Practical Security Measures for Exposing Enterprise Services to Mobile Applications

Greg Kliewer | Solutions Architect | Layer 7 Technologies | gkliewer@layer7.com
Contents

Apps → Web Apps → Services → Mobile

Services → Mobile = New Risks

Mitigations Strategies
Contents

Apps → Web Apps → Services → Mobile

Services → Mobile = New Risks

Mitigations Strategies
Chances are, you work for a going concern
It is likely that your organization rolled out **Web Apps** a long time ago.
And your organization probably, to some extent, adopted **Web Services** over the last decade, in order to facilitate integration between enterprise applications.
And now there are pressures to provide user interfaces into the same enterprise applications and services from new applications, deployed to **Mobile Platforms**.
BUT mobile platforms are different than browsers

**Browser**
- “Thin” clients
- User experience is delivered almost entirely by the server
- Consume “pages” already marked up for human consumption

**Mobile**
- “Thick” clients
- User experience is delivered by the app (servers are just there for “resources”)
- Consume APIs designed for machines only
So organizations are looking to leverage their enterprise services
And they are building / buying bridges from services to mobile

<table>
<thead>
<tr>
<th>Web Services</th>
<th>Mobile Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transport</td>
<td></td>
</tr>
<tr>
<td>HTTP POST</td>
<td>HTTP REST</td>
</tr>
<tr>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>XML</td>
<td>JSON</td>
</tr>
<tr>
<td>Authentication</td>
<td></td>
</tr>
<tr>
<td>PKI, X.509, Kerberos, SAML</td>
<td>Trust-less (keys, tokens)</td>
</tr>
<tr>
<td>Confidentiality &amp; Integrity</td>
<td>XML DSIG / ENC WS-Security</td>
</tr>
</tbody>
</table>

Wait… do we have a problem here?

Compatible

Simple Transformation

© 2012 Layer 7 Technologies
Contents

Apps → Web Apps → Services → Mobile

Services → Mobile = New Risks

Mitigations Strategies
Risk 1: new entities abound

GET http://www.layer7.com/employee?id=270183

Who is making the call?
API Keys

“An application programing interface key (API key) is a code generated by websites that allow users to access their application programming interface. API keys are used to track how the API is being used in order to prevent malicious use or abuse of the terms of service.

API keys are based on the UUID system to ensure they will be unique to each user.”

(Source: wikipedia http://en.wikipedia.org/wiki/Application_programming_interface_key)
Risk 1: new entities abound

GET http://www.layer7.com/employee?id=270183
&APIKey=15458617-7813-4a37-94ac-a8e6da6f6405

A person?

Layer 7 HR

My Manager

Or an app?

Or a device?

15458617-7813-4a37-94ac-a8e6da6f6405
The Identity Profile

Increasingly we need to move toward large number of claims (multiple identity profile)

- **Identities**
  - userID
  - appId
  - deviceID

- **Attributes**
  - Roles
  - Geo location
  - IP
  - User agent
  - Time of day
  - etc
The culture of the web:

“No way we’re implementing client-side certificates”

PROBLEM:
Most of the authentication schemes that we relied on for web services involved asymmetric crypto with consumers of services (clients)
- SSL / TLS Mutual Authentication
- XML DSIG / XML ENC
- WS-Security
- SAML
Risk 2: impracticality of PKI / trust-based solutions
Contents

Apps ➔ Web Apps ➔ Services ➔ Mobile

Services ➔ Mobile = New Risks

Mitigations Strategies
SSL for ALL THE THINGS!

The bridge MUST
• Terminate inbound connections (use new conn on back end)
• Require SSL for all connections
• Require strong cipher suites

Web devs won’t like this, but…
It is the battle you can win!
(as opposed to trying get them to adopt client side certs)

The overhead isn’t what it used to be
Use real tokens to establish / maintain sessions

- API Keys are okay for tracking how applications are using APIs / services, but they are not security tokens
- Use OAuth for:
  - Issuing session token tied to the user
  - Authenticating the client (e.g. device) as well as the user when necessary
- Analogous to web app cookies

For more on OAuth see:
http://oauth.net/

The token system MUST
- Practice good management (e.g. expiry, revocation)
- Treat tokens as shared secrets (hence, SSL all the time)
Leverage existing ACLs where possible

- If you already own the services you are developing mobile apps for, there is no need to over-engineer by adopting fancy 3-legged grant types
  - Resource Owner Credentials grant type will work just fine and allow you to do warm-body AU/AZ the way you do today

Access Tokens
Clients / Secrets

The resource owner (user) MUST be authenticated and authorized for access to the resource requested.
In Summary:
Summary

- There are new complications and risks associated with exposing enterprise (web) services to mobile applications
  - New entities to identify and authorize (e.g. applications and clients)
  - Impracticality of establishing trust in familiar ways (i.e. with certificate distribution)

- BUT there is no need to panic

- There are practical steps to take to reduce security risks while bridging between the enterprise service and mobile worlds
  - Token systems can be placed in front of existing services to provide access control that works much like browser-cookie systems
  - Tokens can be kept safe over their limited life-spans using tried-and-true SSL
  - Token systems provide grant types designed to work with incumbent enterprise identity and access management solutions
THANK YOU!

Questions?