Policy-based WRT Security

Access to device APIs

Problem space

- Allow access to device services from Web content executing in WRT
- Incrementally increasing requirements:
  - Installed widgets → Arbitrary web pages
  - Untrusted content → Trusted content
  - Single installed policy per device → Policy per runtime
  - New device services
- Assumptions and Constraints
  - Web engine core is a given
    - We are addressing sandbox limitations separately
  - Web engine, service API implementations, launchers are trusted
    - For Nokia this is currently addressed by Symbian platform security
  - Executing content may have no usable trust attributes
    - Moving towards support of signed content

NOKIA

Three Approaches

- Security implicit in service implementation:
  - Operations require user interaction via “usual and customary” UI (e.g. show camera viewfinder and require physical shutter press)
- Untrusted content access via user prompting:
  - Usability is not great: e.g., user granted blanket access by accident and wants to revoke it.
- Trusted content access without prompting:
  - Signed code, trusted origin

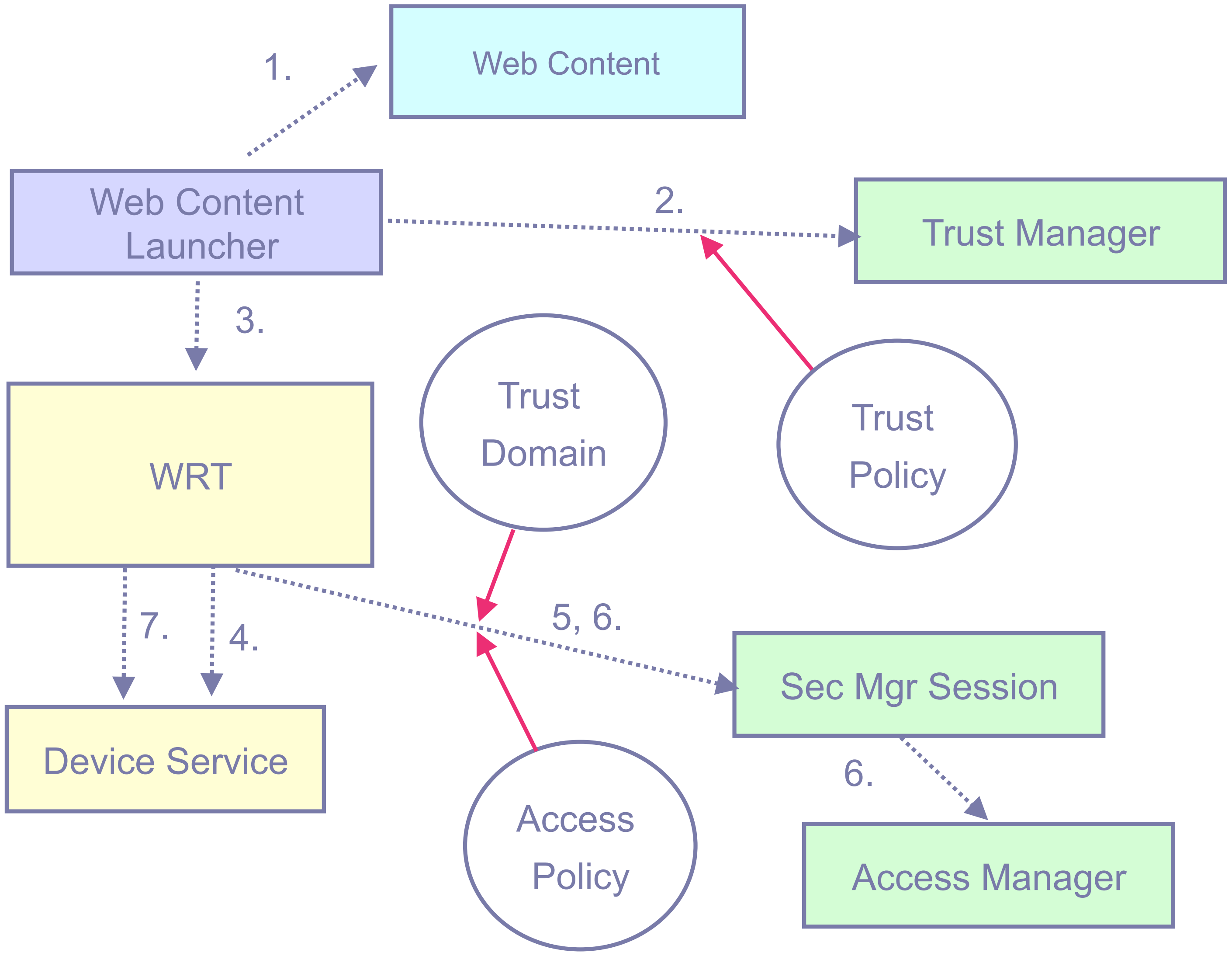
Solution

- Policy-based access control engine
  - Model borrows elements from MIDP
  - Policy format is XACML-like
- Role for standards
  - Policy format (tweak XACML)
  - Capability semantics, which are not defined by policy model
- We will open-source our solution

Policies

- Policy-based model provides the flexibility to meet our changiing set of requirements:
  - Get the policy model right to start
  - Implement what we can/need to
- Appropriate policies vary:
  - Depend on quality of Web engine (e.g. the sandboxing model), belief about the reliablity of trust attributes, API access model, etc.
- Trust Policy
  - Map code attributes to trust domains
  - Code attributes extensible/attribute handlers
- Access Control Policy
  - Map trust domains to capabilities
  - Must support both user-queried and non-queried access
    - Conditional capabilities

Usage Sample



NOKIA

Sample Sequence

1. Get content attributes
2. Request trust domain from trust manager
   - Trust policy maps code attributes to trust domain
3. Launch content with trust domain associated
4. Instantiate device service API
   - Get required capabilities from service
5. Create session with access manager
   - Embodies trust domain + access policy
   - Access policies maps trust domain to capabilities
6. Request access decision for required capabilities
7. Access service operation

Trust Policy

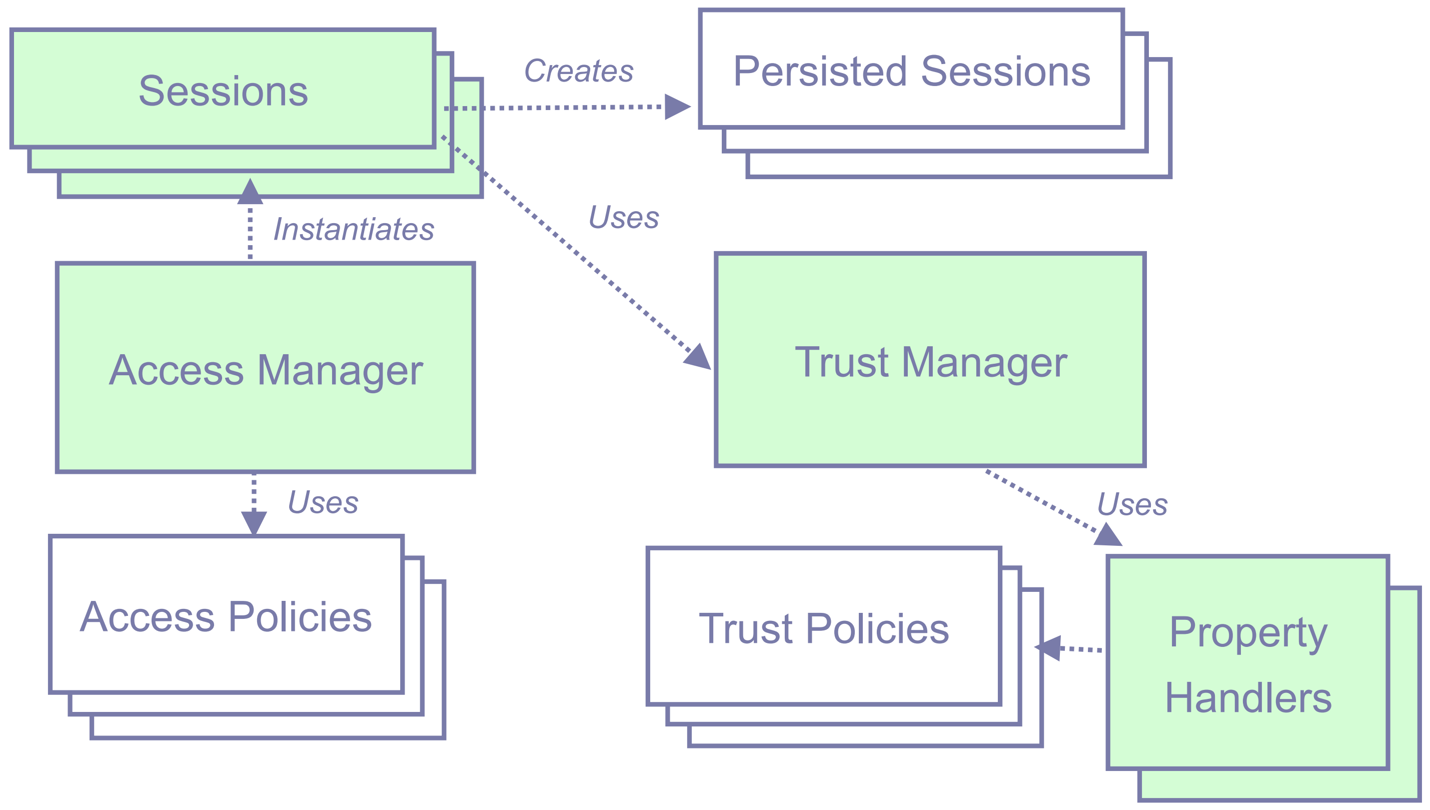
```
<trustpolicy>
  <defaultdomain name = "Untrusted"/>
  <domain name="NokiaService">
    <origin url = "http://www.nokia.com/services"/>
  </domain>
  <domain name="NokiaPublic">
    <origin url = "http://www.nokia.com"/>
  </domain>
</trustpolicy>
```

Access Policy

```
 <accesspolicy>
   <domain name="Untrusted">
    <!-- always granted capabilities for this domain -->
    <capability name="UserDataGroup"/>
    <capability name="NetworkGroup"/>
    <!--  user-grantable capabilities for this domain -->
    <user>
      <defaultScope type="session" />
      <scope type="oneshot" />
      <scope type="permanent" />
      <capability name="DeviceResourcesGroup"/>
      <capability name="Location"/>
    </user>
  </domain>
</accesspolicy>
```

NOKIA

Security Manager Components


Sessions
Creates
Persisted Sessions
Instantiates
Uses
Access Manager
Trust Manager
Uses
Uses
Access Policies
Trust Policies
Property Handlers