Widget security model based on MIDP and Web Application based on a security model with TLS/SSL and XMLDsig

Claes Nilsson
Technology Area Group Leader
Web Browsing

Marcus Liwell
Technology Area Group Leader
Security and DRM

Differences of the security approaches in PCs and mobile devices

- Established as open environment – users are used to install and uninstall program
- User aware of and accepts security problems – firewalls and virus protection generally used

- Currently user's awareness of security issues are low Mobile devices are considered as "secure"
- Mobiles are getting more and more based on open environments – user's awareness of security issues will increase

GOALS

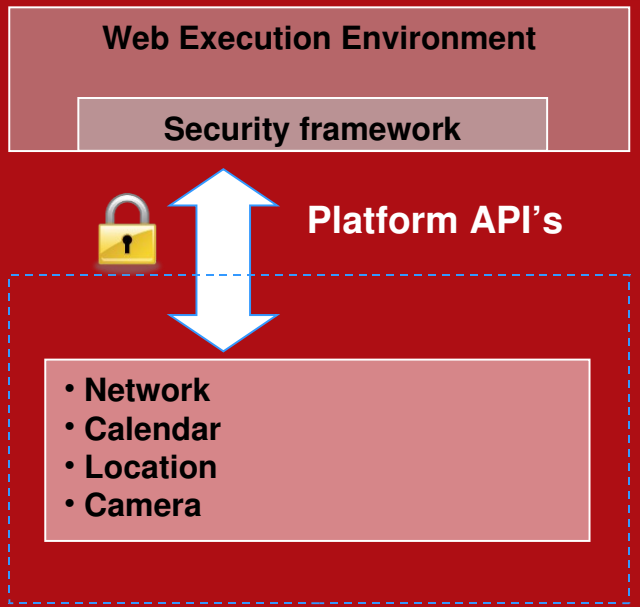
- Create a mobile security framework that is both reliable and "user friendly"
- Preserve the user's view that mobile devices are trusted and secure

Sony Ericsson

Difference in security approaches between Web Applications and Widgets

Web App

- Find new services by "browsing around"
- No installation process
- Invoked by the browser by a URL, bookmark or shortcut
- Content resides behind several URLs and is often dynamically generated



Widget

- Find new services in widget gallery
- Downloaded and installed on the device
- Started from standby screen or shortcuts
- Executable content, i.e. scripts, contained within a single package

Sony Ericsson

Widgets

Proposed security solution for Widgets

Base on MIDP security principles and improve

- Digital signing to authenticate the Widget creator and verify the integrity of the Widget. Is independent of the delivery solution, i.e. the server the Widget is fetched from
- Protection domain concept to create a default policy to ease the IOT burden on application developers
- To be added: Mechanism to dynamically define API permission policies for the different domains

Focus on usability

- Avoid pop-ups when the Widget is under execution. MIDP implementations often launch pop-ups during execution that are difficult to relate to the current user context.
- Use requested API permissions in the Widget manifest to execute user dialogues asking for permissions at installation time
- User may have the possibility to impact when user dialogues asking for permissions are executed

Untrusted and trusted domains

Untrusted Widgets

- Origin and integrity of the Widget can NOT be trusted by the device
- Must execute in the untrusted domain using a restricted environment:
  - Access to un-sensitive APIs, e.g. UI, Playback of sound, vibration etc
  - Access to some protected APIs with explicit user confirmation, e.g. http and https

Trusted Widgets

- Digitally signed and verified
- Security model based on protection domains
- Each protection domain defines a set of permissions to authorize access to protected APIs or function groups

Permission policies for protection domains

API permission policies for the different protection domains,
e.g.3rd party, operator, manufacturer, are dynamically loaded into the device.

Requesting permissions at Widget installation

- At installation validate against the permission policies for the Widget's protection domain
- Hierarchy of APIs to same functionali "use best available in device"
- User may be allowed to configure us dialogues. For example, impact if location API is always available or if the user should confirm every time
- If API is not allowed for the Widget's protection domain, it shall be up to the Widget to decide if it still shall be installed or not.

Web Applications

Proposed security solution for Web App

Verify origin and integrity of Web Application

- Transport layer security (TLS/SSL)
  Authenticates the server from which the page was loaded and achieves integrity protection during the transport from server to client
- XMLDsig of page or parts of the page
  Authenticate the content creator if needed (some sensitive APIs)

Focus on usability

- Avoid irritating pop-ups
- Consider asking for user permissions when the page is loaded

Permission policies for protection domains

Similar protection domain concept as for Widget also for Web Applications

Possible security levels:

- **No secure transport and signing:**
  Only “harmless” APIs can be accessed (battery level, beep, vibration etc)
- **Secure transport:**
  Medium sensitive APIs can be accessed (Positioning, Camera, Call Handling etc)
- **Secure transport and signing:**
  Highly sensitive APIs can be accessed (SIM, DRM etc)

- **Transport layer security (TLS/SSL)**
  Authenticates the server to identify which API access policy shall be used
- **Transport layer security and XML Digital signing**
  Combination of transport layer security and digital signing gives the highest security level and ensures end to end security. Cross server applications will not cause illegal use of sensitive APIs by a Web application hosted on a non trusted server when it is accessed through a trusted server.


Thank you
Sony Ericsson