Week 9 – Web and Mobile Application Hacking

Agenda

OWASP top 10 attack and defense class and labs

Mobile security labs

- Mobile security attacks
- Crack mobile apps (e.g. Android apps)

Mobile Securing methods

OWASP Top 10 & Countermeasures

A2 – BROKEN AUTHENTICATION AND SESSION MANAGMENT

Authentication

Authentication is the process of verifying the user is really the one he/she claimed to be

Most web application authenticate by user their ID + password

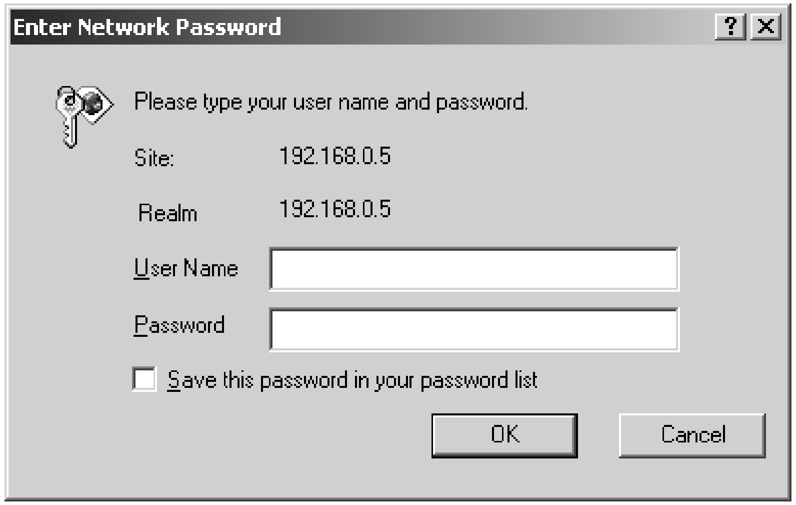
HTTP Basic Authentication

Supported by most browsers

Username/password passed in the clear with each request

Password may be stored in any format

Authentication through Authorization header in Base 64 decoder



Enter Network Password

Please type your user name and password.

Site:	192.168.0.5
Realm	192.168.0.5
User Name	
Password	

☐ Save this password in your password list

HTTP Digest Authentication

Supported by few browsers

Username/password used to build a request-dependent hash for each request

Effective password is the MD5 hash of the user's password

Effective password must be stored in the clear (encryption doesn't count)

HTTP Digest Authentication

Digest Authentication Example

- Server request authentication in a 401 message:

  -
    ```
    WWW-Authenticate: Digest realm="realm",
            qop="auth,auth-int",
            algorithm="MD5",
            nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
            opaque="5ccc069c403ebaf9f0171e9517f40e41"
    ```
- Client send authentication information:

  -
    ```
    Authorization: Digest username="user",
            realm="realm",
            nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
            uri="/dir/index.html",
            qop=auth, nc=00000001, nonce="0a4f113b",
            response="6629fae49393a05397450978507c4ef1",
            opaque="5ccc069c403ebaf9f0171e9517f40e41"
    ```
- Involves the use of hashing algorithm (by default MD5) and nonce to produce digest and checksum

Certificate-based Authentication

Using digital certificate for authentication

Based on the certificate of the client to identify the owner of the certificate

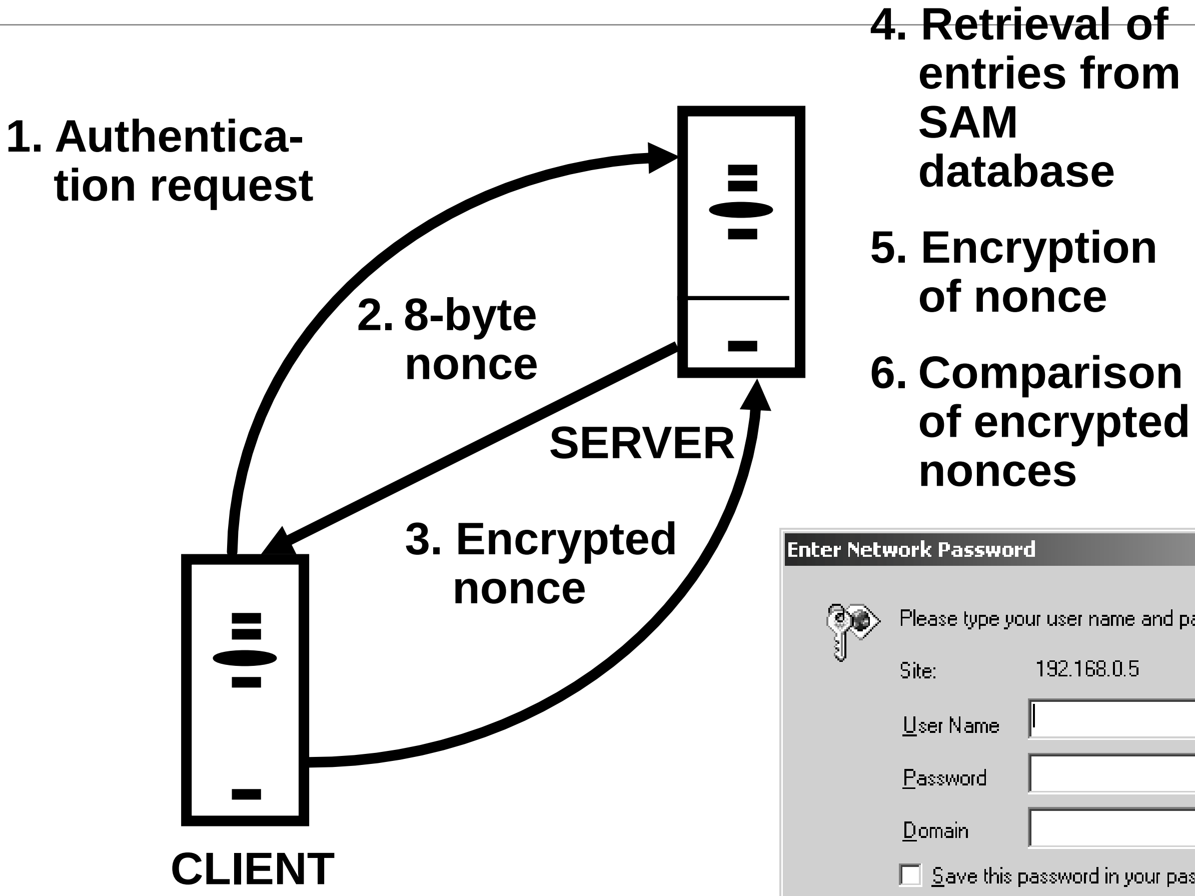
Server authentication (by public certificate)

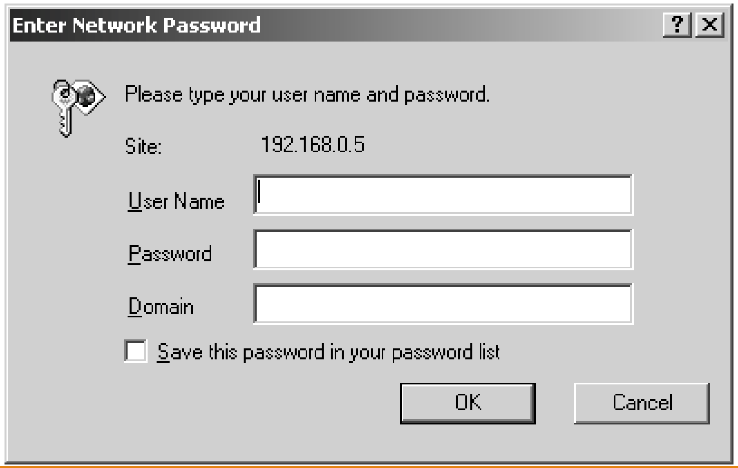
Client authentication (Optional)

Can also be supported through, LDAP (a directory access protocol)

SSL/TLS is a generic method of encrypting application-layer network traffic using x.509 certs for authentication

NTLM Authentication


1. Authentication request
2. 8-byte nonce
SERVER
3. Encrypted nonce
CLIENT
4. Retrieval of entries from SAM database
5. Encryption of nonce
6. Comparison of encrypted nonces


Enter Network Password

Please type your user name and password.

Site: 192.168.0.5

User Name

Password

Domain

☐ Save this password in your password list

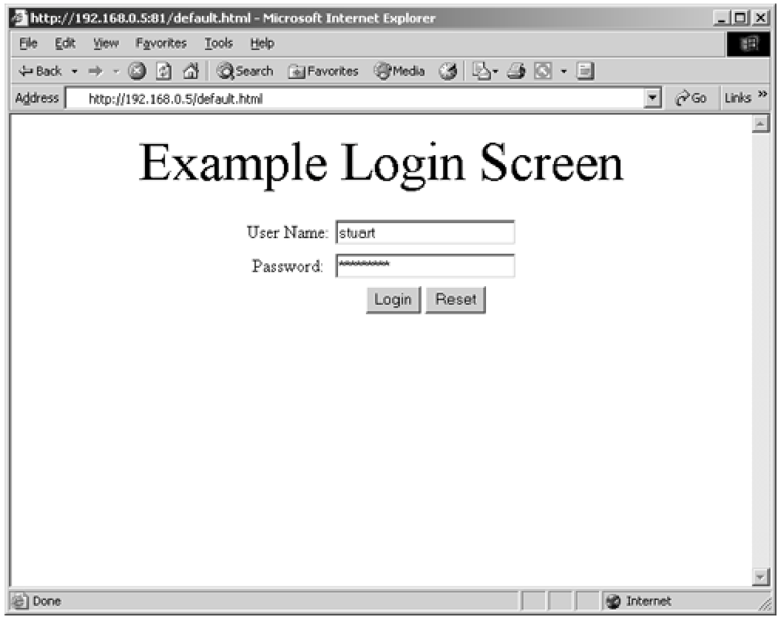
Forms-based Authentication

Authentication through web forms

Composed of <FORM> and <INPUT>

Forms contain hidden field called state or in cookie

Mainly storing the credentials in database


http://192.168.0.5:81/default.html - Microsoft Internet Explorer

Address http://192.168.0.5/default.html

Example Login Screen

User Name: stuart

Password: *********

Factor of Authentication

There are 3 factors of authentication

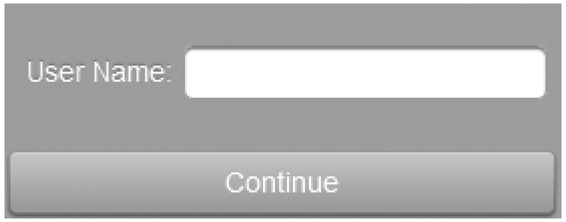
- What you know
  - Password, secrets, user ID, etc
- What you have
  - Hardware tokens, smart cards, etc
- Who you are
  - Biometric information: iris, finger prints, palm data

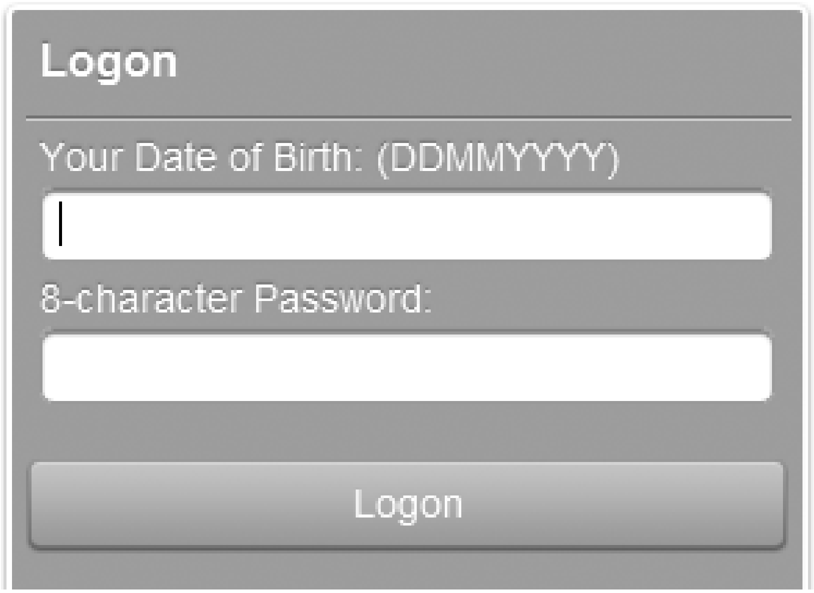
Multi-factor authentication:

- Require the user to provide valid form of more than 1 of the above factors

Two Factor Authentication?

Is this considered two factor authentication?


User Name:


Logon

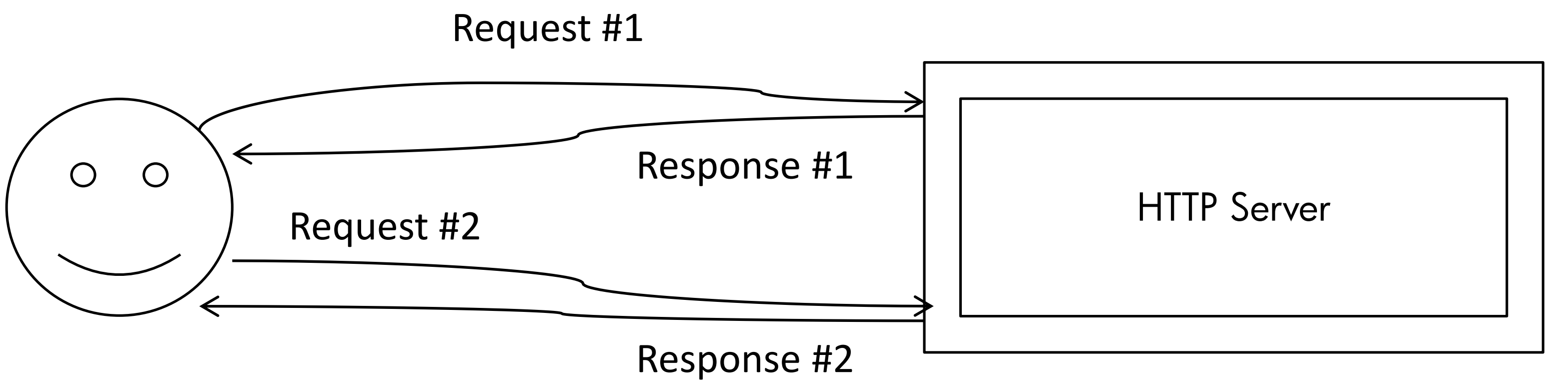
Your Date of Birth: (DDMMYYYY)

8-character Password:

HTTP Sessions

HTTP does not provide built in session support

- Requests are basically independent of each other



However, many server side script (e.g. ASP, PHP, JSP) can provide server side session support by identifying HTTP client by their assigned *session ID or session tokens*

HTTP Session ID / token

There are a few ways to store session ID / token in client side

- Browser cookies (most common)
- URL parameters
  - Append the session ID to all the URL of the page
- Other client storage
  - Browser object store (e.g. Flash persistent storage)
  - HTML5 Web Store API

Some common session ID name:

- ASP.NET_SessionId
- .ASPXAUTH
- JSESSIONID
- PHPSESSIONID

Authentication Session

A type of session which is specifically for maintaining the state of authentication

The ID/token for authentication session should be handled as securely as the authentication factors (e.g. username & passwords)

- Avoid repeated authentication
  - the more times the secrets need to be inputted/transferred, the higher chance it might be leaked

- Limited usage compared to authentication factors
  - Often can't be used for re-authentication.
  - Limited timespans
  - Limited context

Session Management

Anyone with a valid session ID / token will be treated as the corresponding user by the application

- Obtaining the session ID / token can impersonate an user

The scheme of handling session ID / token, creating / destroying / renewing sessions objects and etc → session management

Involve both client side and server side mechanisms

Broken Authentication and Session Management

Application functions related to authentication and session management may not be implemented correctly, allowing attackers to compromise passwords, keys, session tokens, or exploit other implementation flaws to assume other users' identities

Common flaws:

- Session fixations
- predictable session IDs
- improper use of session tokens, etc

Session Fixation Attack

Instead of breaking / stealing the assigned session ID of a user, the attacker use various approach to "fix" a session ID to be used by a user

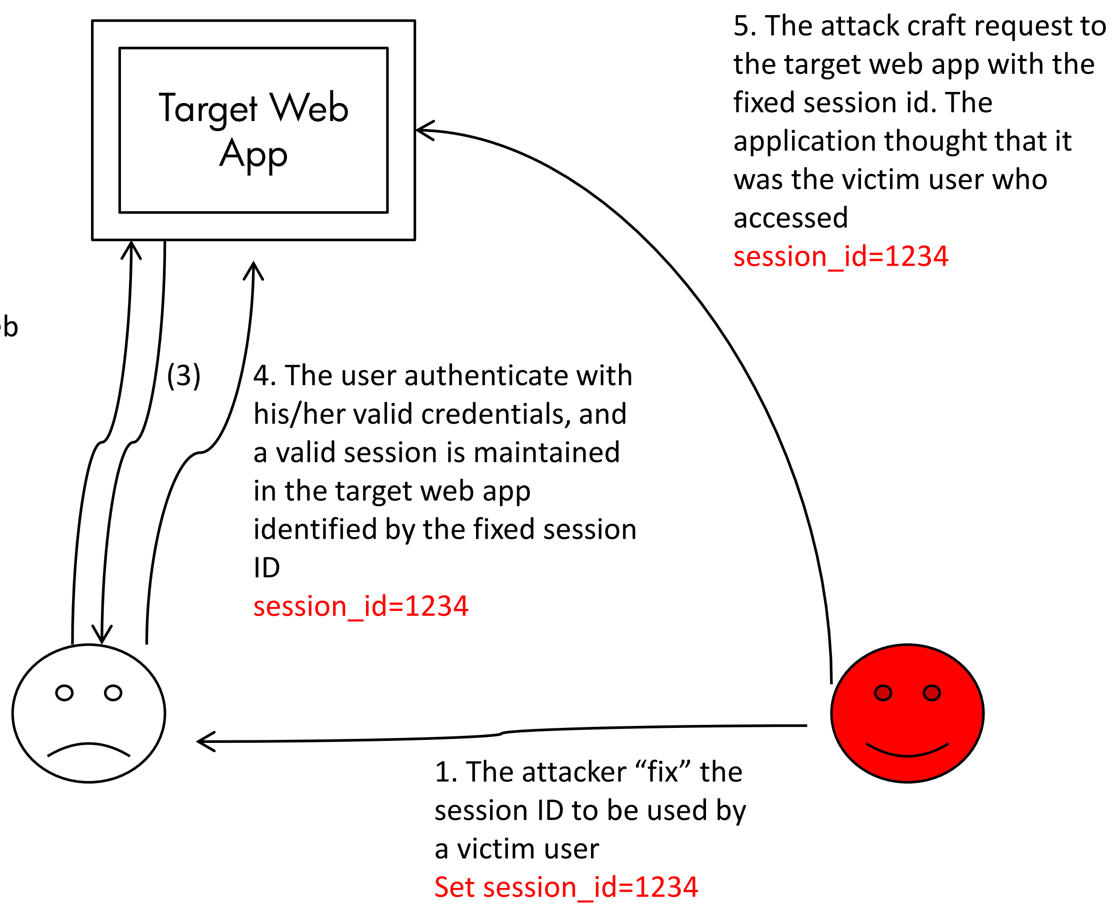
Potential ways of fixing a session ID

- XSS vulnerability in a page within the same cookie domain;
- file upload/modification vulnerability in any web server within the same cookie domain;
- The attacker has access to the web server within the same cookie domain;
- Compromised browser; or
- The attacker use the victim's browser to "touch" the target web application, so that a session ID is assigned to the user, and the attacker read and store the session ID

Prerequisites of a successful session attack

- Attacker is able to fix the session ID used by the user; and
- The session ID does not update upon successful authentication or security context change

Session Fixation Attack Illustration


Target Web App
5. The attack craft request to the target web app with the fixed session id. The application thought that it was the victim user who accessed
session_id=1234
2. The victim user access the target web app and supply the fixed session ID
session_id=1234
(3)
4. The user authenticate with his/her valid credentials, and a valid session is maintained in the target web app identified by the fixed session ID
session_id=1234
3. The server use the user supplied session ID and associate it with a newly created server session
session_id=1234
1. The attacker “fix” the session ID to be used by a victim user
Set session_id=1234

How to Prevent Session Fixation Attack?

Application should not associate user provided session ID, if not already exists, with a newly created session.

- Instead a newly generated session ID should always be used in new session

Upon security context change (e.g. switch of protocol, successful authentication, successful logout, enter/exist of sensitive area, the existing session ID should be discarded and a new session ID should be generated

- So that even an attacker can fix a session ID, it cannot be used to enter more sensitive area

Separate sensitive and non-sensitive content into different cookie domains

Consider adding more constraints in session checking, e.g. source IP address

- To increase the difficulties of session fixation attack

For really sensitive area, or with high classification, session ID should be changed regularly

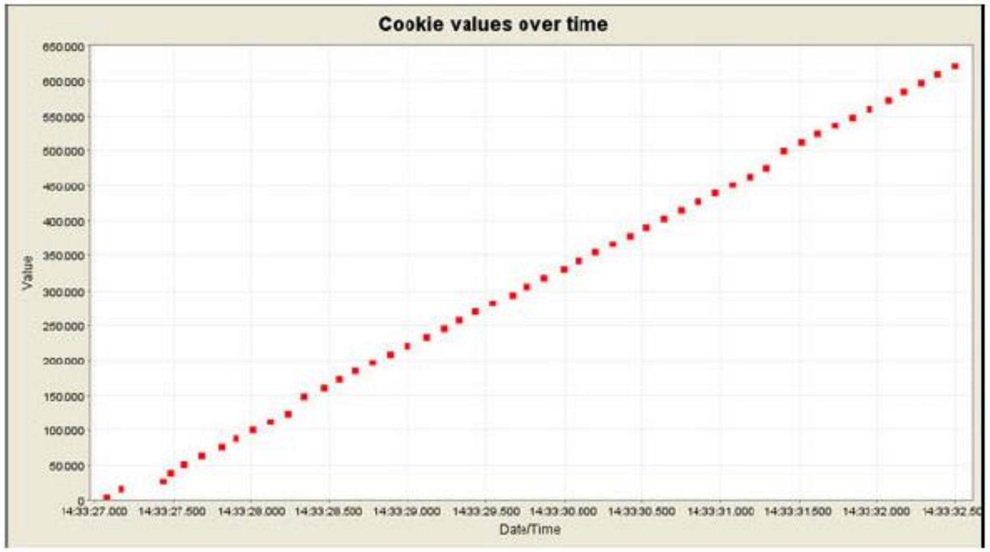
Predictable session ID

Some application generate session ID purely depending on sequential data source (e.g. sequence number, or time)

It allow attackers to easily guess the next brunches of session ID that may be used by users and thus impersonate them to perform actions on the application

Possible solutions:

1. Add more random factor to the session ID generation
2. Use hash function to process the sequential factor before using to generate session ID



Other Problems in Session Management / Authentication

Logical flaw in application causing session problems

- Especially when the session is “incomplete” – some set by one page and some set by another

Authentication process isn’t “atomic”. There may be some sessions at intermediate state of logon which may cause problem

Some authentication process may by mistake obtain the same information from different source (e.g. one from server session variable, another from user provided parameters), which allow attackers to create uncertain situation by assigning two different values to those different sources

Session object not destroyed after critical exceptions or user log out / timeout

Other Good Practices in Session Management / Authentication

- When a user logon at another place, invalidate the old session ID unless multiple logon is supported by design
- Let session time out after certain length of user idling.
- Destroy all the session objects properly after session ends.
- Avoid using multiple source of the same data during authentication
- Session cookies should have the `secure` and `HttpOnly` flag set

Lab

ATTACK ON SESSION MANAGMENT, AUTHENTICATION AND COUNTERMEASURES

OWASP Top 10 & Countermeasures

A4 – INSECURE DIRECT OBJECT REFERENCES

Insecure Direct Object References

A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data

Common direct object references:

- File name
- File path (directory transversal attack too?)
- Database ID
- User ID

Examples

- Original URL (with access control in application logic): http://vulnerable.site/download.php?file=my.doc
- Attack: http://vulnerable.site/download.php?file=not_my.doc

Insecure Direct Object References

Countermeasures

Avoid expose direct object reference to the users. User per user or per session indirect object references

- E.g. ID mapping created in the session or for specific user
- E.g. myupload[1], myupload[2].. Instead of file[2551], file[3022]…

Check whether the user has proper privilege before returning the object being referenced.

OWASP Top 10 & Countermeasures

A7 – MISSING FUNCTION LEVEL ACCESS CONTROL

Missing Function Level Access Control

Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.

Example: http://vulnerable.site/restricted_area/hidden_page.html

Missing Function Level Access Control

Countermeasures

The application should have a consistent and easy to analyze authorization module that is invoked from all of your business functions.

The enforcement mechanism(s) should deny all access by default, requiring explicit grants to specific roles for access to every function.

If the function is involved in a workflow, check to make sure the conditions are in the proper state to allow access.

OWASP Top 10 & Countermeasures

A5 – SECURITY MISCONFIGURATION

Security Misconfiguration

Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. All these settings should be defined, implemented, and maintained as many are not shipped with secure defaults.

For example,

- Directory listing enabled
- Default pages of servers
- Non-hardened services
- Overwriting of production configurations by less secure one from staging / test / development environment by mistake

Security Misconfiguration

Countermeasures

1. Document hardening process. Include a process to check against that after every deployment
2. Establish good patch management scheme which deploy security patches for OS, software and libraries in a timely manner
3. Maintain a strong application architecture that provides good separation and security between component
4. Regularly perform vulnerability scans on the server
5. Pay attention to the security configuration in staging, testing or development environment. Restrict access to them properly.

OWASP Top 10 & Countermeasures

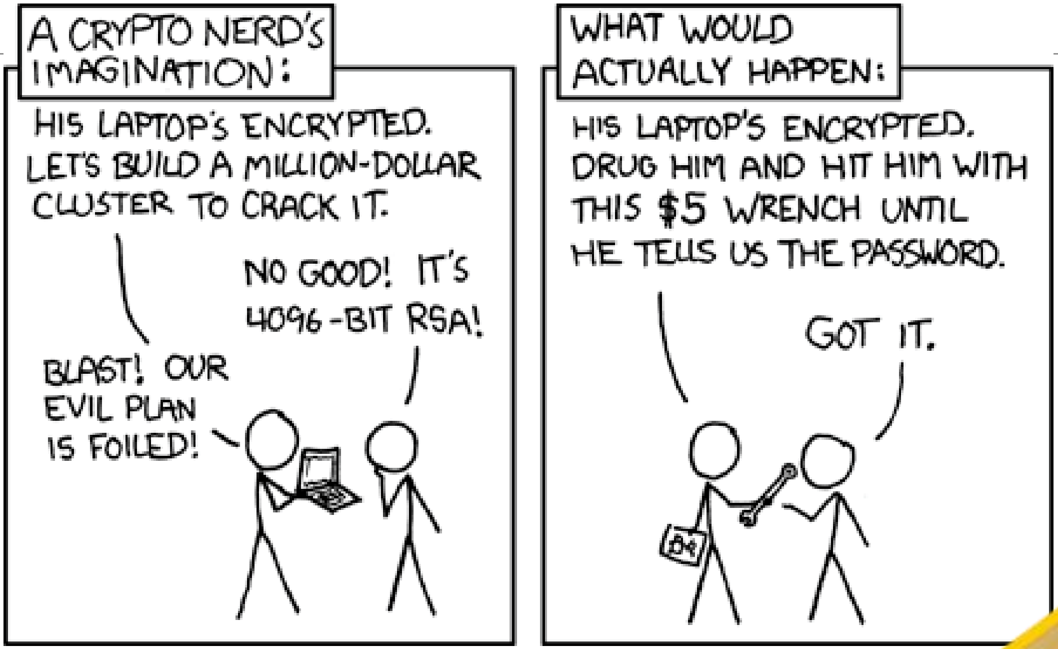
A6 – SENSITVE DATA EXPOSURE

Sensitive Data Exposure

Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.

Common reasons

- Sensitive Data stored in clear text
- Sensitive Data transmitted in clear text
- Old/Weak cryptographic algorithms used
- Weak crypto keys generated or improper key management


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

Source: http://xkcd.com/538/

Sensitive Data Exposure

Countermeasures

1. Encrypt all sensitive data at rest and in transit.
2. Do not store sensitive data unnecessarily. Discard it as soon as possible.
3. Ensure strong standard algorithms and strong keys are used, and proper key management is in place.
4. Ensure passwords are stored with an algorithm specifically designed for password protection
5. Disable autocomplete on forms collecting sensitive data and disable caching for pages that contain sensitive data.

OWASP Top 10 & Countermeasures

A9 – USING COMPONENTS WITH KNOWN VULNERABILITIES

Using Components with Known Vulnerabilities

Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.

Problem

- Vulnerability reports for commercial or open source software do not always specify exactly which versions of a component are vulnerable in a standard, searchable way.

Using Components with Known Vulnerabilities

Countermeasures

Upgrading the components to new versions whenever possible

Software projects should have a process in place to:

- Identify all components and the versions you are using, including all dependencies.
- Monitor the security of these components in public databases, project mailing lists, and security mailing lists, and keep them up to date.
- Establish security policies governing component use, such as requiring certain software development practices, passing security tests, and acceptable licenses.
- Where appropriate, consider adding security wrappers around components to disable unused functionality and/ or secure weak or vulnerable aspects of the component.

OWASP Top 10 & Countermeasures

A10 – UNVALIDATED REDIRECTS AND FORWARD

Unvalidated Redirects and Forwards

Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

Common problems:

- http://vulnerable.site/redirect.php?dest=/folder/page.html
- During redirection, file content may be returned bypassing privilege checks

Unvalidated Redirects and Forwards

Countermeasures

Minimize the number of redirects and forwards

Avoid using user provided content (e.g. URL parameters, cookies, for data) in building the destination. It can be a good idea to store destination and reference by ID as well.

Validate and sanitize any user provide content if they are necessary

Ensure proper privilege checking is performed before redirection and forwards

Other Common Web App Vulnerabilities

OWASP 2007 Top 10 – A3 Malicious File Execution

Problem

- Attacker may be able to load malicious file to the remote web server and execute it

Major causes

- Attacker is able to upload arbitrary files or modify to the remote web server
- Attacker may be able to execute commands on the remote web server to obtain malicious files from the Internet

Countermeasures

- Implement strict file upload type control:
  - E.g. by filename + by file type
- Disable unnecessary API on the remote web server
- Install Anti-virus software to the servers
- Consider mounting the file upload storage as non-executable

Mobile attack labs

Mobile Attack Labs

Mobile security attacks

Crack mobile apps (e.g. Android apps)

Mobile Security

Hack Brief: The Android Text Attack

A serious flaw in Android operating systems uncovered by a researchers at Zimperium zLabs could be the worst ever reported for Android devices. (27 Jul 2015)

A hacker could gain access to an android device by sending a text message containing a malware-infected media attachment. It compromise Android phone even user did not open the text or view the media in some cases.

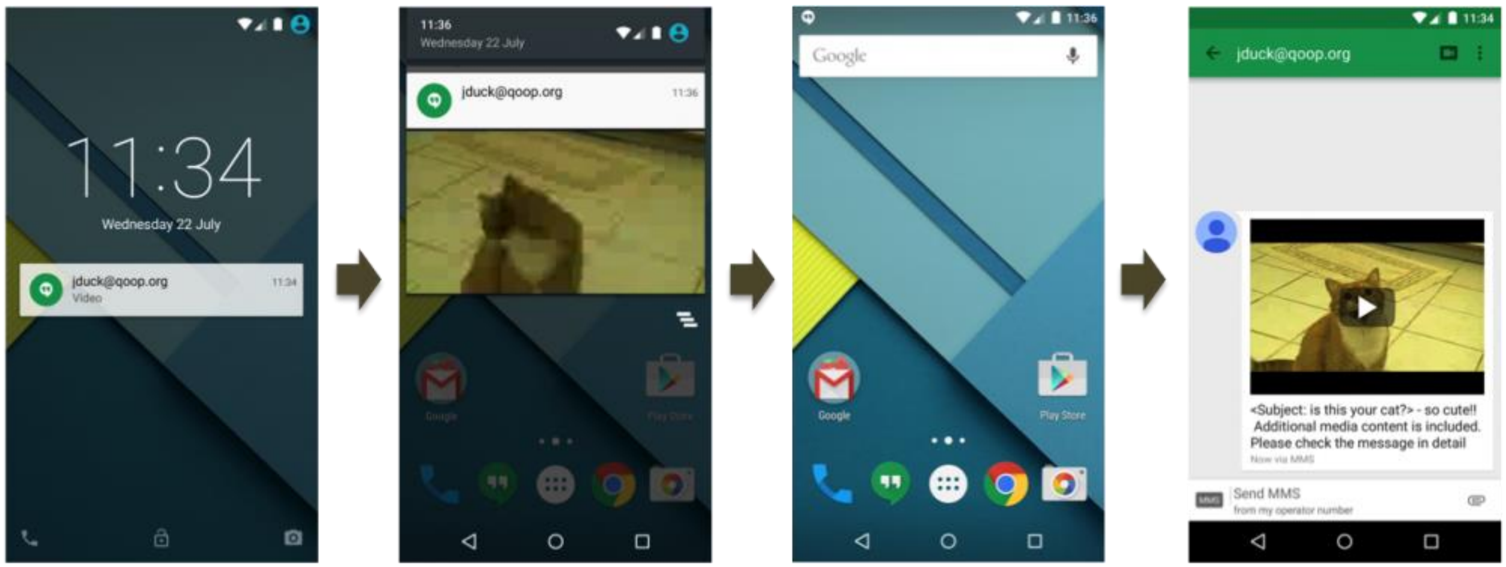
The remote MMS attack makes use of six critical vulnerabilities in Android operating systems 2.2 or later

Some chat client may not notify user after receiving the text message. For instance, hangouts will decipher the content before notifying the user

An example of exploitation is Stagefright remote code execution exploit that attack the stagefright media library

StageFright Android attack

Attackers only need to attack at selected mobile number with a specially crafted media file delivered via MMS



<Subject: is this your cat?> - so cute!! Additional media content is included. Please check the message in detail

An MMS was received by Hangouts. At this point, an attacker may have already executed arbitrary code.

MMS notifications show a preview, which triggers the vulnerable code.

After unlocking the screen, no effects are apparent.

Viewing the MMS message triggers the vulnerable code again. Touching the video or rotating the screen triggers the vulnerable code again and again.

Mobile Security incidents and issues

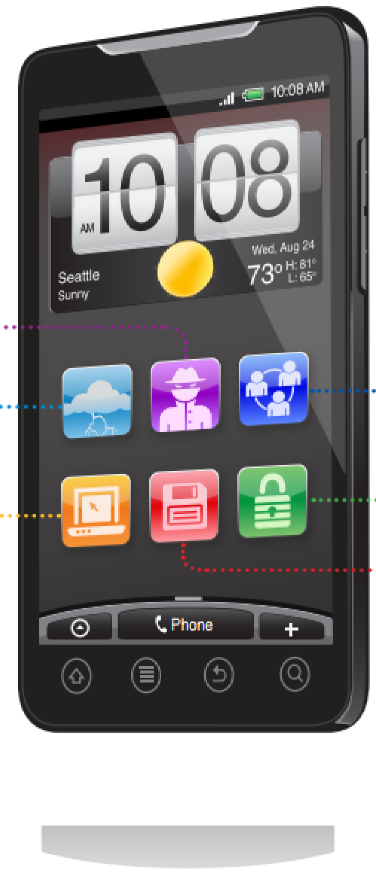
TOP THREATS TO MOBILE (BY CLOUD SECURITY ALLIANCE, 2012)

- 1. Data loss from lost, stolen or decommissioned devices.
- 2. Information-stealing mobile malware.
- 3. Data loss and data leakage through poorly written third-party apps.
- 4. Vulnerabilities within devices, OS, design and third-party applications.
- 5. Unsecured WiFi, network access and rogue access points.
- 6. Unsecured or rogue marketplaces.
- 7. Insufficient management tools, capabilities and access to APIs (includes personas).
- 8. NFC and proximity-based hacking.

TOP 10 OWASP MOBILE RISKS – FINAL LIST 2014

- M1: Weak Server Side Controls
- M2: Insecure Data Storage
- M3: Insufficient Transport Layer Protection
- M4: Unintended Data Leakage
- M5: Poor Authorization and Authentication
- M6: Broken Cryptography
- M7: Client Side Injection
- M8: Security Decisions Via Untrusted Inputs
- M9: Improper Session Handling
- M10: Lack of Binary Protections

Security Threats to Mobile Device (Surface)



Web- & Network-based Attacks

Launched by malicious websites or compromised legitimate sites

Attacking site exploits device's browser

Attempts to install malware or steal confidential data that flows through browser

Malware

Includes traditional computer viruses, computer worms and Trojan horse programs

Example: Ikee worm targeted iOS-based devices

Example: Pjapps enrolled infected Android devices in botnet

Social Engineering Attacks

Leverage social engineering to trick users

Attempts to get users to disclose sensitive info or install malware

Examples include phishing and targeted attacks

Resource Abuse

Attempt to misuse network, device or identity resources

Example: Sending spam from compromised devices

Example: Denial of service attacks using computing resources of compromised devices

Data Loss

Employee or hacker exfiltrates sensitive info from device or network

Can be unintentional or malicious

Remains biggest threat to mobile devices

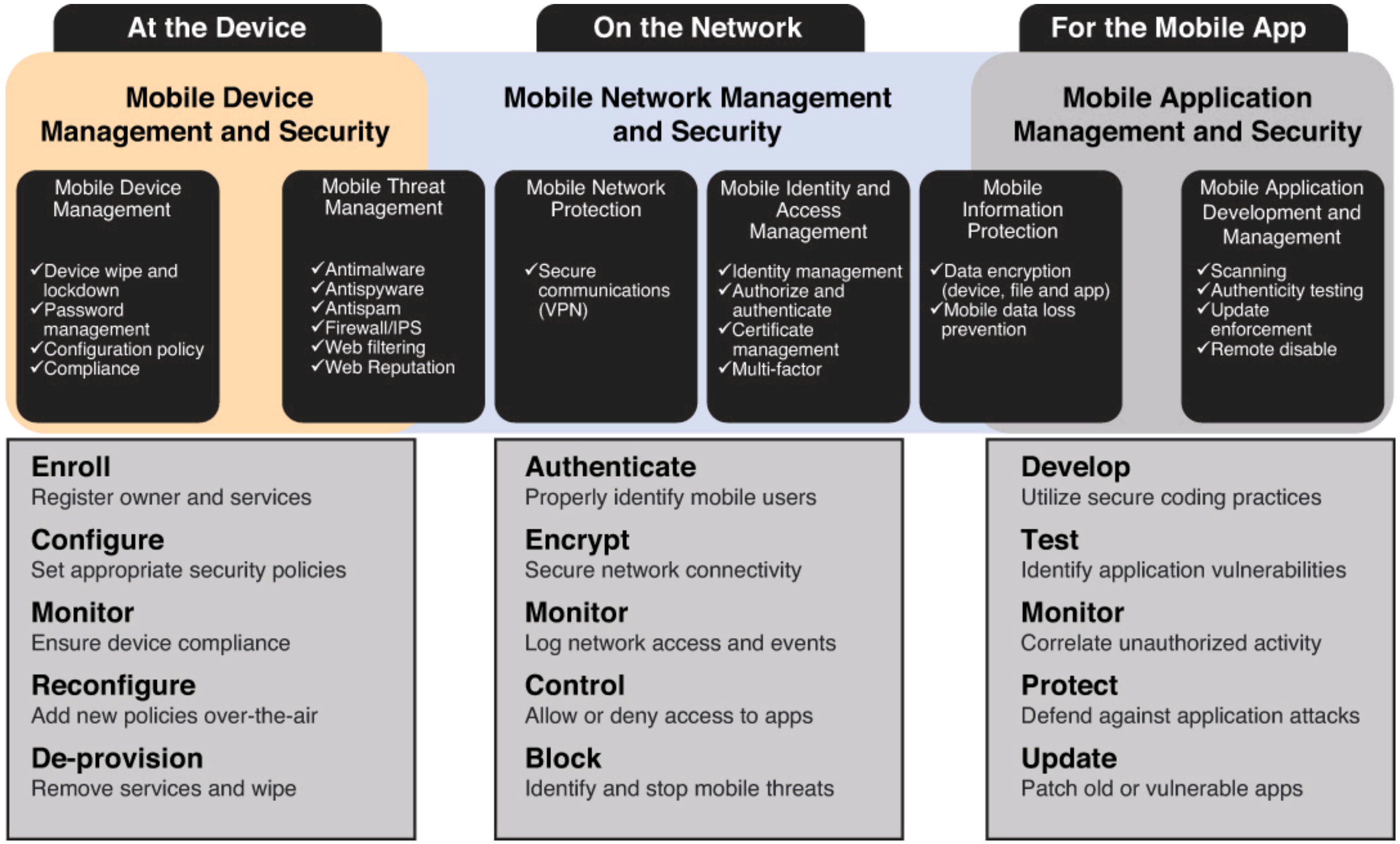
Data Integrity Threats

Attempts to corrupt or modify data

Purpose is to disrupt operations of an enterprise or for financial gain

Can also occur unintentionally

Mobile Security framework from Mobile Life Cycle



At the Device

Mobile Device Management and Security

Mobile Device Management
- ✓Device wipe and lockdown
- ✓Password management
- ✓Configuration policy
- ✓Compliance

Mobile Threat Management
- ✓Antimalware
- ✓Antispyware
- ✓Antispam
- ✓Firewall/IPS
- ✓Web filtering
- ✓Web Reputation

Enroll Register owner and services

Configure Set appropriate security policies

Monitor Ensure device compliance

Reconfigure Add new policies over-the-air

De-provision Remove services and wipe

On the Network

Mobile Network Management and Security

Mobile Network Protection
- ✓Secure communications (VPN)

Mobile Identity and Access Management
- ✓Identity management
- ✓Authorize and authenticate
- ✓Certificate management
- ✓Multi-factor

Authenticate Properly identify mobile users

Encrypt Secure network connectivity

Monitor Log network access and events

Control Allow or deny access to apps

Block Identify and stop mobile threats

For the Mobile App

Mobile Application Management and Security

Mobile Information Protection
- ✓Data encryption (device, file and app)
- ✓Mobile data loss prevention

Mobile Application Development and Management
- ✓Scanning
- ✓Authenticity testing
- ✓Update enforcement
- ✓Remote disable

Develop Utilize secure coding practices

Test Identify application vulnerabilities

Monitor Correlate unauthorized activity

Protect Defend against application attacks

Update Patch old or vulnerable apps

Source: Mobile Strategy: How Your Company Can Win by Embracing Mobile Technologies

Holistic view and planning

Business requirement

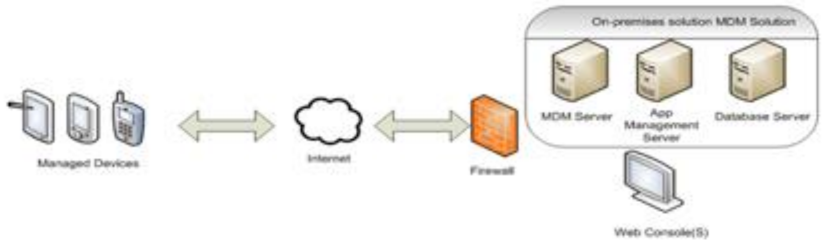
Infrastructure security

Mobile devices security

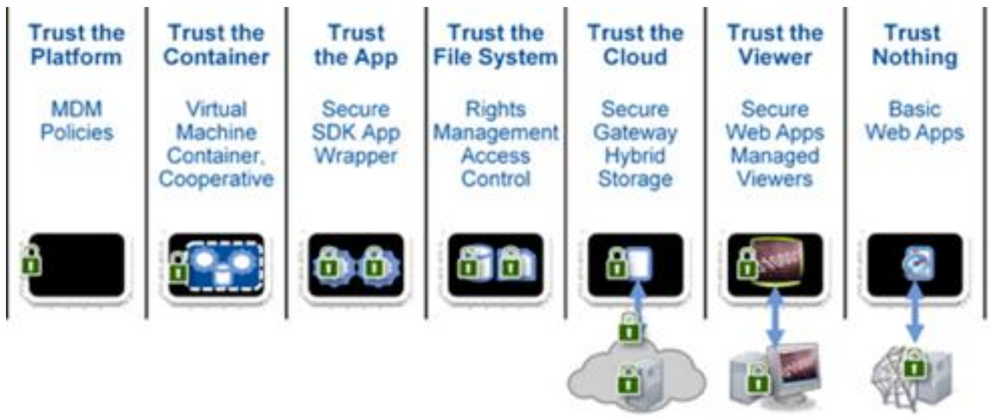
Enterprise Mobile Management

Security Testing

What else?



Source: From OGCIO's Practice Guide on Deploying Mobile Device Management



Source: Introducing the Spectrum of Trust for Mobile Enterprise Design, Gartner, Published: 4 April 2014 ID:G00261172

Mobile needs development consideration

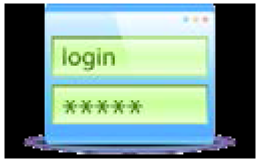
Integrated Development Environment needed?

Development language

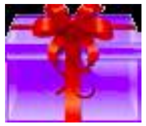
Apps development scheme

- Native, Hybrid, Web?
- Web Access, App Wrapping, Mobile SDK?

Web Access



App Wrapping

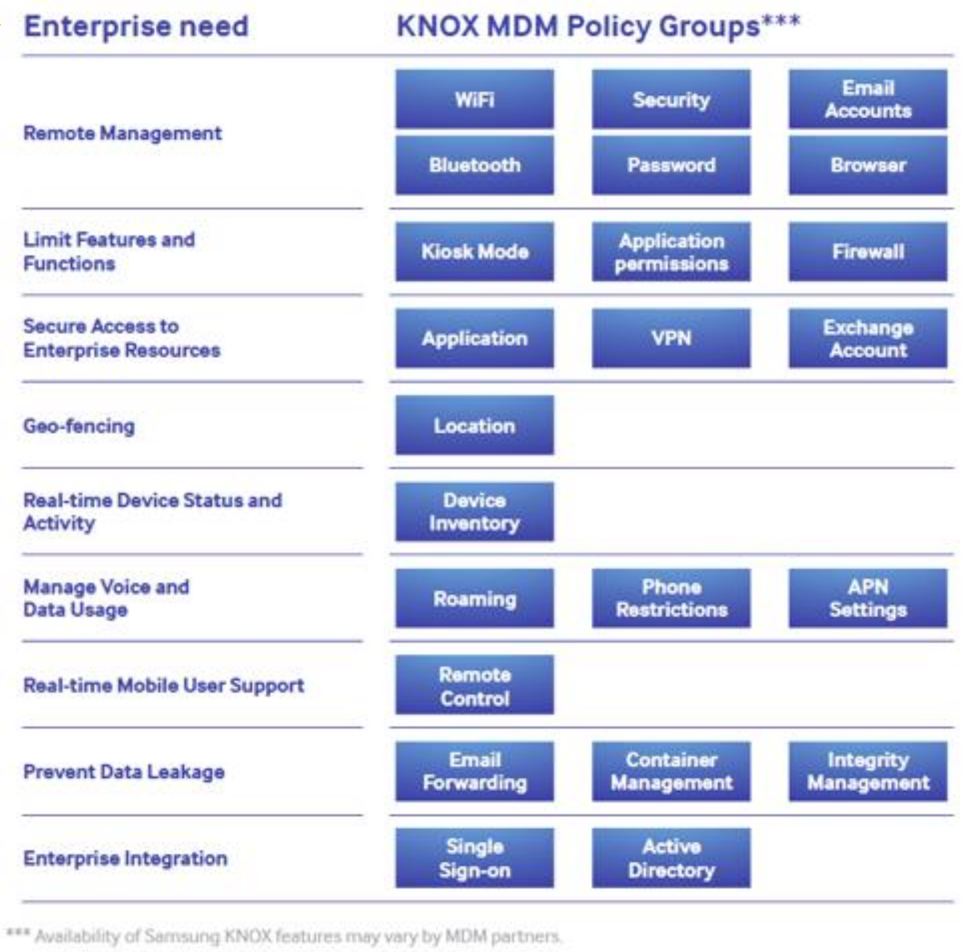


Mobile SDK

Mobile Devices Hardening

Mobile Devices dependent hardening control

Different brand has its scheme of hardening



Enterprise need	KNOX MDM Policy Groups***		
Remote Management	WiFi	Security	Email Accounts
	Bluetooth	Password	Browser
Limit Features and Functions	Kiosk Mode	Application permissions	Firewall
Secure Access to Enterprise Resources	Application	VPN	Exchange Account
Geo-fencing	Location		
Real-time Device Status and Activity	Device Inventory		
Manage Voice and Data Usage	Roaming	Phone Restrictions	APN Settings
Real-time Mobile User Support	Remote Control		
Prevent Data Leakage	Email Forwarding	Container Management	Integrity Management
Enterprise Integration	Single Sign-on	Active Directory	

*** Availability of Samsung KNOX features may vary by MDM partners.

Define the roadmap towards mature mobile security

Types of mobile apps to be supported?

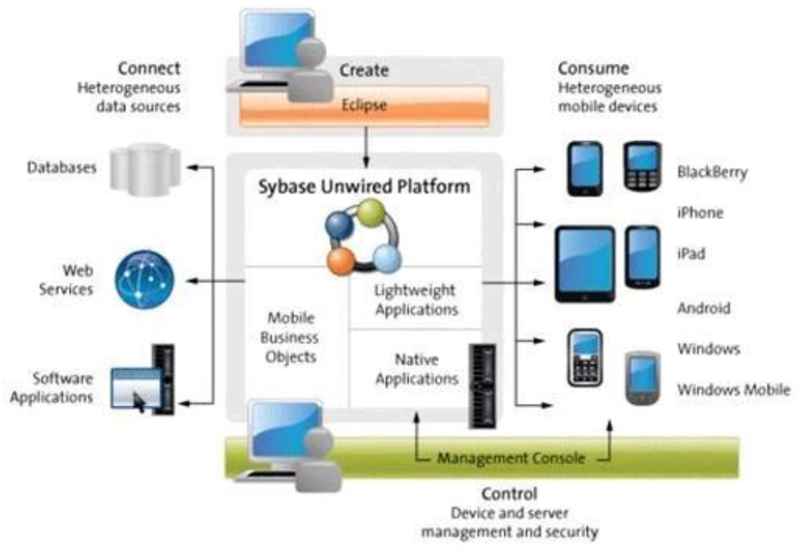
- Web Apps
- Custom Apps
- Commercial off the shelf Apps

Security Solutions required

- Access Management
- Federation Identity Management
- API Security Management
- SDK for Advanced Authentication, SS(

Others

- Enterprise App Store?
- Notification Services
- Adaptation
- Data channel protection



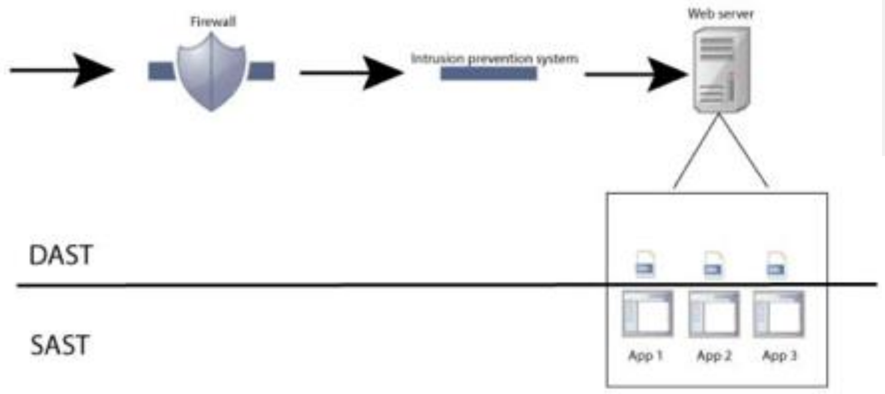
Source: Slideshow.Techworld.com for Sybase

Security Testing

Design Review

Static application security testing (SAST)

Dynamic application security testing (DAST)

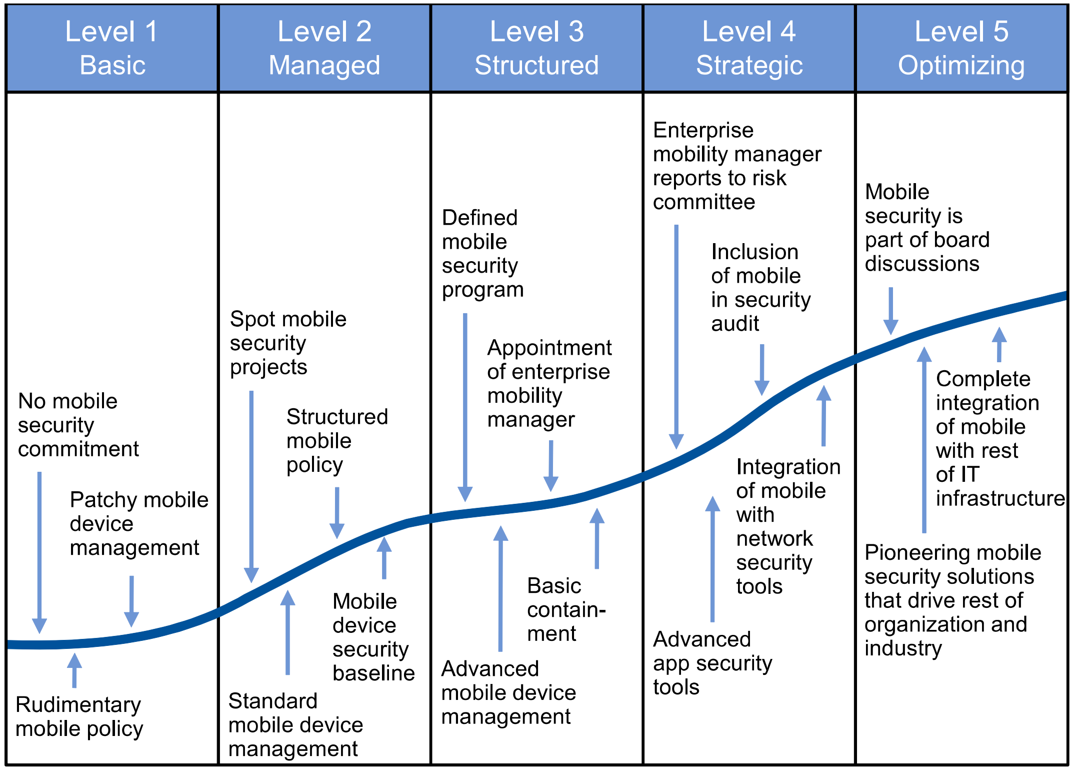


Compatibility Testing
Functional Testing
Usability Testing
Data Exchange & Synchronization Testing
Integration & System Testing
Regression Testing
MOBILE Application Testing

Security Testing?

http://pic.dhe.ibm.com/

Define the roadmap towards mature mobile security


Level 1
Basic
Level 2
Managed
Level 3
Structured
Level 4
Strategic
Level 5
Optimizing
No mobile security commitment
Patchy mobile device management
Rudimentary mobile policy
Spot mobile security projects
Structured mobile policy
Mobile device security baseline
Standard mobile device management
Defined mobile security program
Appointment of enterprise mobility manager
Basic contain-ment
Advanced mobile device management
Enterprise mobility manager reports to risk committee
Inclusion of mobile in security audit
Integration of mobile with network security tools
Advanced app security tools
Mobile security is part of board discussions
Complete integration of mobile with rest of IT infrastructure
Pioneering mobile security solutions that drive rest of organization and industry

Source: Gartner (January 2014)

A Guide to Gartner's Enterprise Mobile Security Self-Assessment
Published: 13 January 2014, G00246739

Reference Books

Related Content	Book	Chapter
W8, 9: Web Security	Guide to Computer Network Security (2015)	Chapter 6: Scripting and Security in Computer Networks
Web Security	OWASP web site	OWASP Top 10
Web Authentication and Authorization attack	Hacking Web Applications Exposed 3	Chapter 4: Attacking Web Authentication, Chapter 5: Attacking Web Authorization
Mobile Security	OWASP web site	Top 10 OWASP Mobile Risks – Final List 2014
Vulnerable mobile web sites	OWASP web site	https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#M-Tools