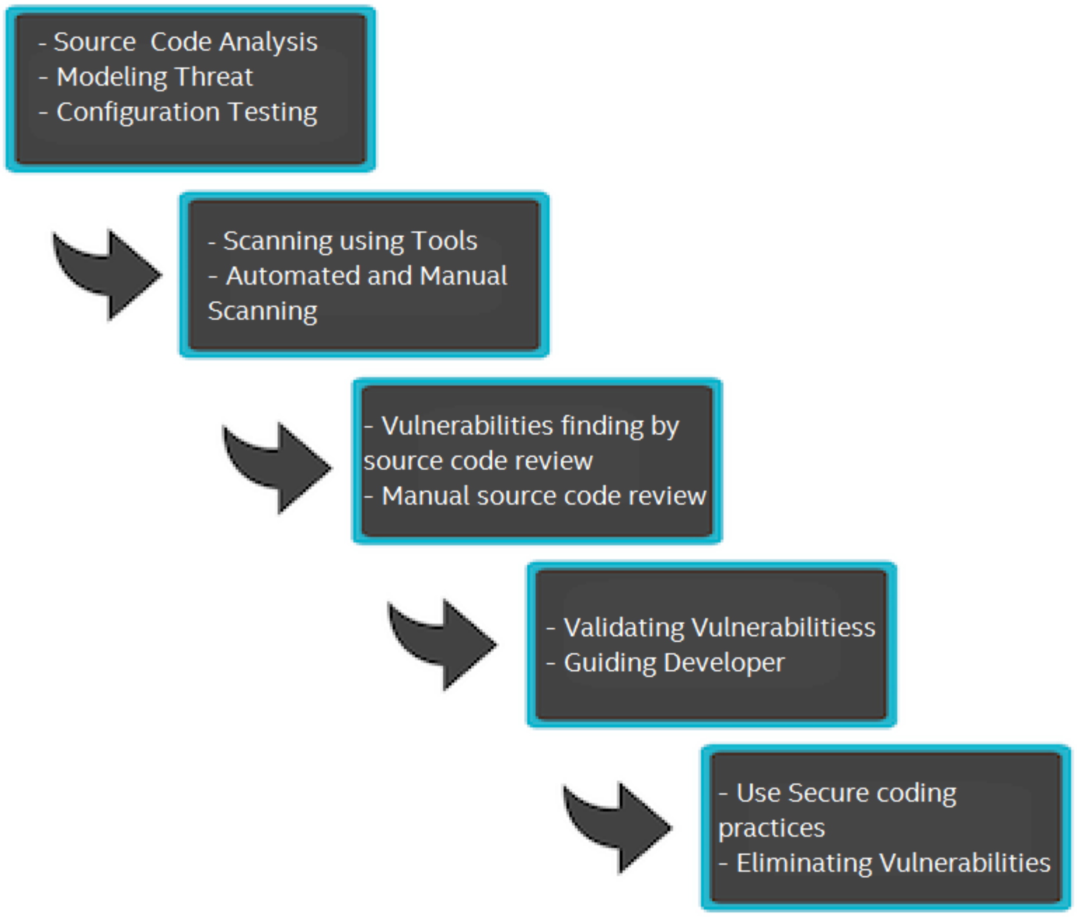

- Source Code Analysis
- Modeling Threat
- Configuration Testing
- Scanning using Tools
- Automated and Manual Scanning
- Vulnerabilities finding by source code review
- Manual source code review
- Validating Vulnerabilitiess
- Guiding Developer
- Use Secure coding practices
- Eliminating Vulnerabilities

Figure 3.2: SAST Process

Solutions which comes of out SAST process are directly integrated to the development environment which allows the developers to check their code continuously.By doing this process vulnerabilities in the source code can be easily avoided. For now our main focus is on SAST code scanning. Specific tool is used for SAST process which takes the data which is source code uploaded by the app owner and scan the whole code line by line, after scanning it will generate one report which shows the list of possible vulnerabilities in the code. This report includes all the information about code which is vulnerable for that application.

Then Security experts will help the developer to fix the issues and again the whole process take plan unless and until all the source code of the application is fully secure.

There some cases where security experts have to do manual scanning for finding bug's in the application's source code also.

There are three different types of vulnerabilities level generated using the tool which we call it as **Criticality**. On the basis of criticality there are three levels, which are High, Medium and low.

- **High:** This is the first type of criticality. High criticality means that action should be taken fast to resolve this type of serious issues. High means that bug needs to be solved first as it has high security alert.
- **Medium:** This is the second level bug that needs to be fixed after high criticality bugs are resolved. This type of bugs have lesser security risks compare to high criticality.
- **Low:** This vulnerability does not really required to be solved as this has nothing to do with the attacks. Its warning kind of thing if developer wants to solve it then they can solve or request team to solve it but its not necessary as such.

3.3 What are the benefits?

- This process is able to find out vulnerability at the proper location of the source code.
- If automated tools are being used then this process is much faster.
- Before the application is being deployed this process is able to expose vulnerabilities in the source code with the help of some tools.
- This tools are checking the source code and some times binaries line by line also and after doing that they are able to find the flaws and show the results to the security expert, which will help the developer to use some secure coding practices and make their application secure.
- By detecting the security issues at very early stages of release of that application, high criticality issues can be easily resolved.

3.4 What is DAST (Dynamic Application Security Testing)?

DAST performs the analysis test on the applications which are in their running state.This type of security testing is very help full when applications are live and security experts wants to check whether there are any security loopholes or not. It is also famous as Black Box Testing just because it does not uses the flaw of SAST testing flaw like viewing the source code of the application, what is does is, this process uses the same flaw which attacker uses with the aim of get into the applications by finding security bridges. This requires tools to perform analysis.There many tools available in the market for DAST process but it depends on the organization which tool they should use according to their needs.

DAST is always in search for different types of security issues like i/o validation problem, which can result in to XSS attack or SQLi. Basically this process gives the full picture of security loopholes in the respective application[9]. DAST misses the mark once more because of its inherited attributes, which empower it to begin working simply after the build is finished.DAST tools also offers the risk analysis facility during the remediation period which means developers have no idea about the exact location of vulnerability. DAST tools can easily find exploitable vulnerabilities compare to SAST.

3.5 What are the benefits?

- DAST analysis is able to track data in the real time and security index is capable of all websites.
- DAST can able to find vulnerabilities which SAST can't find.
- It provide support for almost all the languages like PHP, .NET, JAVA, etc so that it can run all the test cases.
- Analysis can track data in real time and security index is capable of all your websites.
- Fix all issues more rapidly with definite remediation data.

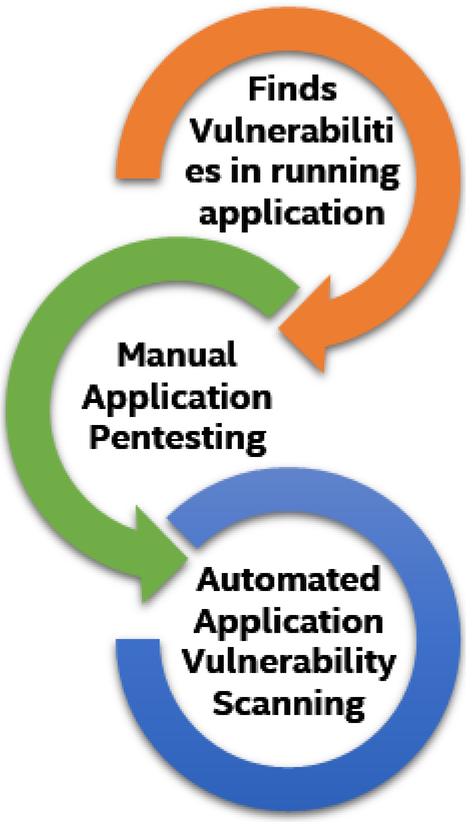

Finds Vulnerabiliti es in running application
Manual Application Pentesting
Automated Application Vulnerability Scanning

Figure 3.3: DAST Process

- False negatives can be eliminated as threat research team is focusing on different new bugs and this will be helpful for the remediation process.
- With the unlimited licence access Threat security team is always there to answer user's queries and help user to solve the vulnerabilities in can he is not able to solve it.

3.6 Modeling Web Applications Vulnerabilities

The essential goals of data security frameworks are to ensure privacy, trustworthiness, and accessibility. From our examples it is obvious that compromising trustworthiness can also cause compromise in privacy and accessibility[10]. The following figure states that how this three main categories can compromise.

Because of this unauthorized data are used to construct authorized output without any secure coding practices or sanitation. There is a clear need for some mechanism that

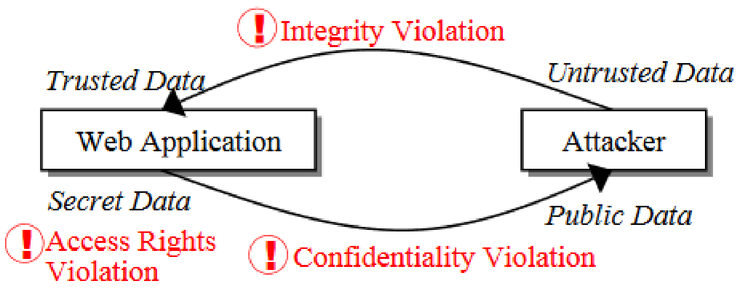


Figure 3.4: Application Vulnerabilities Modeling

specifies very good secure coding practices.

Chapter 4

System Implementation

4.1 Requirement

While developing an application there are vast amount of issues related to security. For securing application, it goes through various process of code review. At the time of identifying the bug during the code review process some of the tools which are used for that process can easily detect security bridges but this is not true every time. Some areas in the application where tools doesn't reach create major problem when application goes live.So this types of bugs which tools are not able to find should be fixed as soon as possible. For now we are creating and automating one portal where security researchers report the bugs which they have discovered and then security experts will help the app owners to fix the issues.

4.2 Existing Tools

For the application security code review process tools which are used for scanning the source code are very useful. They are able to generate report after scanning the full source code line by line. With the help of this report security experts resolve the bugs and help the app owner to fix the issues and send them report. This tools are company certified tool. This tools have all the capabilities of finding the vulnerability line by line. And they can also generate the report as per the requirement. So this will be very helpful for the security experts as well as the app owner to fix the issues in their application.

4.3 Solution

The solution for current challenges are have to be implemented. So we are developing the portal for the application security team, where external researchers will report the vulnerabilities which they have found during their research, with the full detail of him or her and vulnerabilities. We are planning to provide them many other options like they can also upload the files if they want to show where the exact problem in the application is and how they have found the bug. After the external researchers have submitted all the vulnerabilities on the portal, this all comes to the security expert team. They have to decide and check whether all this vulnerabilities which external security researchers have reported are valid or not. They also do penetration testing to find exact path for the issues. After the vulnerabilities are marked as fix, security experts send an email to the relevant app owner regarding this issues and help them to remediate this issues. For the future work, planning to automate this process so there are less chances of attacks and also planning to implement secure coding mechanism that we have used to create the system to identify as much attacks as possible.

Chapter 5

Working of the system

5.1 Implementation

5.1.1 The Code Review Process

As discussed earlier the process of identifying vulnerabilities from the code review, and with the help of report generated by the tool, security experts mark the bugs on the basis of the bugs criticality. And as per the criticality that bug the marked as per the risks levels like, High, Medium and Low.After all this process completed the report with the valid issues is sent to the application owner, so that they can fix the issues. Security researchers also helps the app owner to fix this issues. Developer can use this findings for the improvement of his application by including some secure code mechanisms. To some extent this process does not find each and every bugs for the particular application. So there are some areas which still have some issues which my cause a big damage if they are not taken care properly. Source code review is not a golden bullet but it is very strong in the whole mitigation process.

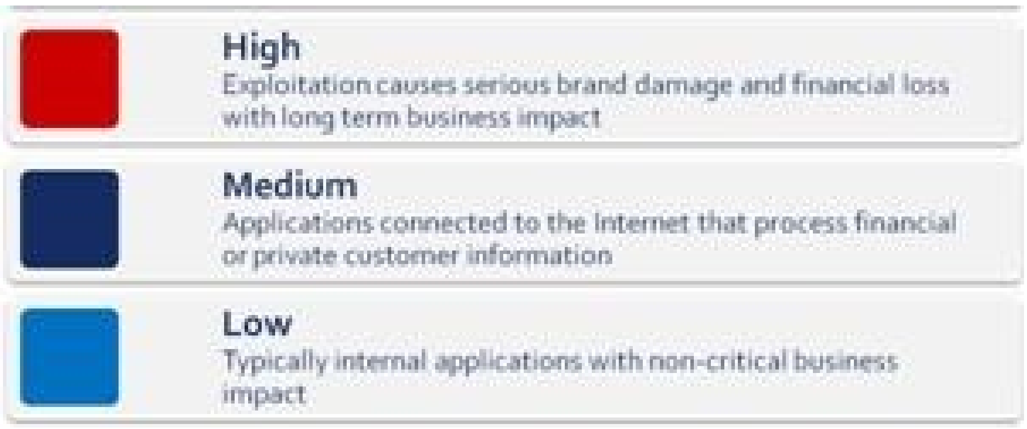
5.1.2 Review Summary

When an application owner develops an application, code review is must before their application goes live because developers may have not used the secure coding practices while developing an application. So first thing they do is, upload the zip format of the application code on the portal. This will directly come to the application security experts. This come to security plan on the portal. where app owner have to answer some questions related to their application and then SAST process starts.Application security researchers review the report generated by the tools and take some actions on it. App Owner has to solve all those issues that are there in the report. In case of anything that App Owner is unable to understand, he can contact to the security team to make him understand. Developer has to solve high category vulnerability first where ever it is in the report. High category application is the big threat to the system. After done with the high, medium one has to be solved.

So this process have main focus are like source code file upload in the zip format,SAST process, tools review the application, report generated by the tool, manual review of the application, helping the app owner to fix the issues.Each member of the application security team assigned with application in the queue for the code review process.

5.1.3 finding Summary

Suppose I have reviewed one application and at the end of the review report have the graphical representation which shows the criticality level, like High, Medium,Low. Which shown in the following figure.



🟥 **High**
Exploitation causes serious brand damage and financial loss with long term business impact

🟦 **Medium**
Applications connected to the Internet that process financial or private customer information

🟦 **Low**
Typically internal applications with non-critical business impact

Figure 5.1: Criticality Level

As per the order of criticality issues which have high criticality need to be fixed first after that medium and then low criticality issues have to be solved. High criticality issues are the one which are very serious, if they are not taken care seriously then big damage to the company can happen.

The following figure shows the sample security code review report. From this we can come to know that which vulnerability is to be solved first. All the vulnerability names are given and which vulnerability falls under which category are also given. Based on that developer has to solve all those.

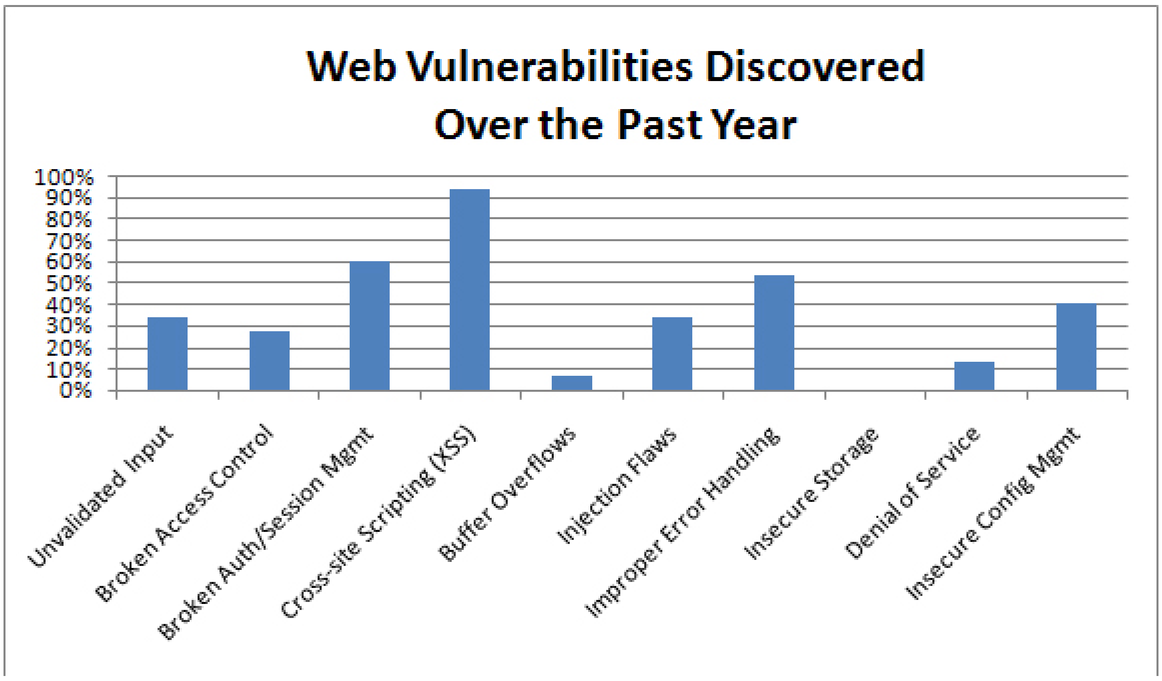
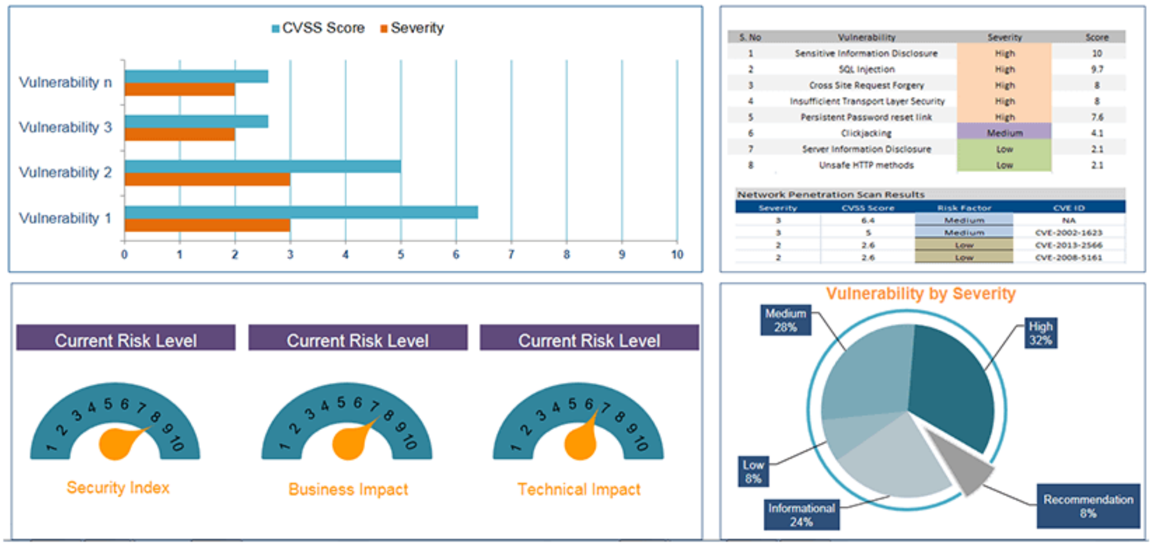


Figure 5.2: Findings by Category

5.1.4 Error Findings

From the report generated by the tool, which shows where the issues are in the source code and that will help the application security experts to guide the developers and help them to fix the issues in their application.Figure 6.3 is the of how the security dash board looks like when the vulnerability detected and where it is, this is not the exact figure but similar to this is generated. Other figure shows how different types of vulnerabilities.



S. No	Vulnerability	Severity	Score
1	Sensitive Information Disclosure	High	10
2	SQL Injection	High	9.7
3	Cross Site Request Forgery	High	8
4	Insufficient Transport Layer Security	High	8
5	Persistent Password reset link	High	7.6
6	Clickjacking	Medium	4.1
7	Server Information Disclosure	Low	2.1
8	Unsafe HTTP methods	Low	2.1

Network Penetration Scan Results

Severity	CVSS Score	Risk Factor	CVE ID
3	6.4	Medium	NA
3	5	Medium	CVE-2002-1623
2	2.6	Low	CVE-2013-2566
2	2.6	Low	CVE-2008-5161

Figure 5.3: Security Testing dash board

5.2 System Modules

There are basically three modules in the current system which I have developed. Based on the requirements we have divided the modules into three parts. First module is for the external security researcher where they can report the bugs which they have found during their research work. Second module is for the internal team where they will get the lost of bugs reported by the external security researcher. And the last module is for sending an automated email to the respective App Owner of the application which is having the bug, to notify that this issue needs to be fixed as soon as possible.

Here is the detail description of each modules. There are also screen shots of the portal which is under development, where external researcher report vulnerabilities and application security experts review that vulnerabilities, give update regarding this to relevant application owner. Also planning to use some automation and source code mechanisms, in the system which I'm developing.

The system flow is as follows:

- **External Researcher Bug Reporting Page**
- **Thank You Message for External Researcher**
- **Application security Security Team Review Page**
- **Application Owner Details from API's**
- **Automatic Email generation to App Owner for reporting bugs**

Here is the diagram of the system flow.

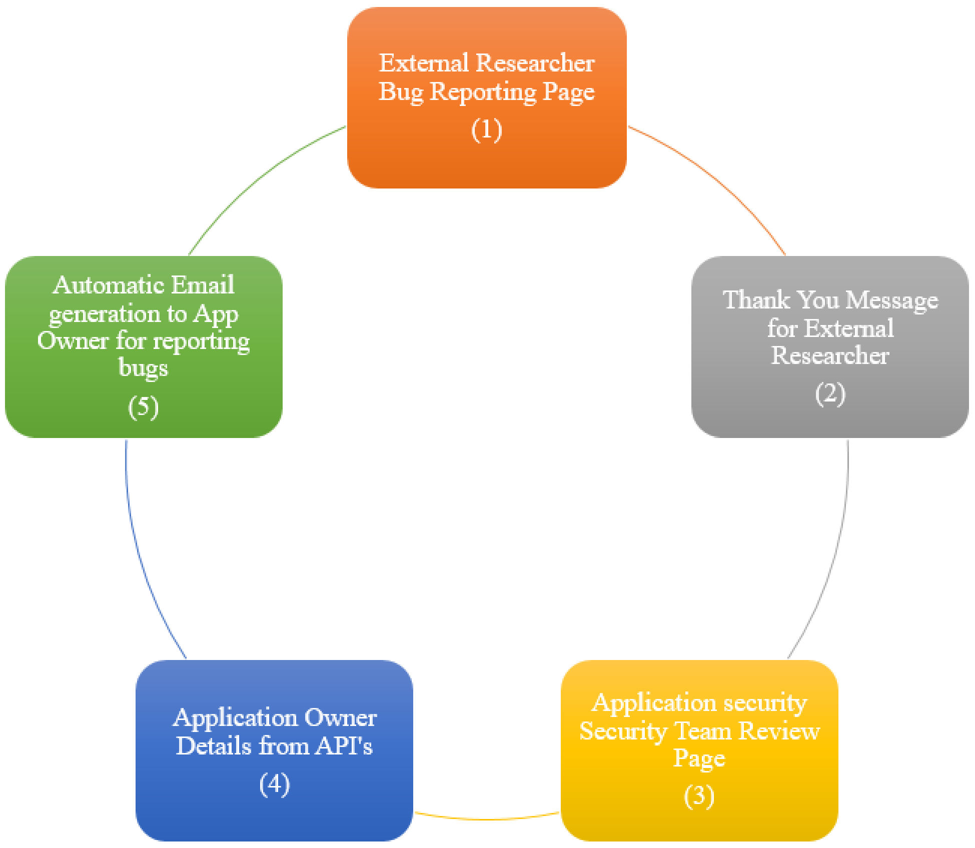
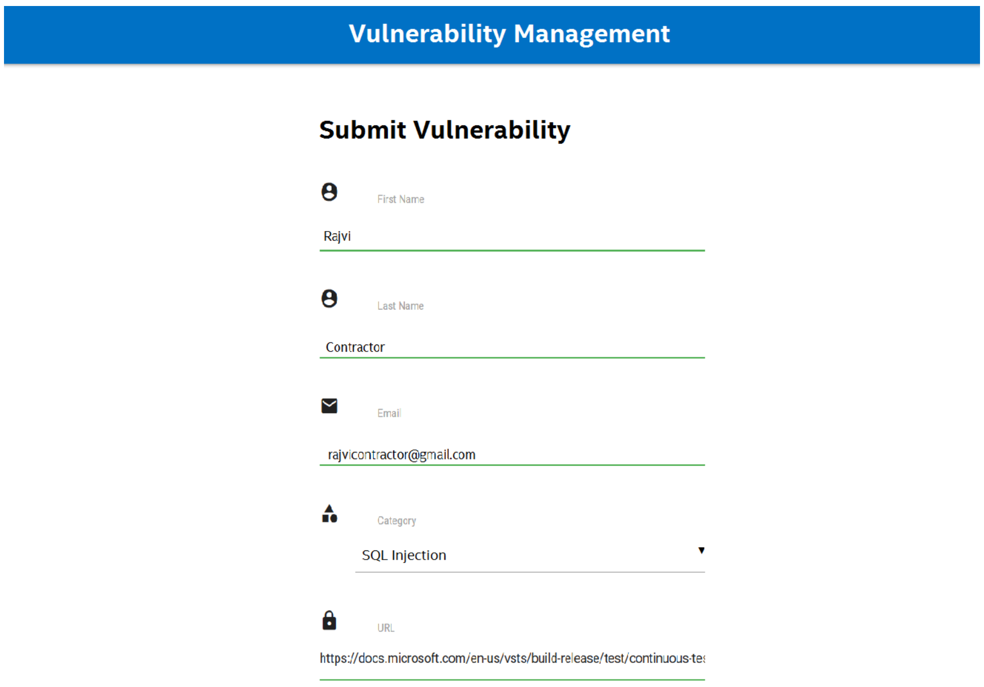

External Researcher Bug Reporting Page (1)
Thank You Message for External Researcher (2)
Application security Security Team Review Page (3)
Application Owner Details from API's (4)
Automatic Email generation to App Owner for reporting bugs (5)

Figure 5.4: User Entry

5.2.1 First Module

First module is for External security researcher. External security researcher finds the vulnerability during their research work on the external application. Then they have to report it to the organization's security research team. Earlier this process was manual where external security researcher have to send an mail to the org's security mailbox. But now they have to submit their findings on the portal which I have developed.

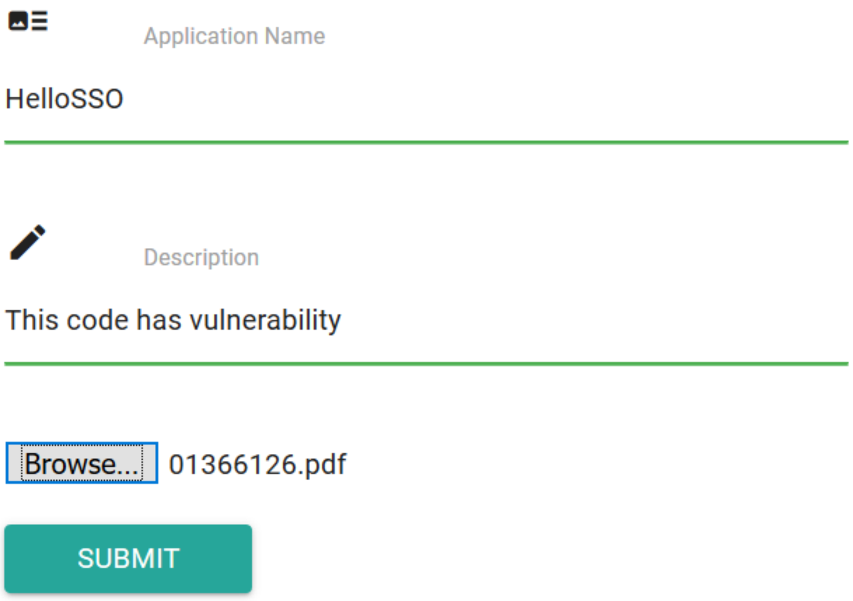
As shown in Fig. 5.5 and Fig. 5.6, External security researcher will fill up the form like his/her basic details, which type of vulnerability he/she has found, provide the URL of vulnerable page, Any extra comment they want to add or any recording/ proof of concept they can attach. As per Fig. 5.7 external security researcher receives thank you message and one navigation link that will redirect him to the same page on which they have reported bugs, where then can report more bugs.


Vulnerability Management

Submit Vulnerability

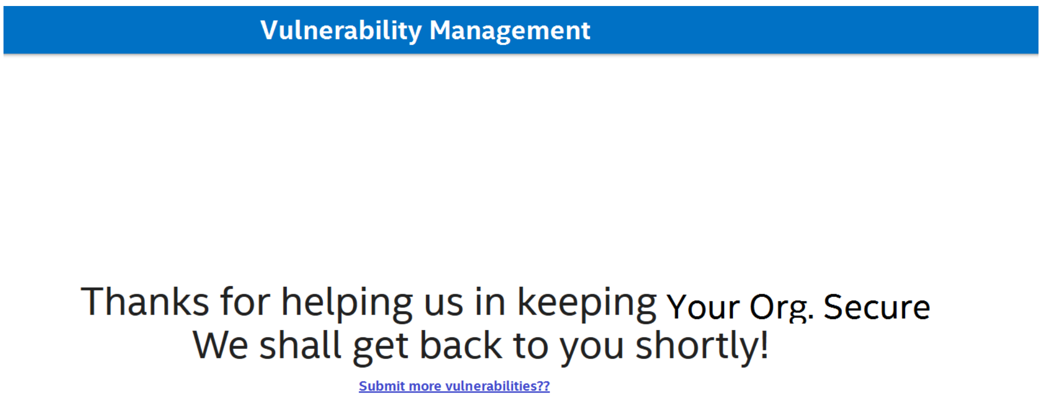
First Name	Rajvi
Last Name	Contractor
Email	rajvicontractor@gmail.com
Category	SQL Injection
URL	https://docs.microsoft.com/en-us/vsts/build-release/test/continuous-te

Figure 5.5: User Entry



Application Name	HelloSSO
Description	This code has vulnerability

Figure 5.6: Thank You for Vulnerability reporting


Vulnerability Management

Thanks for helping us in keeping Your Org. Secure
We shall get back to you shortly!

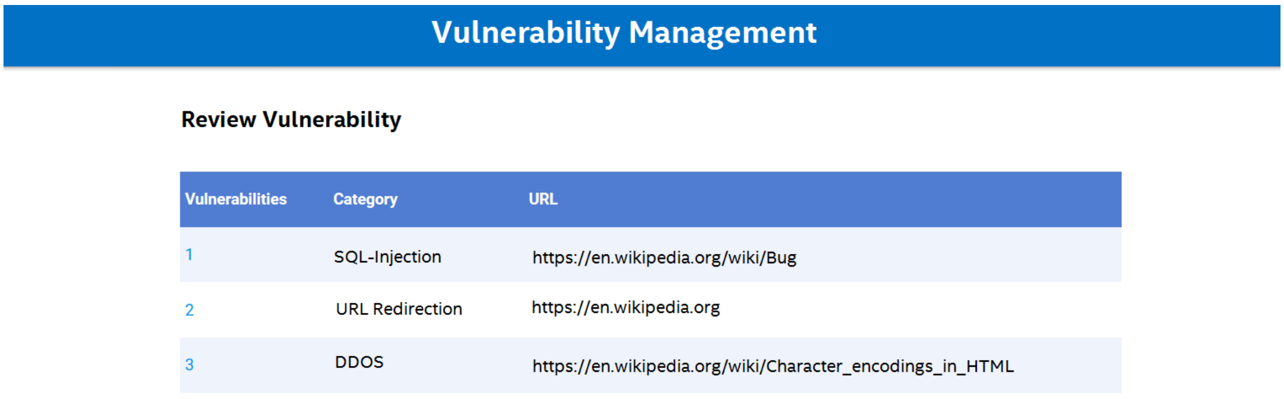
Figure 5.7: Thank You for Vulnerability reporting

5.2.2 Second Module

The second is for Internal security team which are us. Earlier we received bugs reported by external security researcher through email, but now I have created a portal for the internal security team where they can see list of bugs reported by external security researchers as shown in Fig. 5.8 . Here Internal security team can select the vulnerability which he/she want to review as per the vulnerability name.

After selecting the vulnerability they will be redirected t the next page where they can see the web form containing the data of vulnerability they have selected. This data we are fetching it from the database. Here there are some more fields like IAP number, infosec comments, bug is valid or not this all things are filled by Internal security team. And basis on the vulnerability, it's criticality will be decided and according to the criticality we have specified the number of day to close the issue. After filling the data reviewer will validate the issue, means reproduce it. If the issue is valid then reviewer will click on accept button otherwise reviewer will click on the reject button to ignore the issue because it's not valid or that issue will be already resolved.

Now if the accept button is clicked then list of only valid issues will go to the higher level internal researcher team. And also there is one auto-generated mail which is send to the respective external security researcher for appreciating their efforts which is shown in Fig. 5.9 and Fig. 5.10 . Now as shown in Fig. 5.11 they will be redirected to the page where their review is done and one link is given through which they can navigate to the page where they get list of vulnerabilities to review more. This is where internal security team work is done and issue is pass it to Higher level team to take actions on it.

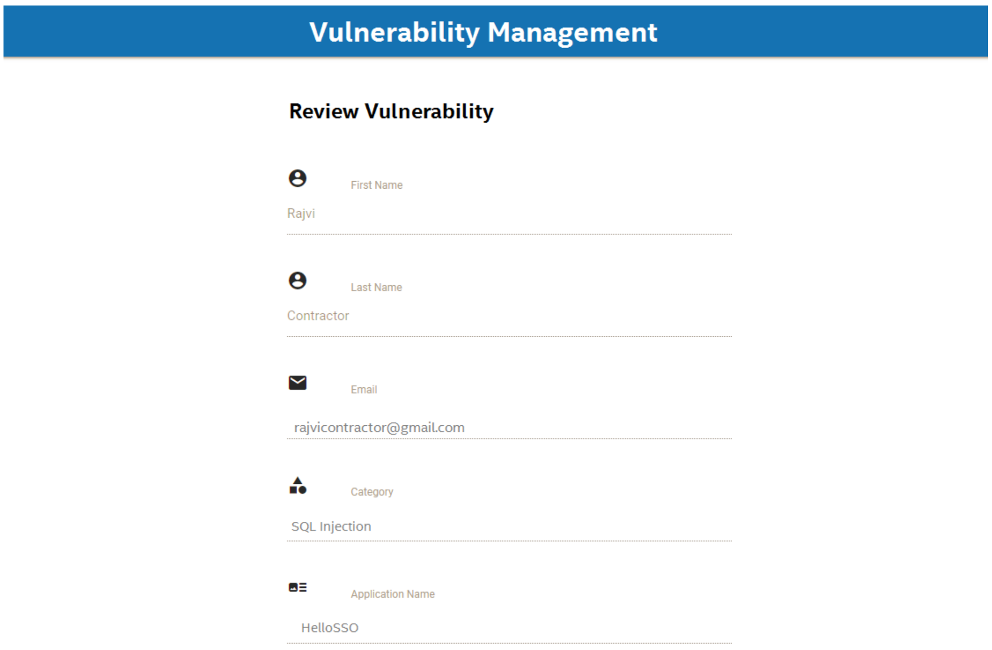


Vulnerability Management

Review Vulnerability

Vulnerabilities	Category	URL
1	SQL-Injection	https://en.wikipedia.org/wiki/Bug
2	URL Redirection	https://en.wikipedia.org
3	DDOS	https://en.wikipedia.org/wiki/Character_encodings_in_HTML

Figure 5.8: List of Vulnerabilities reported by the External security researcher

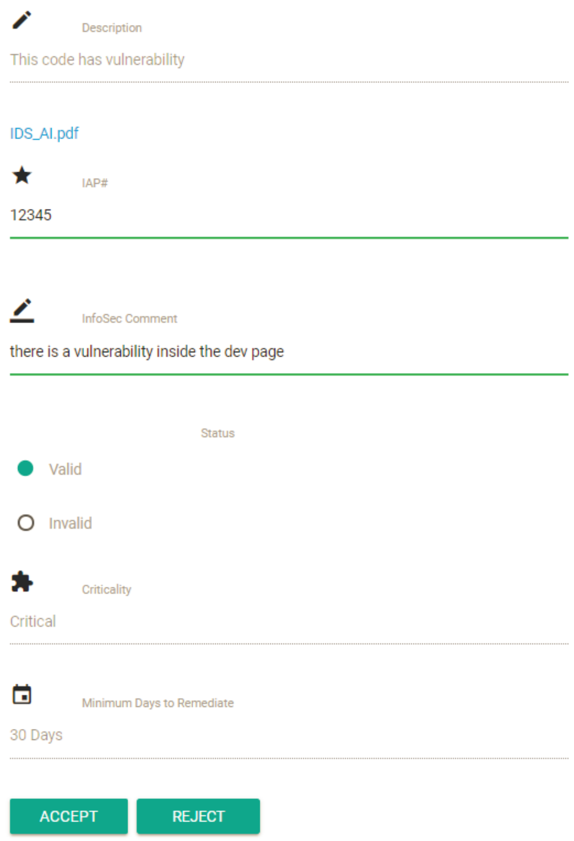


Vulnerability Management

Review Vulnerability

First Name	Rajvi
Last Name	Contractor
Email	rajvicontractor@gmail.com
Category	SQL Injection
Application Name	HelloSSO

Figure 5.9: Internal security team will validate the bug and add some data


Description
This code has vulnerability

IDS_AI.pdf

IAP#
12345

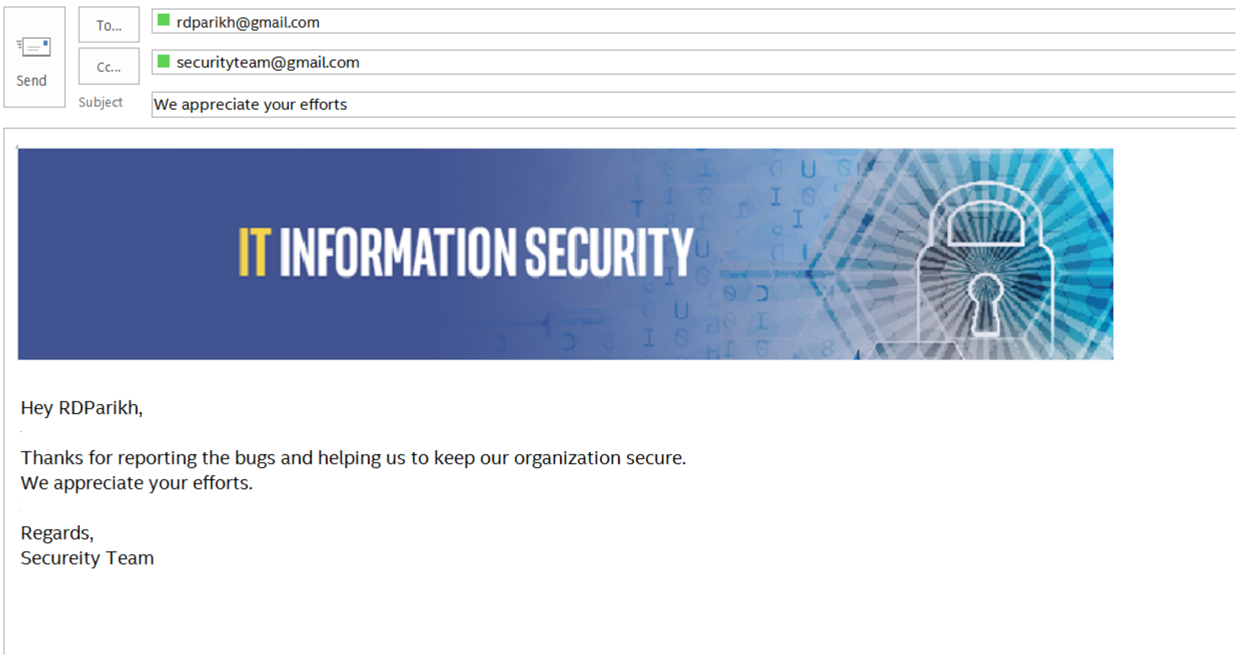
InfoSec Comment
there is a vulnerability inside the dev page

Status
● Valid
○ Invalid

Criticality
Critical

Minimum Days to Remediate
30 Days

Figure 5.10: Internal security team validate the bug and pass it to higher level team



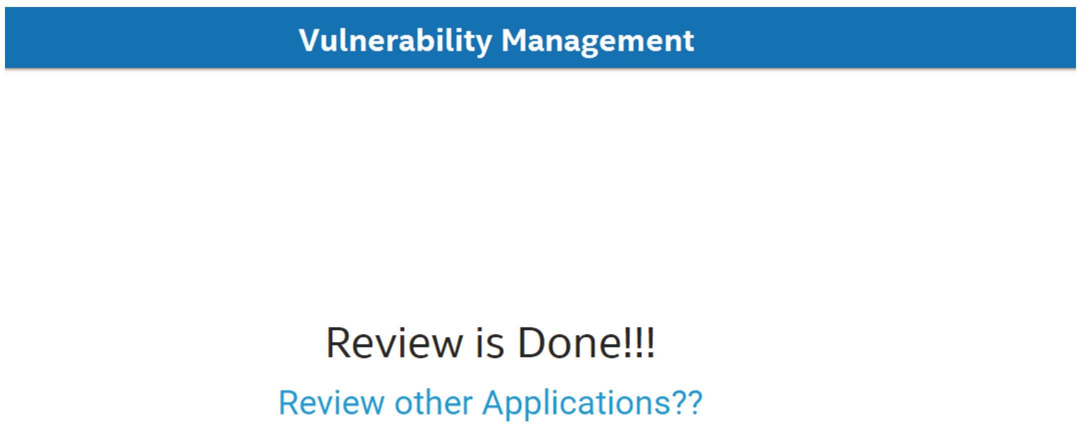
To... rdparikh@gmail.com
Cc... securityteam@gmail.com
Subject We appreciate your efforts

Hey RDParikh,

Thanks for reporting the bugs and helping us to keep our organization secure.
We appreciate your efforts.

Regards,
Secureity Team

Figure 5.11: Appreciation mail to the external security researcher


Vulnerability Management

Review is Done!!!

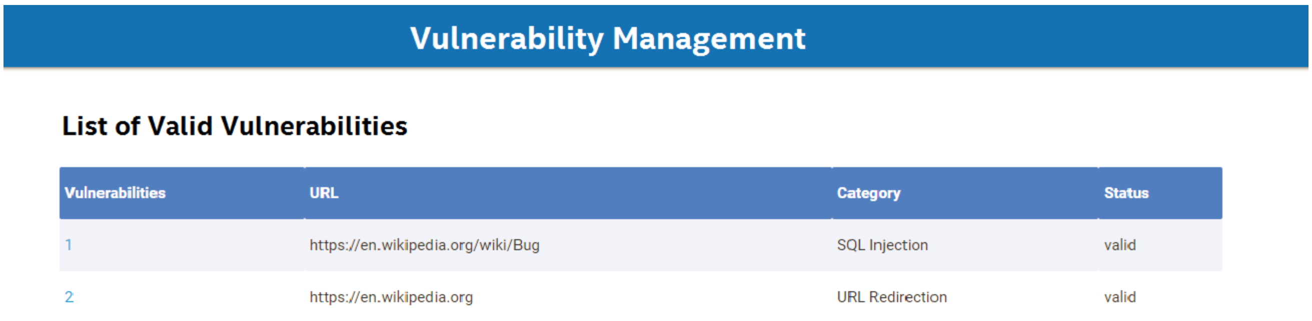
Review other Applications??

Figure 5.12: Review completed and Navigate to other page to review more Bugs

5.2.3 Third Module

In the third module all the vulnerabilities validated by the internal security team comes under higher level security team as Shown in Fig. 5.13. Now higher level security team will choose one of the vulnerabilities and it will fetch all the data according bugs selected. Now As the data shown in Fig. 5.14 and Fig. 5.15 can be changed by the higher level security team. Suppose they have found that the vulnerability is not SQL Injection but it's a Cross site scripting (XSS) then they have rights to change this fields.

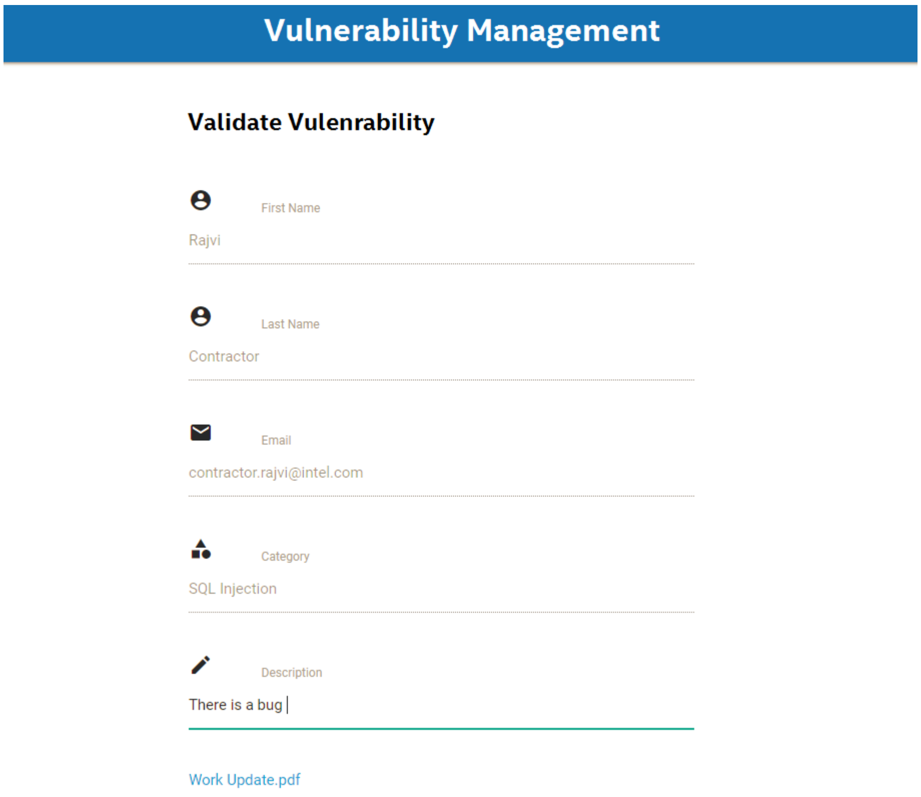
After the vulnerability is validated second time by the higher level security team then they will click on the accept button. At that time automated mail will be generated to the App Owner of the respective vulnerability as shown in Fig. 16. Now as shown in Fig. 5. 18 higher level security team can be navigated to review more bugs, but they can see only the bugs which are valid as shown in Fig.5.18.


Vulnerability Management

List of Valid Vulnerabilities

Vulnerabilities	URL	Category	Status
1	https://en.wikipedia.org/wiki/Bug	SQL Injection	valid
2	https://en.wikipedia.org	URL Redirection	valid

Figure 5.13: List of only valid Vulnerabilities comes from internal security researcher


Vulnerability Management

Validate Vulenrability

First Name
Rajvi

Last Name
Contractor

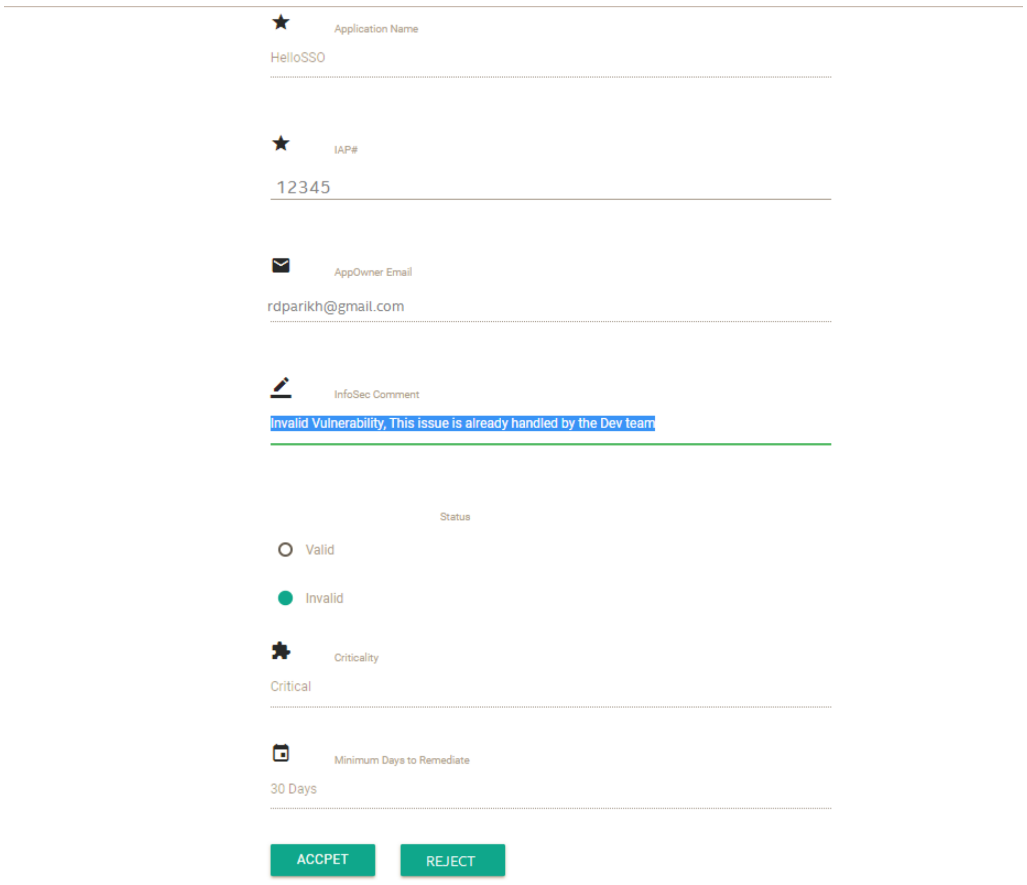
Email
contractor.rajvi@intel.com

Category
SQL Injection

Description
There is a bug

Work Update.pdf

Figure 5.14: Higher level team have rights to do any changes on the data


Application Name

HelloSSO

IAP#

12345

AppOwner Email

rdparikh@gmail.com

InfoSec Comment

Invalid Vulnerability, This issue is already handled by the Dev team

Status

○ Valid

● Invalid

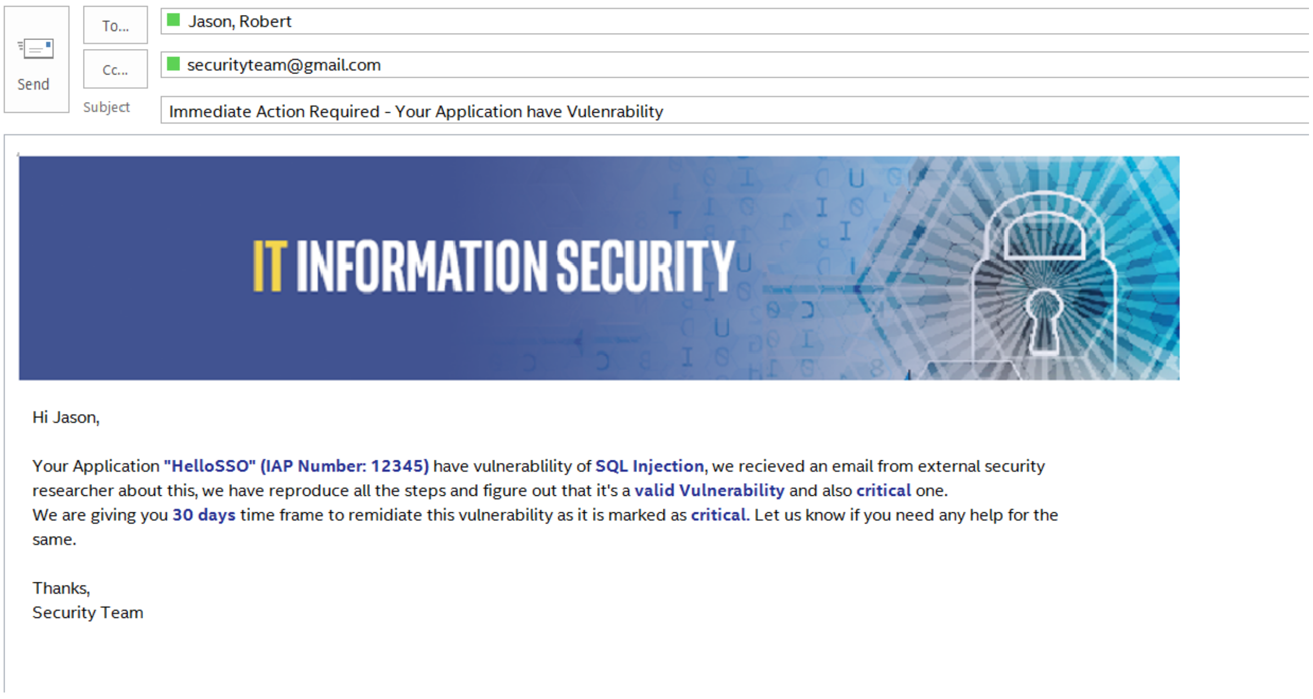
Criticality

Critical

Minimum Days to Remediate

30 Days

Figure 5.15: Higher level security team further validate the bug



To... Jason, Robert

Cc... securityteam@gmail.com

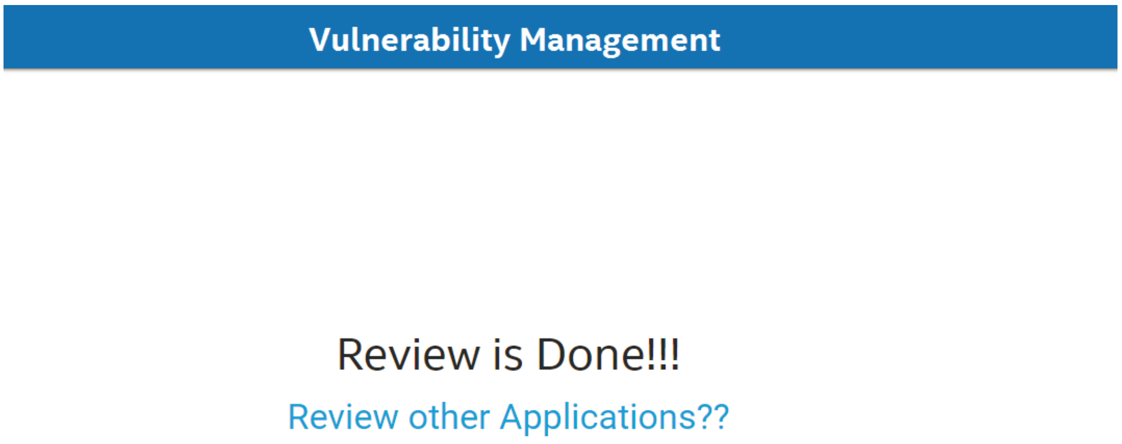
Subject Immediate Action Required - Your Application have Vulenrability

Hi Jason,

Your Application **"HelloSSO" (IAP Number: 12345)** have vulnerablility of **SQL Injection**, we recieved an email from external security researcher about this, we have reproduce all the steps and figure out that it's a **valid Vulnerability** and also **critical** one.
We are giving you **30 days** time frame to remidiate this vulnerability as it is marked as **critical**. Let us know if you need any help for the same.

Thanks,
Security Team

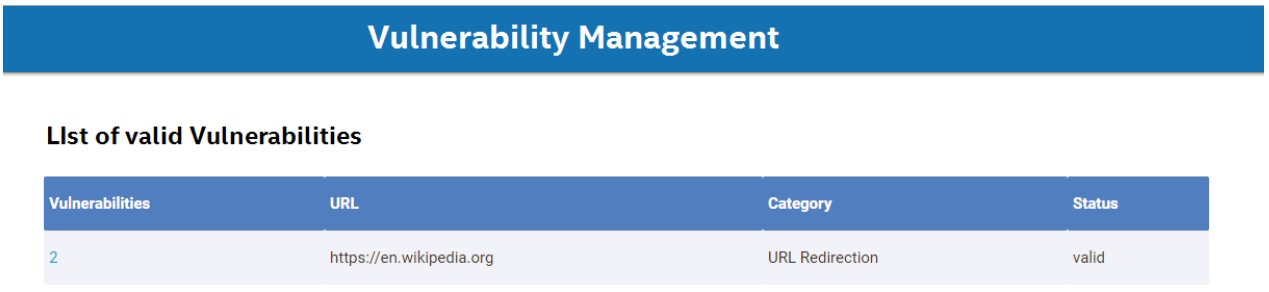
Figure 5.16: Auto-generated mail sent to the App Owner


Vulnerability Management

Review is Done!!!

Review other Applications??

Figure 5.17: Review completed and Navigate to other page to review more Bugs


Vulnerability Management

LIst of valid Vulnerabilities

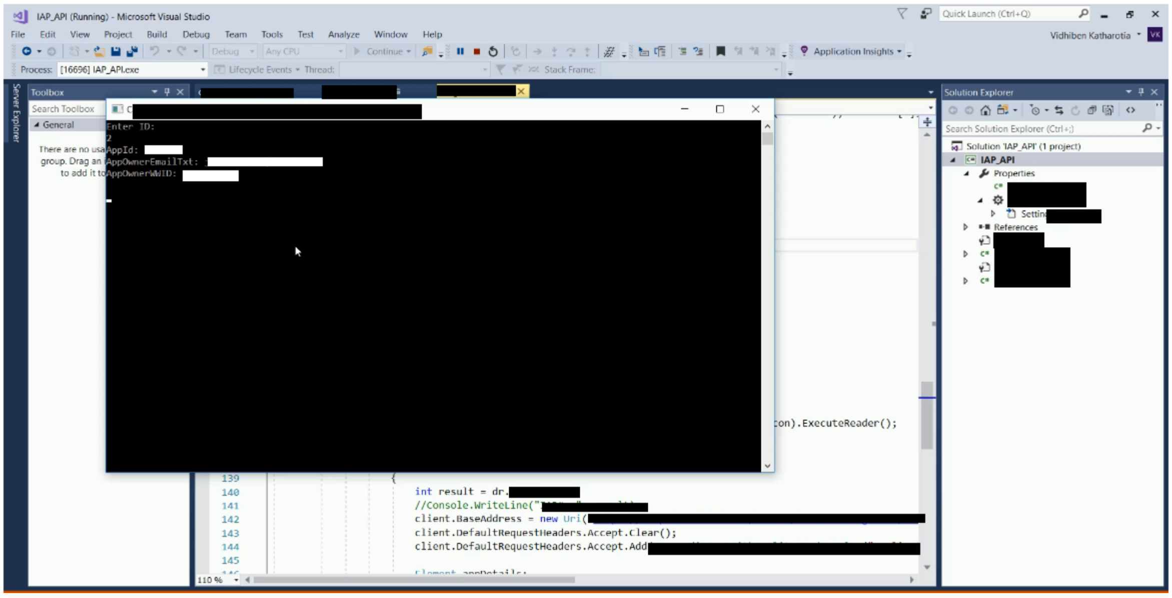
Vulnerabilities	URL	Category	Status
2	https://en.wikipedia.org	URL Redirection	valid

Figure 5.18: Once Higher level security team mark the bug as invalid it will not reflect on this page

5.2.4 Fourth Module

In this module I have used API service to fetch the App Owner's details from the restful APIs. This data is used to send an automated email to respective App Owners to notify them that this are the list of vulnerabilities external researcher have found in your application. All the details provided by the App Owner is fetched from the database.

As shown in Fig. 5.19 all the data can be fetch using c from the restful web APIs. Now the counter emails are generated and send to the app owner as per the criticality of the vulnerability. This is the batch job which runs twice in a week to all the app owners who's application have vulnerability.


IAP_API (Running) - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Quick Launch (Ctrl+Q)
Vidhiben Katharotia
Application Insights
Process: [16696] IAP_API.exe
Toolbox
Search Toolbox
General
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'IAP_API' (1 project)
IAP_API
Properties
References
110 %

```
Enter ID:
2
AppId: [redacted]
AppOwnerEmailTxt: [redacted]
AppOwnerWWID: [redacted]
```

```
                                                        on).ExecuteReader();
                {
                    int result = dr.[redacted]
                    //Console.WriteLine("[redacted]
                    client.BaseAddress = new Uri([redacted]
                    client.DefaultRequestHeaders.Accept.Clear();
                    client.DefaultRequestHeaders.Accept.Add[redacted]
```

Figure 5.19: Fetching data from the Restful Web APIs to get the details about App Owner

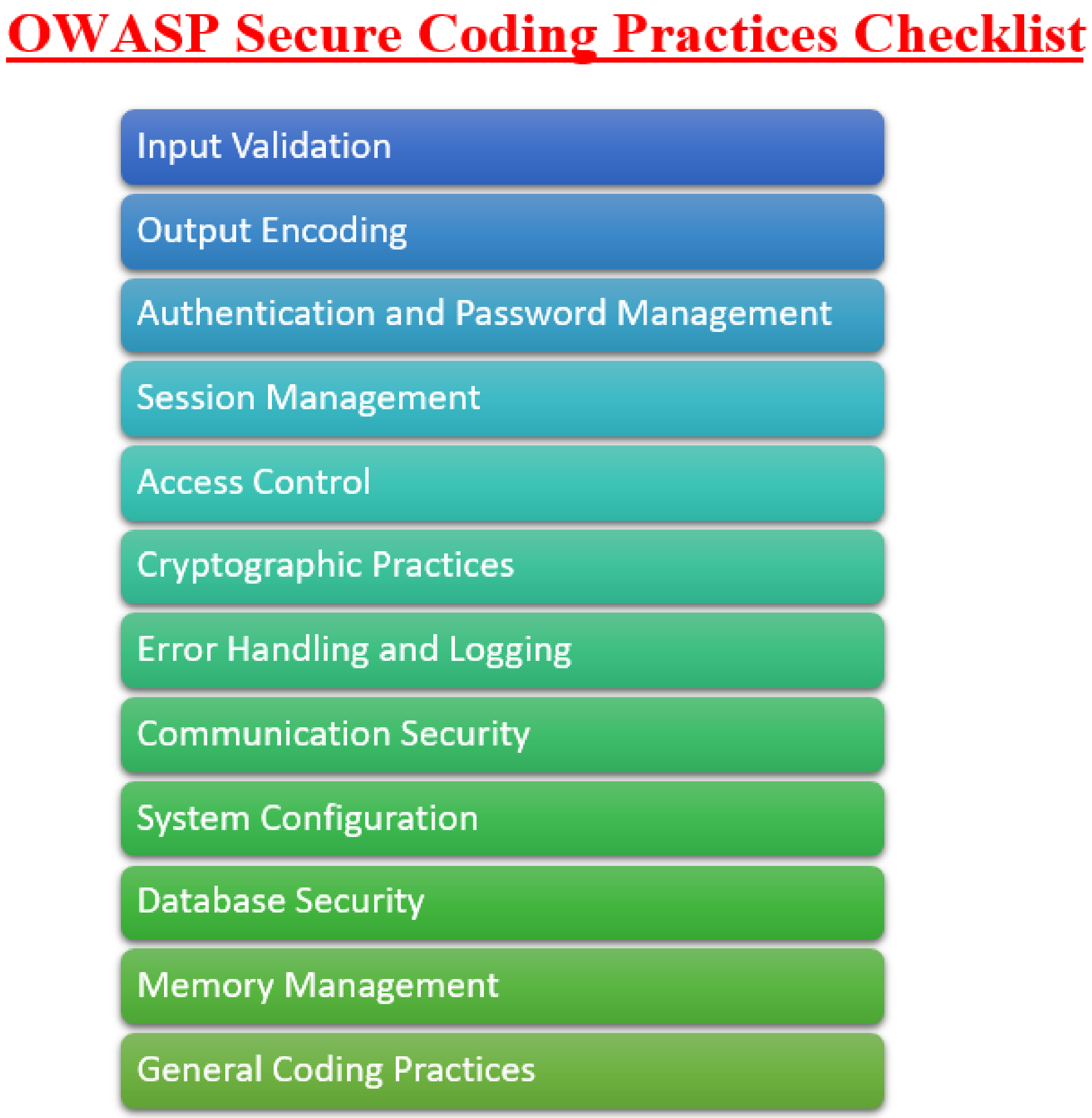
Chapter 6

Secure Coding Practices

As per the estimation of Gartner through 2020, 99/100 vulnerabilities which are exploited by the attacker are the ones which Security and the IT professionals already knows about it from last one year. Most of the organizations are facing hacks and security breaches which covers security bugs which security researchers are already aware about it. Before any application goes live it should go through the secure coding life cycle, so that there are very minimal chances of an attack like SQL injection, encryption exploit, XSS, etc.. can be reduced. There are certain ways to avoid such attacks which are listed below, this are nothing but some secure coding practices which any development team can adopt to make their application secure before it is externalized. Some basic understanding of security guidelines are required while building an secure application/software.

The main goal of securing the software is to maintain CIA which are Confidentiality, Integrity and Availability of data and information stored in the application.There are some security controls which needs to be implemented while developing any application. Now a days attackers are getting into application layer which is very dangerous. The listed techniques can be adopted to mitigate critical vulnerabilities in the application to prevent attacks.

Before mitigating the risks we need to understand what is Risk? Risk is nothing but the factor which causes a threat to the business, which leads to a big loss of reputation, money, etc. There are two different approaches from the development team side and attacker side, that how the application can be hacked. In most of the cases Development team's point of view of developing an application is what is the current requirement and what can done to full those requirements.While attacker's view to the application is to exploit it in such way that they can use that application to get important data, to harm organization's reputation or for personal use. So listed guidelines are designed for identifying the issue and how to mitigate from the issue to keep your application as well as organization secure. Here are some Secure coding practices suggested by OWASP which are listed below.


OWASP Secure Coding Practices Checklist

- Input Validation
- Output Encoding
- Authentication and Password Management
- Session Management
- Access Control
- Cryptographic Practices
- Error Handling and Logging
- Communication Security
- System Configuration
- Database Security
- Memory Management
- General Coding Practices

Figure 6.1: Secure coding Practices Checklist

So listed guidelines are designed for identifying the issue and how to mitigate from the issue to keep your application as well as organization secure. Here are some Secure coding practices suggested by OWASP which are listed below. Here is the detailed description of each of the Secure coding best practices.

1. **Input Validation**

    - Trusted entities like Server's must have all kinds of Data Validations
    - If the Validation fails there should be input rejection also
    - Data range should be validated
    - Data length should be validated
    - When redirection is happening data must be validated to check there is no malicious content
    - Also it is good to check the header request and response having on ASCII values
    - Apply white listing for validation
    - List out which are the trusted and untrusted data source and apply proper validation according to that
    - All the data coming from client side must be validated before it will go for further processing
    - Use proper encoding schema for inputs which application is taking

2. **Output Encoding**

    - All the encoding must be performed on the trusted entities only
    - Find out standard and the routine which is already tested for the outbound encoding
    - All the characters should be encoded properly though they seems to be safe
    - Sensitization should be done on all the outputs of the data which is untrusted for database queries.
    - For the OS commands all the untrusted output of the data must be sanitized properly
    - HTML entity encoding can also be used for better security

3. **Authentication and Password Management**

    - Authentication should be applied for all the pages except the pages which are not externally visible
    - Trusted system must have all the authentication controls
    - Trusted systems like server's must be provided with Password hashing
    - For transferring auth. credentials us HTTP POST method only
    - It is feasible to transfer only non-temporary passwords on the secure channel
    - There should be proper policies for the password length and characters
    - There should be some secure level of controls while resetting or changing passwords
    - There should be a policy for disabling the account if the user have attempt to logging more then the threshold
    - Before storing any password on the trusted system, it should be hashed properly
    - There should be time stamp give for the password resetting process, after certain time link for the password reset should get expired
    - The password which is temporary should be enforced to change while using it next time
    - User should get any notification via email or message that his/her's password is being resetting
    - Avoid using same password for different accounts
    - Make sure you are not marking remember me while entering the password
    - If user is doing some critical task make sure they are re-authenticated first

4. **Session Management**

    - Session id must be created on the trust worthy servers only
    - All the pages which are under protection of authentication must have logout feature

- Every time when re-authentication happens there should be generation of new session id
- While transmitting cookies through TLS/SSL connection make sure secure attribute is set
- Use proper access controls for server so that it will provide protection for server side session data which is accesses by unauthorized user
- There should be different session prior to login and after the login is successful
- Session id's should not be displayed directly or indirectly in the URL or any error message

5. **Access Control**

- It security information is not access by the application then deny all the requested access
- Any request comes first there should be authorization check to be done
- Provide resources to authorized users only
- Users who are authorized should be provided access for the protected URLs, rest should not
- Grant access for the application data and confidential data to only authorized users only
- Particular services should be provided to the specific authorized user only
- Protected features and functions should be assign to the users who are unauthorized only
- Applications which have logic flows must be compliant with the business rules
- Provide a time frame in which users or devices can perform specific number of transactions only

6. **Cryptographic Practices**

- Protect the valuable and sensitive information from unauthorized users
- It cryptographic modules fails, it should fail very securely
- Make use of the cryptographic algorithms to protect the data in the application

- Cryptographic algorithms should be kept on the secure servers only
- When your application is using random numbers or creating random numbers for provideing more security, it should use cryptographic random number generation algorithms only
- Make sure that cryptographic keys must be managed as per the security policies and rules

7. **Error Handling and Logging**

- Make sure that when error message displays it is not containing any sensitive information which may lead to security threat
- To prevent of showing any debugging information or stack trace related information make use of error handlers
- Create an error message that will display every time when error occurs
- When any error occurs the allocated memory should be free properly
- Only trusted system should have all the sensitive logging controls
- Ensure all the logs are secure properly because it have all the important log event related information
- Each and every input validations functionality should be logged properly
- Maintain a data for all the authentications which is being done
- Make sure all the failed authentication details should also be stored properly
- Create an entry of all the attempts which is being made to connect with invalid or the tokens which are expired
- All the exception created in the system must be logged properly
- Any attempt made to the TLS/SSL connection which is failed must be logged properly
- Sever sided code should be protected in a way that user is not able to download
- It is not advisable to store any passwords or cryptographic keys on the client side in clear text
- All the sensitive information must be encrypted properly

8. **Communication Security**

    - While transmitting the sensitive information on the channel it should be encrypted
    - There should be a valid domain name for the TLS certificates
    - If TLS connections fails then it should not go in to any insecure connection
    - Do proper utilization of all the TLC connection for all the external systems
    - There should be proper encoding for the characters for all the connections

9. **System Configuration**

    - All the servers, system components and related frameworks are working on updated versions
    - Not a single directory should be visible to the unauthorized users
    - There should be least privileges given for all the servers and it's process
    - If some files and features/functionality are not required then no need to store it
    - When any exceptions happens, and it fails then it should be failed securely
    - Before the deployment step remove all those unnecessary display of files, codes and comments
    - When HTTP response comes it should not contain any unwanted information in it
    - HTTP Methods which are not required should be disable
    - Production network and development environment should be isolated

10. **Database Security**

    - Parameterized queries must be used
    - Strongly typed has to be applied on variables
    - For accessing the database make use of the credentials which are secure only
    - While creating an application connection string should not be there as hard coded

- The connection should always be closed as early as possible
- If certain features and functionality are not required then turn it off
- If vendor's content is not necessary then remove that too
- Some accounts which are default is no longer required then disable them too

11. **File Management**

    - Any dynamic function should not be provided with the data which is supplied by the user directly
    - If the file is being uploaded then there should be some mechanism for re-authentication
    - Only some types of files are allowed to be uploaded
    - After the file is uploaded make sure that it is being validated by checking it's headers and extensions etc
    - File should be stored on the web server or the relevant database
    - File directories privileges should be turned off
    - Client should not be provided with the full/absolute file path
    - All the files and related data should be read-only
    - After the file is uploaded scan it to check whether it have virus or malware

12. **Memory Management**

    - For the data which is untrusted, there should be proper utilization done for the input and output validation controls
    - To prevent from buffer overflow make sure that buffer is as large as needed
    - Also check that source and destination buffer size are equal
    - When non executable stacks are available make use of that only
    - Also check the boundaries of buffer if the function which is calling is in the loop
    - Don't use the functions which are vulnerable and not known
    - After the function is completed free the memory which is allocated

13. **General Coding Practices**

- Use the manage code which is already tested and approved
- OS should not be issued with the commands directly
- The variables which are shared and some resources must be protected from unwanted access
- To make sure the that the interpreted code is secure make use of check-sums or hashes
- The functions which are dynamically executed should not be passed with data supplied by the user
- Do not allow the user to create a new code or do any alteration for the existing code
- Make use of safe updating, if automatic updates are there then make use of cryptographic signatures

Chapter 7

Remediation

After finding all the possible vulnerabilities in application, one report related to all this vulnerabilities is generated. With the help of this report this loopholes needs to be taken care otherwise attacker will take advantage of all this weakness in the application and perform some unethical steps which may lead to exploitation of sensitive information. So remediation process is very important after vulnerabilities are conformed. Security expert will review all the listed possible vulnerabilities in detail.

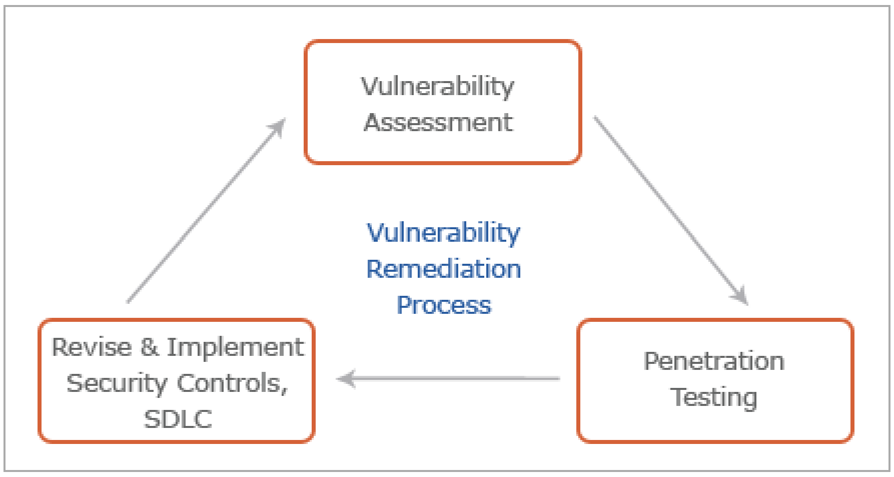

Vulnerability Assessment
Vulnerability Remediation Process
Penetration Testing
Revise & Implement Security Controls, SDLC

Figure 7.1: Vulnerability Remediation Process

As shown in Fig. 7.1 there are basically three process in vulnerability remediation. One is Vulnerability Assessment, second is Penetration testing, and last is Revise and Implement security controls.

- **Vulnerability Assessment:** It is the initial step of searching, prioritizing and reviewing the vulnerabilities in the application. This process have so many things which is common with risk assessment. Which also includes finding vulnerabilities and threats to each application.

- **Penetration testing:** It is the process of exploiting the possible vulnerabilities in the application to check whether application is responding to some specific types of malicious inputs or not. It also shows if the application is actually vulnerable then what it can damage during the real time attack.
- **Revise and Implement security controls:** In this step AppSec security expert revise all the step of the application which have vulnerabilities and try to cooperate with developers to make development process secure means help the developers to code securely.

To make sure this remediation plan is working efficiently the AppSec security experts will work with the developers to guide them how to use secure coding during building an application so that at the very basic level such kind of vulnerabilities can be avoided. AppSec security experts also suggest following secure coding practices to the developers for the secure development of any application.

- Input Validation should be done properly
- Prepared Statements should be used
- CSRF Tokens to be used
- HTML Encoding to be taken care
- Use escape sequence
- Password protection
- Technical message should not be displayed

Chapter 8

Conclusion

As the existing tools are not able to capture all the vulnerabilities during SAST process. So some of the vulnerabilities needs to be taken care during the source code review. For this manual code review needs to be done and this manual process is little bit time consuming. So as per the Application Security standards if code review is done properly also with the help of existing tools application can be prevented from the attackers.There is always loophole into the system but AppSec team is making sure that we can provide as much security as possible to your system. And to reduce this too much of manual work I have developed a web portal where all the data is stored in the database, so that it can be accessible in future also. In the development of this portal I have added some additional features which are earlier on that in the manual system, this will increase the potential of the system.

Bibliography

[1] Y.-W. Huang, F. Yu, C. Hang, C.-H. Tsai, D.-T. Lee, and S.-Y. Kuo, "Securing web application code by static analysis and runtime protection," in *Proceedings of the 13th international conference on World Wide Web*, pp. 40–52, ACM, 2004.

[2] N. Jovanovic, C. Kruegel, and E. Kirda, "Pixy: A static analysis tool for detecting web application vulnerabilities," in *Security and Privacy, 2006 IEEE Symposium on*, pp. 6–pp, IEEE, 2006.

[3] Y.-W. Huang, F. Yu, C. Hang, C.-H. Tsai, D.-T. Lee, and S.-Y. Kuo, "Securing web application code by static analysis and runtime protection," in *Proceedings of the 13th international conference on World Wide Web*, pp. 40–52, ACM, 2004.

[4] A. D. Brucker and T. Deuster, "Static application security testing," Nov. 4 2014. US Patent 8,881,293.

[5] M. Bartel, J. Boyer, B. Fox, B. LaMacchia, and E. Simon, "Xml-signature syntax and processing," *W3C recommendation*, vol. 12, p. 2002, 2002.

[6] M. McIntosh and P. Austel, "Xml signature element wrapping attacks and countermeasures," in *Proceedings of the 2005 workshop on Secure web services*, pp. 20–27, ACM, 2005.

[7] J. Somorovsky, "On the insecurity of xml security," *it-Information Technology*, vol. 56, no. 6, pp. 313–317, 2014.

[8] N. Antunes and M. Vieira, "Comparing the effectiveness of penetration testing and static code analysis on the detection of sql injection vulnerabilities in web services," in *Dependable Computing, 2009. PRDC'09. 15th IEEE Pacific Rim International Symposium on*, pp. 301–306, IEEE, 2009.

[9] M. Bruhn, M. Gettes, and A. West, “Identity and access management and security in higher education,” *EduCause Quarterly*, vol. 26, no. 4, pp. 12–17, 2003.

[10] A. Petukhov and D. Kozlov, “Detecting security vulnerabilities in web applications using dynamic analysis with penetration testing,” *Computing Systems Lab, Department of Computer Science, Moscow State University*, 2008.