Sample Report

Security Assessment Report

November 1, 2017

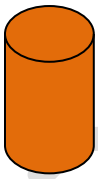
Report Prepared by:

InstaSafe Technologies

Executive Summary

INSTASAFE was contracted by Demo Limited to conduct a penetration test in order to determine its exposure to a targeted attack. All activities were conducted in a manner that simulated a malicious actor engaged in a targeted attack against demobile app with the goals of discovering application vulnerabilities

APPLICATION HEALTH

HIGH	MEDIUM	LOW
		
1	**6**	**2**

Introduction

Demo LTD engaged INSTASAFE to perform android app testing on Demobile beginning on 20th MAR 2017 and ending on 21st MAR 2017.

Objective

The objective of this assessment was to assess the overall security posture of the application from grey box & black box perspective. This includes determining the application's ability to resist common attack patterns and identifying vulnerable areas in the internal or external interfaces that may be exploited by a malicious user.

Scope

The scope of this engagement was limited to components and interfaces specific to Demobile android application.

Methodology

Assessment Type

Automated testing checked for false positive, than intensive manual testing using owasp top10 mobile app testing methodology.

Risk Assessment Methodology

The severity assigned to each vulnerability was calculated using the NIST 800-163 standard. The vulnerabilities in this appendix are broken into three hierarchical levels, A, B, and C. The A level is referred to as the vulnerability class and is the broadest description for the vulnerabilities specified under that level. The B level is referred to as the sub-class and attempts to narrow down the scope of the vulnerability class into a smaller, common group of vulnerabilities. The C level specifies the individual vulnerabilities that have been identified. The purpose of this hierarchy is to guide the reader to finding the type of vulnerability they are looking for as quickly as possible.

Table 1. Android Vulnerabilities, A Level

Type	Description	Negative Consequence
Incorrect Permissions	Permissions allow accessing controlled functionality such as the camera or GPS and are requested in the program. Permissions can be implicitly granted to an app without the user's consent.	An app with too many permissions may perform unintended functions outside the scope of the app's intended functionality. Additionally, the permissions are vulnerable to hijacking by another app. If too few permissions are granted, the app will not be able to perform the functions required.
Exposed Communications	Internal communications protocols are the means by which an app passes messages internally within the device, either to itself or to other apps. External communications allow information to leave the device.	Exposed internal communications allow apps to gather unintended information and inject new information. Exposed external communication (data network, Wi-Fi, Bluetooth, NFC, *etc.*) leave information open to disclosure or man-in-the-middle attacks.
Potentially Dangerous Functionality	Controlled functionality that accesses system-critical resources or the user's personal information. This functionality can be invoked through API calls or hard coded into an app.	Unintended functions could be performed outside the scope of the app's functionality.
App Collusion	Two or more apps passing information to each other in order to increase the capabilities of one or both apps beyond their declared scope.	Collusion can allow apps to obtain data that was unintended such as a gaming app obtaining access to the user's contact list.
Obfuscation	Functionality or control flows that are hidden or obscured from the user. For the purposes of this appendix, obfuscation was defined as three criteria: external library calls, reflection, and native code usage.	1. External libraries can contain unexpected and/or malicious functionality. 2. Reflective calls can obscure the control flow of an app and/or subvert permissions within an app. 3. Native code (code written in languages other than Java in Android) can perform unexpected and/or malicious functionality.
Excessive Power Consumption	Excessive functions or unintended apps running on a device which intentionally or unintentionally drain the battery.	Shortened battery life could affect the ability to perform mission-critical functions.
Traditional Software Vulnerabilities	All vulnerabilities associated with traditional Java code including: Authentication and Access Control, Buffer Handling, Control Flow Management, Encryption and Randomness, Error Handling, File Handling, Information Leaks, Initialization and Shutdown, Injection, Malicious Logic, Number Handling, and Pointer and Reference Handling.	Common consequences include unexpected outputs, resource exhaustion, denial of service, *etc.*

A		B		C	
A1	Permission of the Behavior Error	B1	Over Granting	C1	Over Granting in Code
				C2	Over Granting in API
		B2	Under Granting	C3	Under Granting in Code
				C4	Under Granting in API
		B3	Developer Created Permissions	C5	Developer Created in Code
				C6	Developer Created in API
		B4	Implicit Permission	C7	Granted through API
				C8	Granted through Other Permissions
				C9	Granted through Grandfathering
A2	Exposed Communications	B5	External Communications	C10	Bluetooth
				C11	GPS
				C12	Network/Data Communications
				C13	NFC Access
		B6	Internal Communications	C14	The purpose of unprotected
				C15	Unprotected Activity
				C16	Unprotected Services
				C17	Unprotected Content Providers
				C18	Unprotected Broadcast Receivers
				C19	Debug Flag
A3	Potentially Dangerous Functionality	B7	Direct Addressing	C20	Memory Access
				C21	Internet Access
		B8	Potentially Dangerous API	C22	Cost Sensitive APIs

				C23	Personal Information APIs
				C24	Device Management APIs
		B9	Privilege Escalation	C25	Altering File Privileges
				C26	Accessing Super User/Root
A4	Application Collusion	B10	Content Provider/Intents	C27	Unprotected Content Providers
				C28	Permission Protected Content Providers
				C29	Pending Intents
		B11	Broadcast Receiver	C30	Broadcast Receiver for Critical Messages
		B12	Data Creation/Changes/Deletion	C31	Creation/Changes/Deletion to File Resources
				C32	Creation/Changes/Deletion to Database Resources
		B13	Number of Services	C33	Excessive Checks for Service State
A5	Obfuscation	B14	Library Calls	C34	Use of Potentially Dangerous Libraries
				C35	Potentially Malicious Libraries Packaged but Not Used
		B15	Native Code Detection	C36	
		B16	Reflection	C37	
		B17	Packed Code	C38	
A6	Excessive Power Consumption	B18	CPU Usage	C39	
		B19	I / O	C40	

Summary of Vulnerabilities Found

Finding	Severity
Mpin Leakage Via Log	A1B4C3
Application is vulnerable to PIN authentication bypass vulnerability	A1B1C2
Application has set insecure Permissions	A1B4C8
Weak Encoding Technique used in the application	A5B17C34
Attacker can have full access to the mobile application source code	A1B1C1
Option method Enabled	A4B13C33
Default server page found on server	A4B12C31
Sensitive Activity Exported	A2B6C16
Application displays web server banner in response	A5B16C36

Vulnerability Details

1.Issue – Mpin Leakage Via Log

RISK –Medium

Description-when developer of the application accidently leaks the data. Well any developer would never want to leak the data but in some scenarios he assumes that the particular data is only accessible to the application not to any adversary. Often Developers leave debugging information publicly. So any application with READ_LOGS permission can access those logs and can gain sensitive information through that

Proof Of Concept –

Steps to produce –

1. Open app
2. Run this command pidcat com.demobile
3. Enter MPIN and it will be reflected in log

RECOMENDATION - Avoid creating logs when applications crashes and if logs are sent over the network then ensure that they are sent over an SSL channel. And Use strong encoding.

REFERENCES-

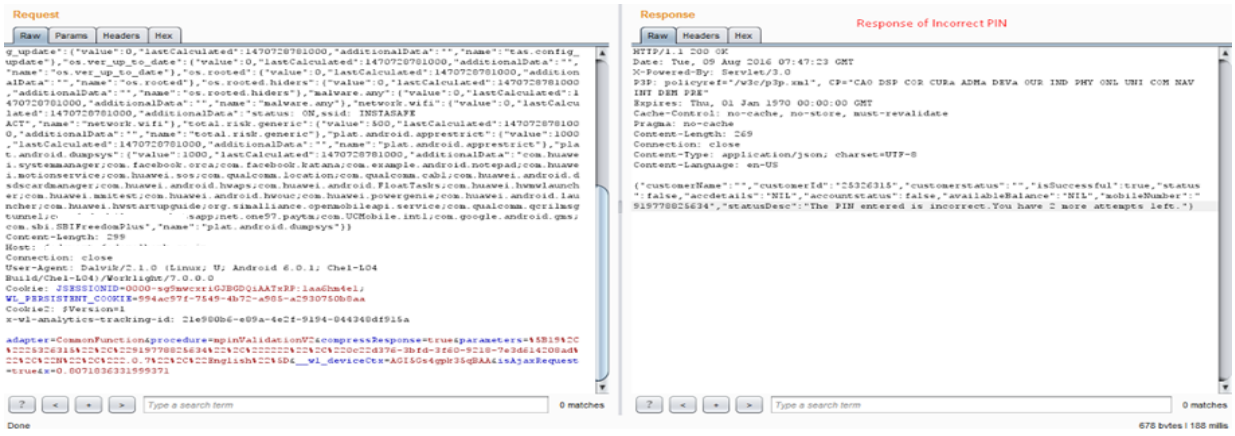
https://www.owasp.org/index.php/Mobile_Top_10_2014-M4

2.Issue – Application is vulnerable to PIN authentication bypass vulnerability

RISK –High

Description-An attacker can log in to the application without pin

Prood Of Concept - 1.when we give an incorrect pin



2.Replace the incorrect pin response with correct pin possible response

{"customerName":"MOBILETEST A\/C","customerId":"25326315","customerstatus":"A","isSuccessful":true,"status":true,"WL-Authentication-Success":{"wl_remoteDisableRealm":{"userId":"null","attributes":{},"isUserAuthenticated":1,"displayName":"null","deviceId":"null"}},"accdetails":["25326315;00352200000037;CA0037;0.00"],"accountstatus":true,"availableBalance":"NIL","mobileNumber":"919778825634","statusDesc":"success"}

3.Now you can see the loggedin screen.

RECOMENDATION - Properly check authentication request and response at both client and server side.

References-

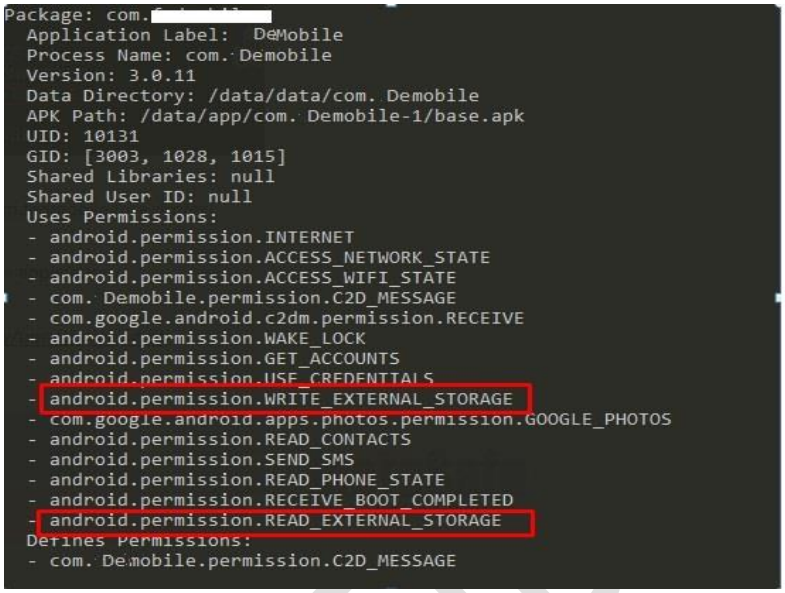
https://www.owasp.org/index.php/Testing_for_Bypassing_Authentication_Schema_%28OWASP-AT-005%29

3. Issue – Application has set insecure Permissions

RISK –Medium

Description-Permission mechanism that enforces restrictions on the specific operations that a particular process can perform, and per-URI permissions for granting ad hoc access to specific pieces of data. It was observed that application has set insecure permissions, which will create security threat to an application.

POC -



Steps to produce –

1. Run command : run app.package.info –a com.Demobile (use drozer)

RECOMENDATION–

Implement or set only necessary permissions to your application.

References:

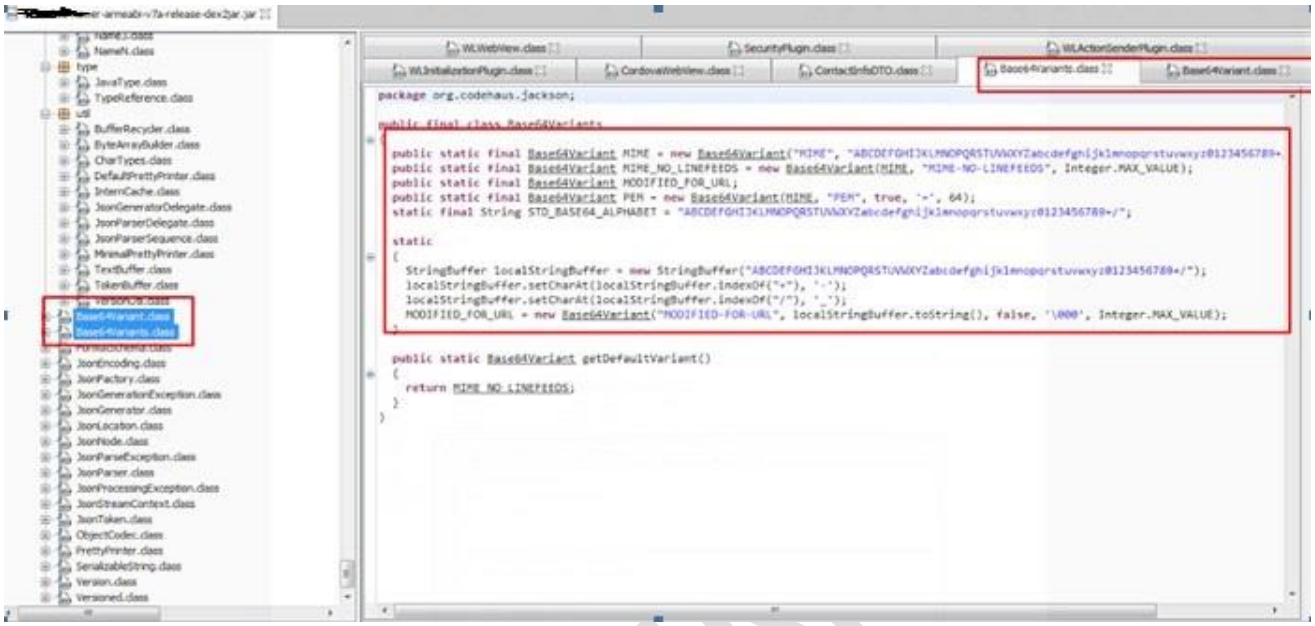
http://developer.android.com/guide/topics/security/permissions.html

4.Issue – Weak Encoding Technique used in the application.

RISK –Medium

Description-In computers, encoding is the process of putting a sequence of characters (letters, numbers, punctuation, and certain symbols) into a specialized format for efficient transmission or storage. Decoding is the opposite process -- the conversion of an encoded format back into the original sequence of characters. Encoding and decoding are used in data communications, networking, and storage. The term is especially applicable to radio (wireless) communications systems.

Proof Of Concept –



Steps to produce –

1. Follow the screenshot

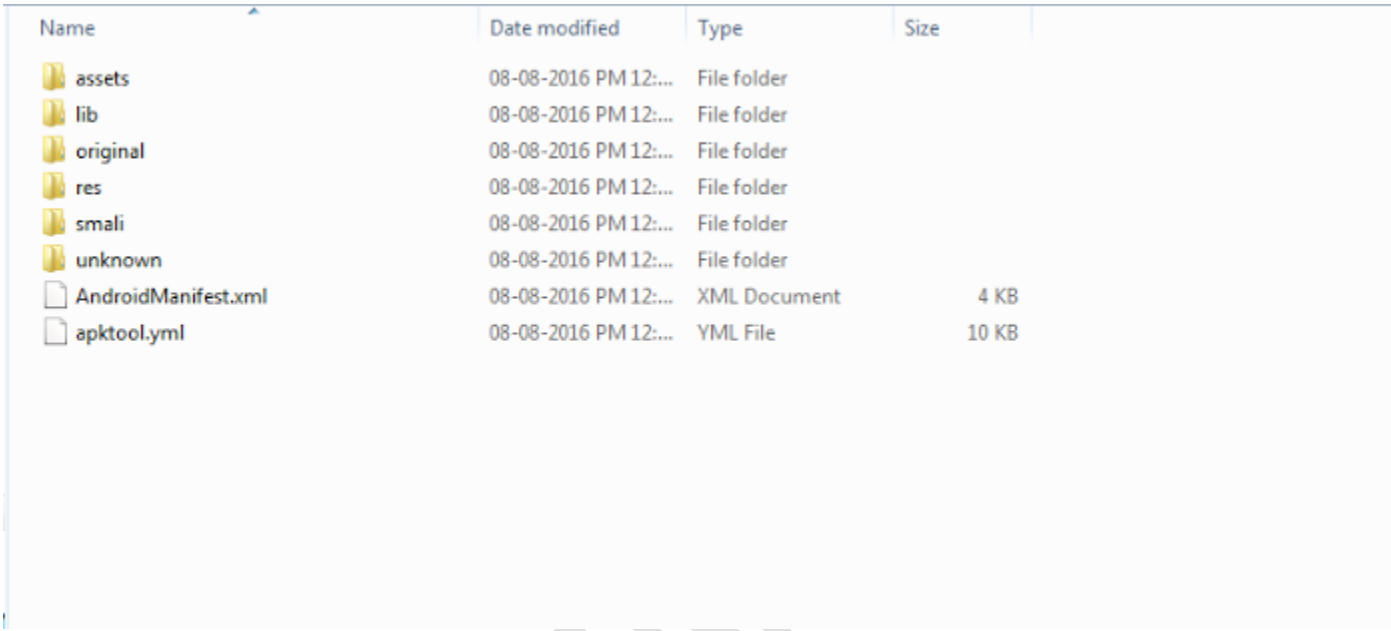
RECOMENDATION –Use strong encryption techniques or salted hash approach instead of encoding.

5.Issue – Attacker can have full access to the mobile application source code.

RISK – Medium

Description-It was observed that the application source code can be accessed easily with the help of several tools. By this an attacker can able to access all packages inside the ".APK" file which contains resource files, different bundles, package information and preference information.

POC -



Name	Date modified	Type	Size
assets	08-08-2016 PM 12:...	File folder	
lib	08-08-2016 PM 12:...	File folder	
original	08-08-2016 PM 12:...	File folder	
res	08-08-2016 PM 12:...	File folder	
smali	08-08-2016 PM 12:...	File folder	
unknown	08-08-2016 PM 12:...	File folder	
AndroidManifest.xml	08-08-2016 PM 12:...	XML Document	4 KB
apktool.yml	08-08-2016 PM 12:...	YML File	10 KB

Steps to produce –

1. For source code run command apktool d demobile.apk
2. For java code run command d2j –dex2jar demobile.apk

RECOMENDATION -

Critical or sensitive information should not be disclosed in application source code rather the code should be obfuscated properly.

Implement copy protection or activation scheme mechanism. Also implement code obfuscation techniques.

References-

http://www.techrepublic.com/blog/software-engineer/protect-your-android-apps-with-obfuscation/

6.Issue – Option method Enabled

RISK –Medium

Description- The OPTIONS method provides a list of the methods that are supported by the web server, it represents a request for information about the communication options available on the request/response chain identified by the Request-URI.

POC –

Steps to produce –

1. Run $nmap --script http--methods smart.demo.co.in

RECOMENDATION -

Disable OPTION methods from the server.

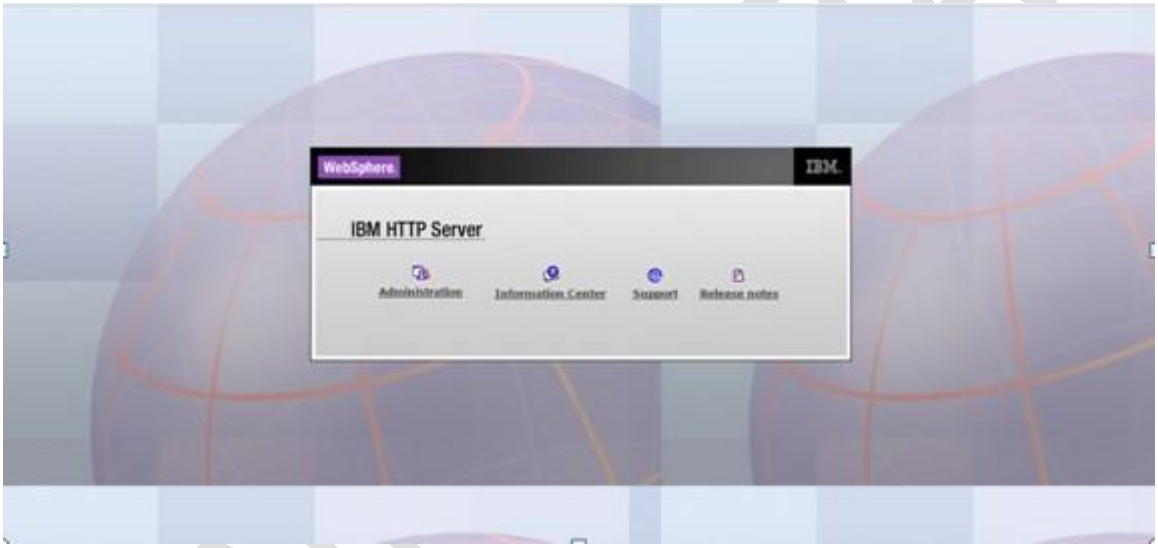
Reference-http://acunetix.com/vulnerabilities/web/options-method-is-enabled

7.Issue – Default server page found on server

RISK – Low

Description- Every website is built inside directories on a Web server. And each Web page is a separate file on that Webserver. But sometimes, when you go to a URL, there is no file listed in the URL. But there is still a file that the Web server needs to serve in order for that URL to display anything other than an error page. This file is the default page for that directory.

Proof Of Concept –



Steps To Produce –

1. Visit https://smart.demo.co.in/

RECOMENDATION –

Remove the default page from server

REFERENCES-

https://tools.cisco.com/security/center/viewAlert.x?alertId=38700

http://www-304.ibm.com/support/docview.wss?uid=swg24039898

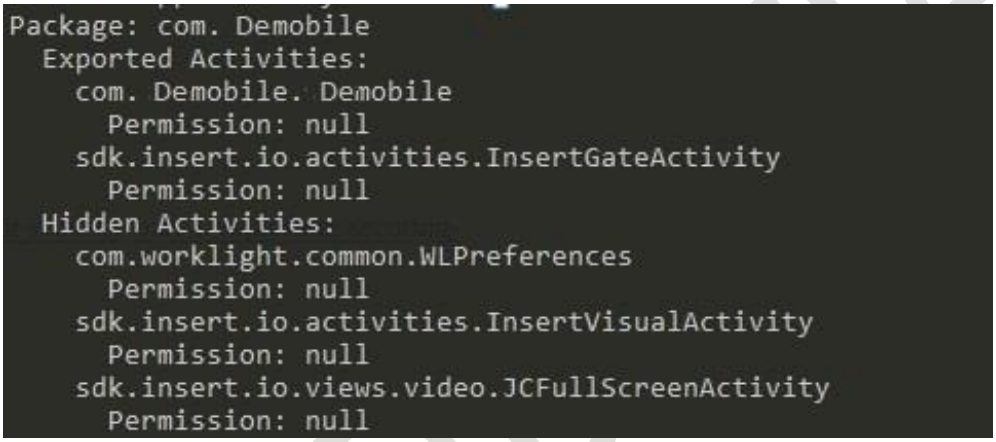
http://www.securiteam.com/securitynews/5MP331FHPW.html

8. Issue – Sensitive Activity Exported

RISK –Medium

Description- If access to an exported Activity is not restricted; any application will be able to launch the activity. This may allow a malicious application to gain access to sensitive information, modify the internal state of the application, or trick a user into interacting with the victim application while believing they are still interacting with the malicious application. Here the below activity shown in screenshot is not restricted.

Proof Of Concept–



Steps to produce –

1. Cmd: run app.activity.info -a com.Demobile –u (Use Drozer)

RECOMENDATION –

set android:exported="false" for that activity

REFERENCES-

https://cwe.mitre.org/data/definitions/926.html

http://resources.infosecinstitute.com/android-hacking-security-part-1-exploiting-securing-application-components/

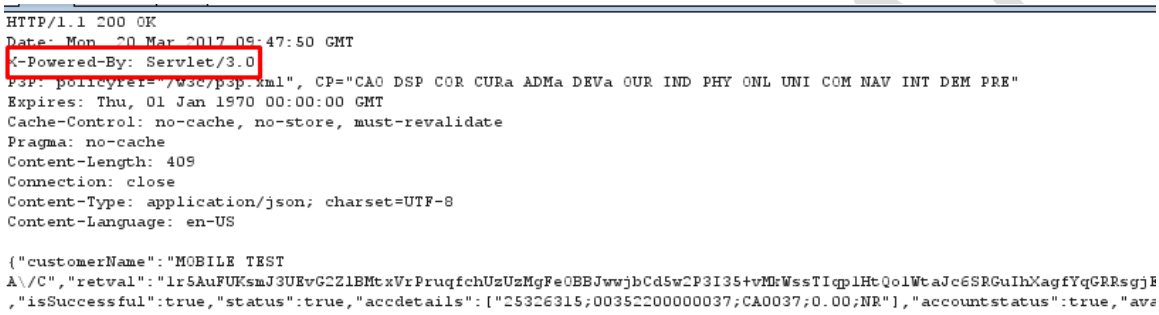
9.Issue – Application displays web server banner in response

RISK –Low

Parameter –X-powered by

Description-HTTP responses from the web server reveal information about the type and version of the web server, which can be used by an attacker. An attacker can exploit the publicly known vulnerabilities of servlet version.

Proof Of Concept -



Steps To Produce –

1. Check server header in screenshot

RECOMENDATION -

Remove default banner wherever possible. Also update to current version.

Some of our customers:

Recognitions:

Did we get you interested?

hello@instasafe.com
+91 8880220044
www.instasafe.com
facebook.com/instasafe
@instasafe
linkedin.com/company/instasafe