# Web Application Security Testing Scope

**IMImobile Pvt Ltd**

**Plot no 770, Road no 44**

**Jubilee Hills, Hyderabad**

## Document Details

| | |
|---|---|
| **Document Title** | **IMImobile Web Application Security Testing Scope** |
| **Company** | IMImobile Pvt Ltd |
| **Date** | February 14 2018 |
| **Classification** | Confidential |
| **Version** | 1.0 |
| Author | Venkat Reddy Sreepuram |
| **Reviewer 1** | Srinivas Rao Allada |
| **Reviewer 2** | Santosh Vijaykumar Chadalavada |
| **Approved By** | Sunil TM |
| **Department** | Testing Hyderabad |

## Version History Information

| Date | Version | Author | Comments |
|---|---|---|---|
| Feb 14 2018 | v1.0 | Venkat Reddy Sreepuram | Initial Release |

## Contact

| | |
|---|---|
| **Name** | Venkat Reddy Sreepuram |
| **Address** | IMImobile Hyderabad |
| **Phone** | 91 86 86 85 84 86 |
| **Email** | Venkatreddy.s@imimobile.com |

# Table of contents

## 1. Introduction:

IMImobile Security Team will perform two types of Security assessments to ensure product security was good.

- Vulnerability Scanning
- Penetration Testing

## 2. Vulnerability Scanning:

Vulnerability scanning is also called as Vulnerability assessment. This is performed by using automated tools such as Acunetix Vulnerability Scanner, Qualys Guard Vulnerability Scanner, IBM App scan etc. The tool will itself performs scan on entire application and report to the user if any known Vulnerabilities are existing in the application.

## 3. Penetration testing

A vulnerability assessment simply identifies and reports noted vulnerabilities, whereas a penetration test (Pen Test) attempts to exploit the vulnerabilities to determine whether unauthorized access or other malicious activity is possible. This is simply called as Manual Security testing, where human will perform various web attacks in order to break into the web application. There are two different types of Penetration testing approaches.

- Black box Penetration Testing Approach
- White Box Penetration Testing Approach

### 3.1. Black Box Penetration Testing Approach

In penetration testing, black-box testing refers to a methodology where an ethical hacker has no knowledge of the system being attacked. The goal of a black-box penetration test is to simulate an external hacking. In this type, the attacker will target to make various attacks on Web Applications without knowing the internal structure code and also the user credentials.

### 3.2. White Box Penetration Testing Approach

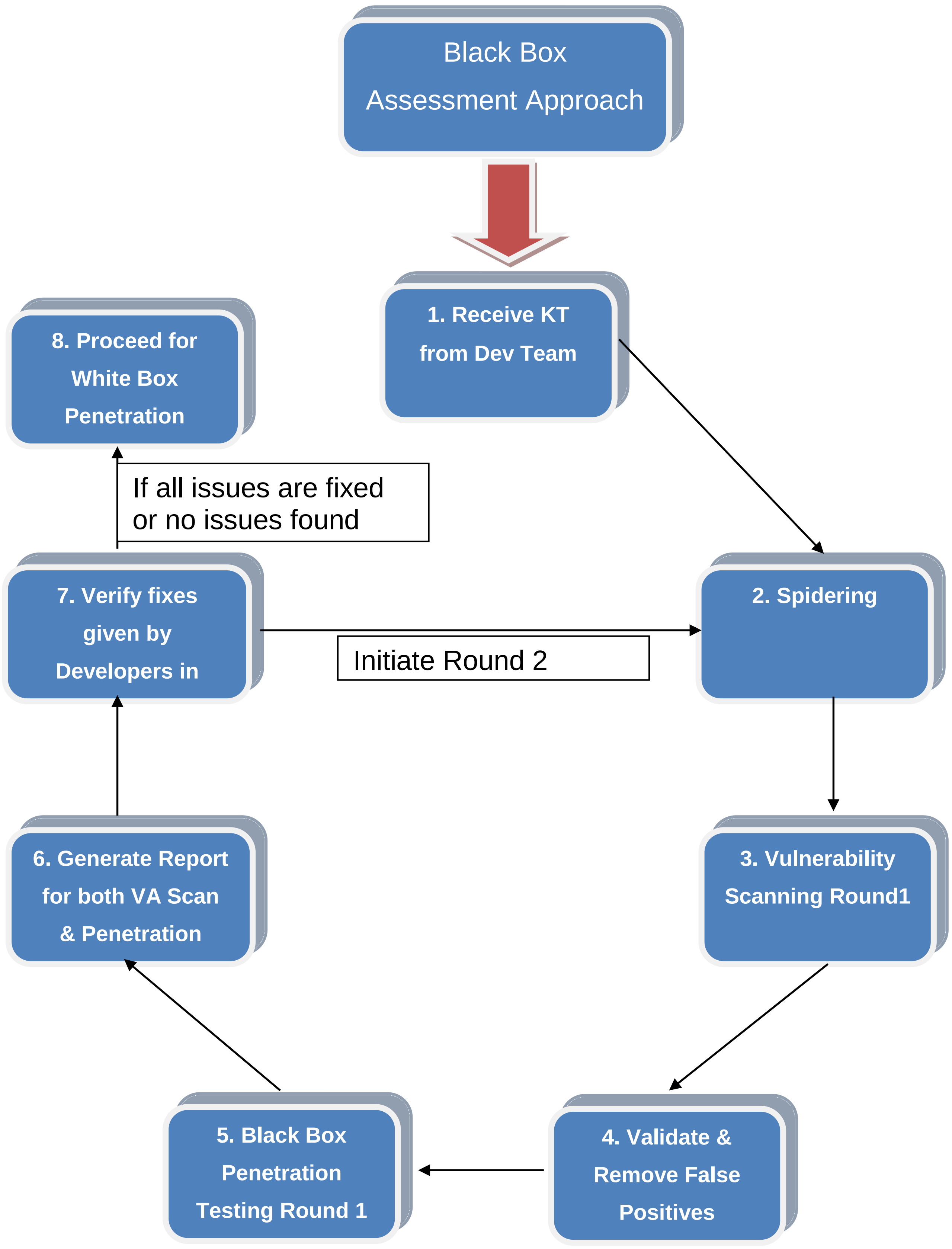
White box Penetration testing, is a penetration testing approach that uses the knowledge of the internals of the target system to elaborate the test cases. Attackers will know the complete information of the platform including the usernames and all the internal applications and access to the pages will be provided.

## 4. Requirements from Product Team:

While initiating a product Security testing, a meeting will be scheduled with Product and Development team with Security Team. After explaining the entire platform of the application, Product team manage will collect the inputs in below format and share with Security Team.

1. Product Name:
2. Type of Security Assessment required? (Scan/Pentest)
3. Apps/URL's to be covered
4. Average Response Time of Application
5. Required Report in (Days) (minimum of 4 days required for standard Pentest)
6. Project Management
   - Manager 1
   - Manager 2
7. Developer Team
   - Developer 1
   - Developer 2
8. QA/Test Team
   - Testing Engineer 1
   - Testing Engineer 2
9. Application Roles and Credentials
   - Admin Role (1 user)
   - Moderator Role (2 Users)
   - Agent Role (2 users)
   - User Role (2 users)

## 5. Black Box Security Assessment Approach


Black Box
Assessment Approach
1. Receive KT from Dev Team
2. Spidering
3. Vulnerability Scanning Round1
4. Validate & Remove False Positives
5. Black Box Penetration Testing Round 1
6. Generate Report for both VA Scan & Penetration
7. Verify fixes given by Developers in
Initiate Round 2
If all issues are fixed or no issues found
8. Proceed for White Box Penetration

## 5.1. Detailed Black Box Security Assessment Approach:

### 5.1.1. Phase 1:

In this phase we will receive KT from developers and architect about the full architecture of the application and what are the frameworks, underlying OS, technology and software's used. We won't go through the internal pages of the application at this phase.

### 5.1.2. Phase 2:

In this phase after collecting the required URL's and all the domains used in the application, we will use spiders that crawl through the Web looking for data. Spiders travel through website URLs and can pull data from web pages like email addresses. Spiders also are used to feed information found on websites to search engines.

### 5.1.3. Phase 3:

In this phase after spidering, we will perform a black box Vulnerability Scanning (Round1).Here we won't give any credentials to the scanner. It will be performed on the blank login page or template.

### 5.1.4. Phase 4:

In this phase we will analyse the vulnerabilities and errors that are raised in the scanner in the process of Vulnerability scan and remove all the false positives generated by the scanner.

### 5.1.5. Phase 5:

In this phase we will enumerate all the subdomains and directories or hidden pages available at the suffixes and prefixes of the website domain. Later we will try to break into the login or any other admin pages on the same domain.

### 5.1.6. Phase 6:

In this phase we will generate report of whatever we did till now and we will log the issues identified in the scanner and Manual, to particular developers and share the entire report.

### 5.1.7. Phase 7:

In this phase, Developers will fix the issues that are assigned to them and reassign to Security Team. Now we will verify each issues fixed by them again. In case issue was not fixed, we will re-assign them again.

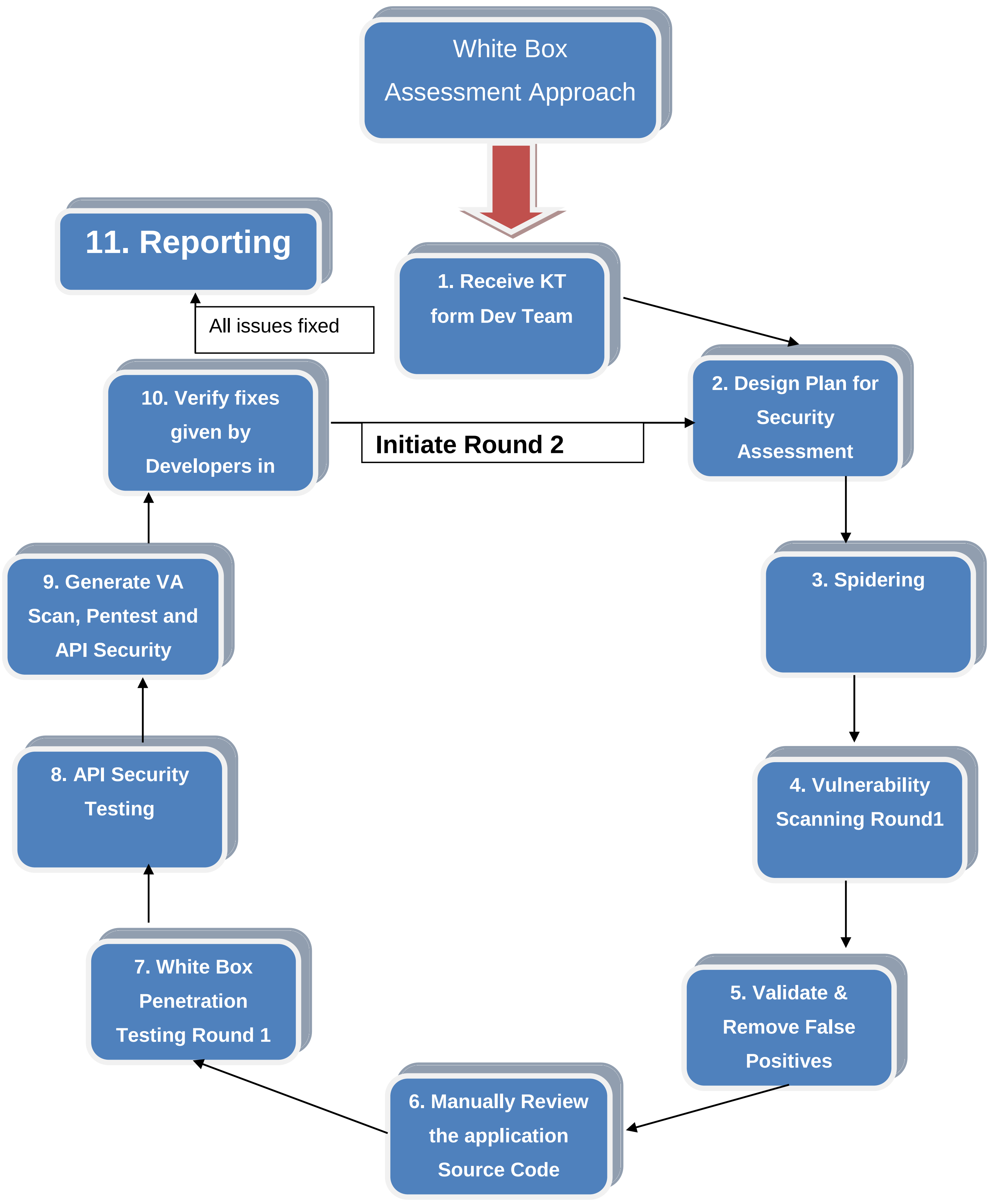
### 5.1.8. Phase 8:

In this phase if all the issues are fixed which are verified manually, raised in earlier stage, we will proceed to Round 2 and perform Scan and Pentest again. We will proceed with multiple rounds until all the count comes to zero.

### 5.1.9. Phase 9:

We will further proceed to white box penetration testing, after multiple rounds of Scans and Pentests.

## 6. White Box Penetration Testing Approach


White Box Assessment Approach
1. Receive KT form Dev Team
2. Design Plan for Security Assessment
3. Spidering
4. Vulnerability Scanning Round1
5. Validate & Remove False Positives
6. Manually Review the application Source Code
7. White Box Penetration Testing Round 1
8. API Security Testing
9. Generate VA Scan, Pentest and API Security
10. Verify fixes given by Developers in
Initiate Round 2
All issues fixed
11. Reporting

## 6.1. White Box Penetration testing Detailed Approach:

### 6.1.1. Phase 1 :( Requirements and KT)

In this phase we will receive KT from developers and architect about the full architecture of the application and what are the frameworks, underlying OS, technology and software’s used. We won’t go through the internal pages of the application at this phase. In this phase we will get the detailed information of Role based Access control and will collect 2 sets of Credentials of each level of users and also Development team and product manager should explain us from end to end including functionality.

### 6.1.2. Phase 2 :( Planning)

In this phase after collecting all the requirements from multiple teams, we will divide into multiple sections likes API calls, Dynamic pages and number of static pages in the application, and also will plan various things like timelines, when to complete what modules till reporting phase.

### 6.1.3. Phase 3 :( Spidering)

In this phase after collecting the required URL's and all the domains used in the application, we will use spiders that crawl through the Web looking for data. Spiders travel through website URLs and can pull data from web pages like email addresses. Spiders also are used to feed information found on websites to search engines. In white box Penetration Testing we will spider the application using the valid credentials. In this process, as we are giving authentication to application, we will get more pages and data.

### 6.1.4. Phase 4 :( Vulnerability Scanning Round 1)

In this phase after spidering, we will perform a white box Vulnerability Scanning (Round1).Here we won’t give any credentials to the scanner. It will be performed on the blank login page or template. We will give a pattern recording for how the tool should act with the URL or even we can give credentials to the scanner and it will automatically login and do scan.

### 6.1.5. Phase 5 :( Validating False Positives)

In this phase we will analyse the vulnerabilities that are raised in the scanner in the process of Vulnerability scan and remove all the false positives generated by the scanner.

### 6.1.6. Phase 6 :( Review source code manually)

In this phase we will get access to entire source code and analyse the source code if any hard corded values are there inside the code and also verify whether id can block malicious user requests.

### 6.1.7. Phase 7 :( In depth Penetration testing)

In this phase we will enumerate all the subdomains and directories or hidden pages available at the suffixes and prefixes of the website domain. Later we will try to break into the login or any other admin pages on the same domain. We will perform in depth penetration testing using various levels of users and perform various web attacks including OWASP TOP10.

### 6.1.8. Phase 8 :( API Security)

In this phase we will collect all the API calls used by the application internally and externally and exploit them as there may be threat from the insiders also.

### 6.1.9. Phase 9 :( Reporting Phase)

In this phase we will generate report of whatever we did till now and we will log the issues identified in the scanner, Manual and API's to particular developers and share the entire report.

### 6.1.10. Phase 10 :( Verification and validation)

In this phase, Developers will fix the issues that are assigned to them and reassign to Security Team. Now we will verify each issues fixed by them again. In case issue was not fixed, we will re-assign them again.

### 6.1.11. Phase 11 :( Decision making)

In this phase if all the issues are fixed which are verified manually, raised in earlier stage, we will proceed to Round 2 and perform Scan and Pentest again. We will proceed with multiple rounds until all the count comes to zero.

### 6.1.12. Phase 12 :( Reporting)

Once all the issue are fixed after multiple rounds, we will provide entire report stating that all the issues fixed along with evidence to the clients.

## 7. IMImobile Application Security Checklist:

Below checklist is the standard checklist used to perform penetration testing in the web applications. Based (static & dynamic pages) checklist will increase and decrease.

| S.No | Vulnerability | Severity | Exploitation | Business Impact |
|---|---|---|---|---|
| 1 | SQL Injection | HIGH | MODERATE | EXTREME |
| 2 | Broken Authentication and Session Management | HIGH | MODERATE | EXTREME |
| 3 | Cross Site Scripting | HIGH | EASY | MODERATE |
| 4 | Insecure Direct Object Reference | HIGH | EASY | MODERATE |
| 5 | Security Misconfiguration | HIGH | MODERATE | EXTREME |
| 6 | Sensitive Data Exposure | HIGH | EASY | EXTREME |
| 7 | Missing Functional Level Access Control | HIGH | EASY | MODERATE |
| 8 | HTTP Parameter Pollution | HIGH | EASY | MODERATE |
| 9 | Other Users Data | HIGH | EASY | EXTREME |
| 10 | Parameter tampering | HIGH | EASY | EXTREME |
| 11 | Credentials Sent in GET Method. | HIGH | EASY | MODERATE |
| 12 | Unauthorized Download/Upload | HIGH | MODERATE | MODERATE |
| 13 | XML Entity Injection | HIGH | MODERATE | MODERATE |
| 14 | Credentials Disclosure in UI | HIGH | EASY | MODERATE |
| 15 | Cross Domain Linkage | HIGH | DIFFICULT | LOW |
| 16 | Authentication/OTP Bypassing | HIGH | EASY | EXTREME |
| 17 | Directory Traversal | HIGH | MODERATE | EXTREME |
| 18 | Default Passwords | HIGH | EASY | MODERATE |
| 19 | Unrestricted Upload | HIGH | MODERATE | MODERATE |
| 20 | Configuration file source code disclosure. | HIGH | DIFFICULT | EXTREME |

| S.No | Vulnerability | Severity | Exploitation | Business Impact |
|---|---|---|---|---|
| 21 | Access URL without Authorization | HIGH | EASY | EXTREME |
| 22 | Browser Closed and Reopened, same session is active | HIGH | EASY | MODERATE |
| 23 | Apache Struts2 remote command execution (S2-045) | HIGH | MODERATE | MODERATE |
| 24 | Directory traversal in Spring framework | HIGH | DIFFICULT | LOW |
| 25 | HTTP.sys remote code execution vulnerability | HIGH | EASY | MODERATE |
| 26 | Microsoft IIS tilde directory enumeration | HIGH | DIFFICULT | LOW |
| 27 | Apache Tomcat directory host App base authentication | HIGH | MODERATE | EXTREME |
| 28 | Apache Tomcat WAR file directory traversal | HIGH | MODERATE | EXTREME |
| 29 | Multiple vulnerabilities fixed in PHP versions 5.5.12 | HIGH | DIFFICULT | LOW |
| 30 | User credentials are sent in clear text | HIGH | EASY | MODERATE |
| 31 | Hidden form input named price was found | HIGH | MODERATE | MODERATE |
| 32 | TRACE method is enabled | HIGH | MODERATE | EXTREME |
| 33 | Possible server path disclosure (Unix) | HIGH | DIFFICULT | EXTREME |
| 34 | Possible username or password disclosure | HIGH | EASY | EXTREME |
| 35 | Cross Site Request Forgery | MEDIUM | MODERATE | LOW |
| 36 | Components With Known Vulnerabilities | MEDIUM | EASY | MODERATE |
| 37 | Unvalidated Redirects and Requests | MEDIUM | MODERATE | LOW |
| 38 | Admin Pages Detected | MEDIUM | DIFFICULT | EXTREME |
| 39 | Internal Schema Like SQL Query Disclosure | MEDIUM | MODERATE | MODERATE |
| 40 | Session management | MEDIUM | MODERATE | MODERATE |

| S.No | Vulnerability | Severity | Exploitation | Business Impact |
|---|---|---|---|---|
| 41 | Broken Access Control | MEDIUM | EASY | LOW |
| 42 | HTML Injection | MEDIUM | MODERATE | MODERATE |
| 43 | Hidden in Source Code | MEDIUM | MODERATE | MODERATE |
| 44 | Host Header Attack | MEDIUM | DIFFICULT | LOW |
| 45 | Insecure Authentication | MEDIUM | EASY | MODERATE |
| 46 | Client Code Quality | MEDIUM | MODERATE | MODERATE |
| 47 | Server Side Validation Bypass | MEDIUM | MODERATE | MODERATE |
| 48 | Passwords in Cookies | MEDIUM | MODERATE | MODERATE |
| 49 | Predictable ID's | MEDIUM | EASY | EXTREME |
| 50 | Cluster Bomb | MEDIUM | MODERATE | EXTREME |
| 51 | Server Side Request Forgery | MEDIUM | MODERATE | MODERATE |
| 52 | Session Token in Cache/History | MEDIUM | EASY | EXTREME |
| 53 | Account Lock Out | MEDIUM | MODERATE | MODERATE |
| 54 | Password policy | MEDIUM | MODERATE | MODERATE |
| 55 | Brute Force Attack | MEDIUM | DIFFICULT | MODERATE |
| 56 | Virtual Path Disclosure | MEDIUM | MODERATE | MODERATE |
| 57 | Rotating Session ID's | MEDIUM | EASY | EXTREME |
| 58 | UnEncoded Characters | MEDIUM | MODERATE | LOW |
| 59 | Browser Cache Storing sensitive information | MEDIUM | EASY | EXTREME |
| 60 | AngularJS client-side template injection | MEDIUM | DIFFICULT | MODERATE |

| S.No | Vulnerability | Severity | Exploitation | Business Impact |
|---|---|---|---|---|
| 61 | Struts 2 development mode | MEDIUM | DIFFICULT | MODERATE |
| 62 | The DROWN attack (SSLv2 supported) | MEDIUM | MODERATE | LOW |
| 63 | Insecure crossdomain.xml file | MEDIUM | DIFFICULT | MODERATE |
| 64 | Vulnerable JavaScript library | MEDIUM | MODERATE | LOW |
| 65 | BREACH attack | MEDIUM | MODERATE | LOW |
| 66 | Development configuration file | MEDIUM | DIFFICULT | MODERATE |
| 67 | Apache JServ protocol service | MEDIUM | MODERATE | MODERATE |
| 68 | Application error message | MEDIUM | DIFFICULT | EXTREME |
| 69 | ASP.NET error message | MEDIUM | DIFFICULT | EXTREME |
| 70 | File upload | MEDIUM | MODERATE | LOW |
| 71 | Cookie(s) without HttpOnly flag set | MEDIUM | DIFFICULT | MODERATE |
| 72 | ASP.NET debugging enabled | MEDIUM | DIFFICULT | MODERATE |
| 73 | ASP.NET MVC version disclosure | MEDIUM | DIFFICULT | MODERATE |
| 74 | Session Cookie scoped to parent domain | MEDIUM | DIFFICULT | LOW |
| 75 | Possible relative path overwrite | MEDIUM | MODERATE | MODERATE |
| 76 | Possible virtual host found | MEDIUM | MODERATE | EXTREME |
| 77 | Content type is not specified | MEDIUM | DIFFICULT | MODERATE |
| 78 | Back Up Files Detected | LOW | MODERATE | EXTREME |
| 79 | Name is Used instead of Email at Login | LOW | DIFFICULT | EXTREME |
| 80 | Service/Banner Finger Printing | LOW | MODERATE | EXTREME |

| S.No | Vulnerability | Severity | Exploitation | Business Impact |
|---|---|---|---|---|
| 81 | HTTPs and Force HTTPs implementation | LOW | MODERATE | MODERATE |
| 82 | Click Jacking | LOW | MODERATE | LOW |
| 83 | AutoComplete Should Not Save Username and Password | LOW | EASY | LOW |
| 84 | Email ID's Detected. | LOW | EASY | EXTREME |
| 85 | Slow Response Time & Functional Issues | LOW | DIFFICULT | EXTREME |
| 86 | Cookie Secure Flag not Set | LOW | DIFFICULT | MODERATE |
| 87 | robots/logs/backup files | LOW | MODERATE | LOW |
| 88 | Documentation file | LOW | MODERATE | LOW |
| 89 | User Name Enumeration | LOW | MODERATE | EXTREME |
| 90 | Internal/Server Errors | LOW | DIFFICULT | EXTREME |
| 91 | Unicode transformation issues | LOW | DIFFICULT | LOW |
| 92 | Possible sensitive directories | LOW | MODERATE | MODERATE |
| 93 | Possible sensitive files | LOW | MODERATE | LOW |
| 94 | Cookie(s) without Secure flag set | LOW | DIFFICULT | LOW |
| 95 | Sensitive page could be cached | LOW | DIFFICULT | LOW |
| 96 | OPTIONS method is enabled | LOW | DIFFICULT | MODERATE |
| 97 | Possible internal IP address disclosure | LOW | MODERATE | MODERATE |
| 98 | Broken links | LOW | DIFFICULT | LOW |
| 99 | Web Application Firewall detected | LOW | DIFFICULT | MODERATE |
| 100 | Error page web server version disclosure | LOW | DIFFICULT | EXTREME |
| 101 | Slow HTTP Denial of Service Attack | LOW | DIFFICULT | EXTREME |

## 8. Tools Used by IMImobile for Web-App Security testing:

IMImobile Security team have licensed versions of below tools for Vulnerability Scanning and Penetration testing.

### 8.1. Professional Tools

- Acunetix Vulnerability Scanner
- Burp Suite Professional
- Qualys Guard

### 8.2. Open Source Tools

Apart from Licensed tools, IMImobile Security team uses multiple Open Source tools and customized scripts in order to perform successful Penetration testing. Below are the Open Source tools used by our team.

- Burp Suite Free and Commercial Version
- Vega
- OWASP ZAP
- w3af
- Iron WASP
- Xenotix
- SQL Map
- SQL Ninja
- OWASP CSRF Tester
- Wapiti
- Dirbuster
- Brutus
- Hydra
- Wireshark
- Google Dorks
- XPath Injector
- Customized Hack Bar
- Wikto and Nikto

### 8.2.1. Acunetix Vulnerability Scanner:

Acunetix allows you to secure your websites and web applications quickly and efficiently. The Web Scanner launches an automatic security audit of a website and generate a detailed report of Vulnerabilities of particular website.

### 8.2.2. Burp Suite Professional:

Burp Suite is a graphical tool for testing Web application security. It’s developed to provide a comprehensive solution for web application security checks. In addition to basic functionality, such as proxy server, scanner and intruder, the tool also contains more advanced options such as a spider, a repeater, a decoder, a comparer, an extender and a sequencer.

### 8.2.3. Qualys Guard:

QualysGuard is a popular SaaS (software as a service) vulnerability management offering. It's web-based UI offers network discovery and mapping, asset prioritization, vulnerability assessment reporting and remediation tracking according to business risk. Internal scans are handled by Qualys appliances which communicate back to the cloud-based system. It can be even installed in our local machines also, if we want to generate the report faster and easier.

### 8.2.4. Vega:

Vega is a free and open source web security scanner and web security testing platform to test the security of web applications. Vega can help you find and validate SQL Injection, Cross-Site Scripting (XSS), inadvertently disclosed sensitive information, and other vulnerabilities. It is written in Java, GUI based, and runs on Linux, OS X, and Windows.

### 8.2.5. OWASP ZAP:

The OWASP Zed Attack Proxy (ZAP) is one of the world’s most popular free security tools. It can help you automatically find security vulnerabilities in your web applications while you are developing and testing your applications. It’s also a great tool for experienced pen testers to use for manual security testing.

### 8.2.6. w3af:

w3af (web application attack and audit framework) is an open-source web application security scanner. The project provides a vulnerability scanner and exploitation tool for Web applications. It provides information about security vulnerabilities for use in penetration testing engagements.

### 8.2.7. Iron WASP:

Iron WASP (Iron Web application Advanced Security testing Platform) is an open source system for web application vulnerability testing. It is designed to be customizable to the extent where users can create their own custom security scanners using it.

### 8.2.8. Xenotix:

Xenotix XSS Exploit Framework is an advanced Cross Site Scripting (XSS) vulnerability detection and exploitation framework. It provides Zero False Positive scan results with its unique Triple Browser Engine (Trident, Web Kit, and Gecko) embedded scanner. Xenotix Scripting Engine allows you to create custom test cases and add-ons over the Xenotix API. It is incorporated with a feature rich Information Gathering module for target Reconnaissance.

### 8.2.9. SQL Map:

SQLmap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections.

### 8.2.10. SQL Ninja:

Sqlninja is a tool targeted to exploit SQL Injection vulnerabilities on a web application that uses Microsoft SQL Server as its back-end. Its main goal is to provide a remote access on the vulnerable DB server, even in a very hostile environment. It should be used by penetration testers to help and automate the process of taking over a DB Server when a SQL Injection vulnerability has been discovered.

### 8.2.11. OWASP CSRF Tester:

Cross-Site Request Forgery (CSRF) is an attack whereby the victim is tricked into loading information from or submitting information to a web application for which they are currently authenticated. The problem is that the web application has no means of verifying the integrity of the request. The OWASP CSRFTester Project attempts to give developers the ability to test their applications for CSRF flaws.

### 8.2.12. Wapiti:

Wapiti is a vulnerability scanner for web applications. It currently search vulnerabilities like XSS, SQL and XPath injections, file inclusions, command execution, XXE injections, CRLF injections... It use the Python programming language.

### 8.2.13. Dirbuster:

DirBuster is a multi-threaded java application designed to brute force directories and files names on web/application servers. Often is the case now of what looks like a web server in a state of default installation is actually not, and has pages and applications hidden within.

### 8.2.14. Brutus:

Brutus Password Cracker is one of the fastest, most flexible remote password crackers you can get your hands on – it's also free to download Brutus. ... A common approach (brute-force attack) is to try guesses repeatedly for the password and check wether the tool can crck the password.

### 8.2.15. Hydra:

Hydra is a parallelized login cracker which supports numerous protocols to attack. It is very fast and flexible, and new modules are easy to add. This tool makes it possible for researchers and security consultants to show how easy it would be to gain unauthorized access to a system remotely.

### 8.2.16. Wireshark:

Wireshark, a network analysis tool formerly known as Ethereal, captures packets in real time and display them in human-readable format. Wireshark includes filters, colour coding, and other features that let you dig deep into network traffic and inspect individual packets. But IMImobile Security team uses it for auditing WebAppliations also in packet tracing and data flowing from Web Application.

### 8.2.17. Google Dorks:

A Google dork query, sometimes just referred to as a dork, is a search string that uses advanced search operators to find information that is not readily available on a website. Google dorking, also known as Google hacking, can return information that is difficult to locate through simple search queries.

### 8.2.18. XPath Injector:

XPath Injection attacks occur when a web site uses user-supplied information to construct an XPath query for XML data. By sending intentionally malformed information into the web site, an attacker can find out how the XML data is structured, or access data that he may not normally have access to. He may even be able to elevate his privileges on the web site if the XML data is being used for authentication (such as an XML based user file).

### 8.2.19. Customized Hack Bar:

This toolbar will help you in testing SQL injections, XSS holes and site security. It is NOT a tool for executing standard exploits and it will NOT teach you how to hack a site. Its main purpose is to help a developer do security audits on his code.

### 8.2.20. Nikto and Wikto:

Nikto is an Open Source (GPL) web server scanner which performs comprehensive tests against web servers for multiple items, including over 6700 potentially dangerous files/programs, checks for outdated versions of over 1250 servers, and version specific problems on over 270 servers. Wikto is a tool that checks for flaws in webservers. It provides much the same functionality as Nikto but adds various interesting pieces of functionality, such as a Back-End miner and close Google integration. Wikto is written for the MS .NET environment and registration is required to download the binary and/or source code.

# 9. Standards and Frameworks used by IMI

Penetration testing methodology and standards are key to success for this ethical hacking technique that can help security professionals evaluate information security measures. The below are the various penetration testing methodologies and standards used by various orgnaisations.Some of the organisations follow their own standards. The below are the various standards followed by **IMI Security Team**, but they will follow their own standards rather than Open Source Security Standards.

- ✓ OWASP (Open Web Application Security Project)
- ✓ PTES (Penetration testing Execution Standard)
- ✓ OSSTMM (Open Source Security Testing Methodology Manual)
- ✓ NIST (National Institute of Standards and Technology)
- ✓ WASC (Web Application Security Consortium)
- ✓ IMImobile Internal Security Testing Standard

## 9.1. OWASP (Open Web Application Security Project)

**The Open Web Application Security Project (OWASP),** an online community, produces freely-available articles, methodologies, documentation, tools, and technologies in the field of web application security and many other domains.

**OWASP Top Ten:** The "Top Ten", first published in 2003, is regularly updated in year 2017.It aims to raise awareness about application security by identifying some of the most critical risks facing organizations

**OWASP Development Guide:** The Development Guide provides practical guidance and includes J2EE, ASP.NET, and PHP code samples. The Development Guide covers an extensive array of application-level security issues.

**OWASP Testing Guide:** The OWASP Testing Guide includes a "best practice" penetration testing framework that users can implement in their own organizations and a "low level" penetration testing guide that describes techniques for testing most common web application and web service security issues.

**OWASP Releases TOP10 Critical Vulnerabilities for every three to 4 years.**

- OWASP2017
- OWASP2013
- OWASP2010
- OWASP2007
- OWASP2003

OWASP TOP 10 Vulnerabilities in Detail

### 9.1.1. A1: Injection

Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or by accessing data without proper authorization.

### 9.1.2. A2: Broken Authentication

Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.

### 9.1.3. A3: Sensitive Data Exposure

Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.

### 9.1.4. A4: XML External Entities (XXE)

Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks

### 9.1.5. A5: Broken Access Control

Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

### 9.1.6. A6: Security Misconfiguration

Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched and upgraded in a timely fashion.

### 9.1.7. A7: Cross-Site Scripting (XSS)

XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

### 9.1.8. A8: Insecure Deserialization

Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.

### 9.1.9. A9: Using Components with Known Vulnerabilities

Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defences and enable various attacks and impacts.

#### 9.1.10. A10: Insufficient Logging & Monitoring

Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

## 9.2. PTES (Penetration testing Execution Standard)

The **penetration testing execution standard** consists of seven (7) main sections. These cover everything related to a penetration test - from the initial communication and reasoning behind a pentest, through the intelligence gathering and threat modelling phases where testers are working behind the scenes in order to get a better understanding of the tested organization, through vulnerability research, exploitation and post exploitation, where the technical security expertise of the testers come to play and combine with the business understanding of the engagement, and finally to the reporting, which captures the entire process, in a manner that makes sense to the customer and provides the most value to it.

This version can be considered a v1.0 as the core elements of the standard are solidified, and have been "road tested" for over a year through the industry. A v2.0 is in the works soon, and will provide more granular work in terms of "levels" - as in intensity levels at which each of the elements of a penetration test can be performed at. As no pentest is like another, and testing will range from the more mundane web application or network test, to a full-on red team engagement, said levels will enable an organization to define how much sophistication they expect their adversary to exhibit, and enable the tester to step up the intensity on those areas where the organization needs them the most. Some of the initial work on "levels" can be seen in the intelligence gathering section.

Following are the main sections defined by the standard as the basis for penetration testing execution:

- ✓ Pre-engagement Interactions
- ✓ Intelligence Gathering
- ✓ Threat Modelling
- ✓ Vulnerability Analysis
- ✓ Exploitation
- ✓ Post Exploitation
- ✓ Reporting

### 9.3. OSSTMM (Open Source Security Testing Methodology Manual)

OSSTMM stands for Open Source Security Testing Methodology Manual. It is a peer-reviewed manual of security testing and analysis which result in verified facts. These facts provide actionable information that can measurably improve your operational security. By using the OSSTMM you no longer have to rely on general best practices, anecdotal evidence, or superstitions because you will have verified information specific to your needs on which to base your security decisions. One way to assure a security analysis has value is to know it has been done thoroughly, efficiently, and accurately. For that you need to use a formal methodology called as OSSTM.

### 9.4. NIST (National Institute of Standards and Technology)

NIST is headquartered in Gaithersburg, Maryland, and operates a facility in Boulder, Colorado. NIST's activities are organized into laboratory programs and extramural programs. Effective October 1, 2010, NIST was realigned by reducing the number of NIST laboratory units from ten to six. NIST Laboratories include:

- ✓ Centre for Nanoscale Science and Technology (CNST)
- ✓ Communications Technology Laboratory (CTL)
- ✓ Engineering Laboratory (EL)
- ✓ Information Technology Laboratory (ITL)
- ✓ NIST Centre for Neutron Research (NCNR)

- ✓ Material Measurement Laboratory (MML)
- ✓ Physical Measurement Laboratory (PML)

### 9.5. WASC (Web Application Security Consortium)

The Web Application Security Consortium (WASC) is a worldwide organization devoted to the establishment, refinement and promotion of Internet security standards. The consortium, which was founded in January 2004, consists of independent members as well as those associated with corporations, government agencies and academic institutions.

The WASC's mandate includes researching, discussing, and publishing information about Web application security issues. The organization individuals and enterprises about such issues and the countermeasures that can be taken against specific threats, and acting as an advocate for users of the Internet, and in particular, for individuals and organizations devoted to Web application security. The WASC is vendor-neutral, although members may belong to corporations involved in the research, development, design, and distribution of Web security related products.

### 9.6. IMImobile Internal Security Testing Standard

IMIMobile Security Team have developed their own security standards following all the Open Source security standards (OWASP,PTES, OSSTMM,NIST,WASC).IMIMobile Security team have made around 100+ security checklist, where IMI Team will execute all those Checklist while performing penetration testing on any application.

# 10. Reporting Format - Sample Finding

## 10.1. Critical - Parameter Tampering in Payment:

### 10.1.1. Description:

Attacker will intercept the request between web browser and remote server using proxy tool, and manipulates the original price to custom price set by attacker. Payment amount will be shown to manipulated amount. If payment amount was 1200 rupees, by intercepting the attacker will modify 1200 to 1 rupee and purchase anything for 1 rupee.

### 10.1.2. Risk Score:

**10.0**

### 10.1.3. Severity:

**CRITICAL**

### 10.1.4. Ease of Exploitation:

**EASY**

### 10.1.5. Business Impact:

**HIGH**

### 10.1.6. Affected URL:

https://imimobilesecurityscope.com/payment

### 10.1.7. Impact:

- Attacker can bypass the payment and can recharge 1000 rupees by paying 0.10 rupee and even lesser.
- Critical Business Impact will occur.

### 10.1.8. Solution:

- Use Encryption to transfer the data.
- Payment must be transferred only while the user uses https mode.
- Use transaction IDs and validate at vendor side.

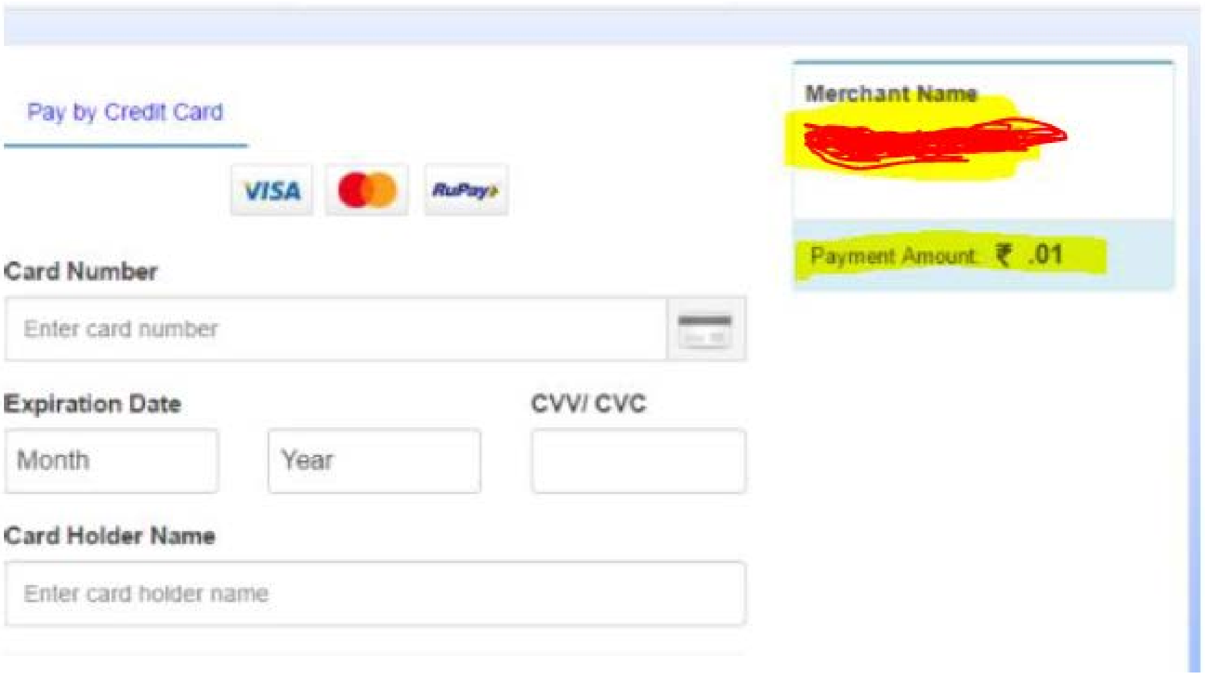
### 10.1.9. Refernce URL's for implementation:

- http://searchsecurity.techtarget.com/magazineContent/Secure-online-payment-system-requires-end-to-end-encryption
- http://www.mastercard.com/gateway/payment-processing/point_to_point_encryption.html

### 10.1.10. Request and Response Analysis:

```
POST /payment HTTP/1.1
Host: https://imimobilesecurityscope.com
Cache-Control: max-age=0
Origin: https://imimobilesecurityscope.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/58.0.3029.110 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: text/html,application/xhtml xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: https://imimobilesecurityscope.com
imipay_ack={"tid":"636319255888888888","mobileNo":"1234567890","channel":"WE
B","Operator":"AR","number":"1234567890","amount":"1200","amntid":
"email":"hbhjhj@gmail.com","username":"1234567890"}&venkat_MerchTxnRef=2342342
34ksdfkjfsdkjfjk23423423
&spv_Amount=10.00&card_type=02&CUST_MOBILE_
NUMBER=&venakat_ReturnURL=https://imimobilesecurityscope.comPaymentStatus&s
ervice=martconnect&action=2&CUST_MMID=
```

## 10.1.11. Evidence:



Pay by Credit Card

VISA RuPay

Card Number

Enter card number

Expiration Date

Month

Year

CVV/ CVC

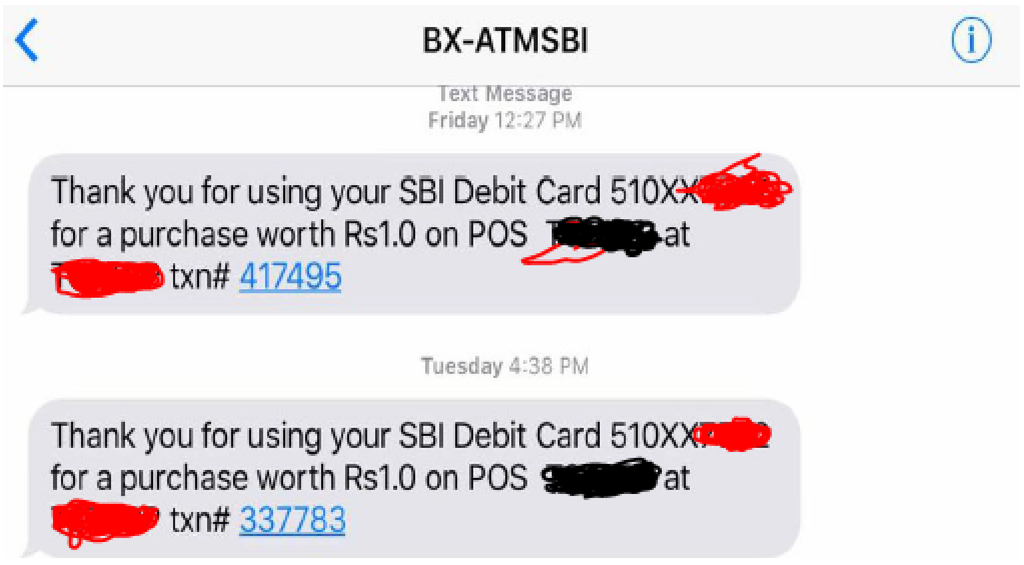
Card Holder Name

Enter card holder name

Merchant Name

[redacted]

Payment Amount ₹ .01



BX-ATMSBI

Text Message
Friday 12:27 PM

Thank you for using your SBI Debit Card 510XX[redacted] for a purchase worth Rs1.0 on POS [redacted] at [redacted] txn# 417495

Tuesday 4:38 PM

Thank you for using your SBI Debit Card 510XX[redacted] for a purchase worth Rs1.0 on POS [redacted] at [redacted] txn# 337783

# Contact

Venkat Reddy Sreepuram
Application Security

Phone: +91 (0)23555945 Ext: 234

## London

IMImobile Europe Ltd.
5 St John's Lane
EC1M 4BH
London
United Kingdom
Phone: +44 20 300 86232
europe.sales@imimobile.com

## Atlanta

IMImobile
Tower Place 200
3348 Peachtree Rd. NE.
Atlanta, GA. USA
30326
Phone: +1 407 216 1984
america.sales@imimobile.com

## Dubai

IMImobile VAS Ltd. FZE
P.O. Box 293598
Office # 624, Building 5EA
Dubai Airport,
U.A.E.
Phone: +971 46091 690
mea.sales@imimobile.com

## Hyderabad

IMImobile Pvt. Ltd.
Plot No.770, Road No 44
Jubilee Hills
Hyderabad
India
Phone: +91 40 23 555 945
apac.sales@imimobile.com

## Johannesburg

Clearwater Office Park
Building No. 3, First Floor
Corner Christiaan De Wet and Millennium Boulevard
Strubensvalley
Roodepoort, Gauteng, South Africa
Phone: +27 11 991 2900
info@archerdigital.co.za