Utilising Component Composition for Secure Ubiquitous Computing

2nd UK-Ubinet Workshop

5th – 7th May 2004

Dr. David Llewellyn-Jones, Prof. Madjid Merabti, Dr. Qi Shi, Dr. Bob Askwith

School of Computing and Mathematical Statistics
Liverpool John Moores University
James Parsons Building
Byrom Street
Liverpool, L3 3AF, UK

{D.Llewellyn-Jones, M.Merabti, Q.Shi, R.Askwith}@livjm.ac.uk
http://www.cms.livjm.ac.uk/PUCsec/
General Problem – Security in a Ubiquitous Environment

• Critical because:
  – Ubicomp must be a ‘mainstream’ technology
  – Users will not accept it if their fridge catches a virus

• Difficult because
  – Limited user interface
  – Users will not be security aware

• Extra difficult because
  – Everything is networked (fluid data environment)
  – Code will be moving between devices (fluid code environment)
    – Programmable networks
    – Software updates
    – Mobile agents
    – etc..
• Security solution must be:
  – Automated and transparent
  – Capitalise on the properties of a ubiquitous environment

• Determining the security properties of executable code
  – Certification
  – Proof Carrying Code
  – Direct Code Analysis
  – Model Checking
Application Components

- These methods can be improved by breaking an application into its constituent parts
  - Improves performance and resource usage
  - Allows distribution of resources
If we know the properties of individual components, what are the properties of the composed application?

The answer is not straightforward.
Many theoretical security properties exist, for example

- **Separable security properties**
  - NI: non-interference
    - Definition: An interface $E$ of a component is said to satisfy non-interference iff for any trace $t \in T_E$ there exists a trace $t' \in T_E$ such that
      $$t' \upharpoonright HIE = \emptyset \text{ and } t' \upharpoonright (LI_E \cup LO_E) = t \upharpoonright (LI_E \cup LO_E).$$
    - BSD: backward strict deletion
    - BSI: backward strict insertion

- **Other security properties**
  - CA: composable assurance

We aim to harness such results
Solution

• Harness theoretical results using a scripting technique
• Script
  – Written using XML application
  – Defines a set of component topologies
  – Components distinguished by their individual properties
• Add new scripts to include new results in the security analysis process
Example

• Composable Assurance

```xml
<configuration id="c1">
  <component id="c2">
    <input format="" cycle="disallow"/>
    <output format="CA*" cycle="disallow"/>
    <component>
      <input format="CA*" cycle="disallow"/>
      <output config="c2" format="CA*" cycle="disallow"/>
      <input format="" cycle="disallow"/>
      <output config="c2" format="" cycle="disallow"/>
    </component>
  </component>
  <component>
    <input format="id1*" cycle="disallow"/>
    <output format="external"/>
  </component>
</configuration>
```
• Non-Interference

```
<configuration id="c3">
  <component id="c4">
    <input format=""/>
    <output format="NI*"/>
    <component>
      <input format="NI*"/>
      <output config="c4" format="NI*"/>
      <output format=""/>
      <output format="external"/>
    </component>
  </component>
</configuration>
```
Framework

• System needs incorporating into a framework

• Collaborating with the Network Appliance Group to incorporate into a realistic prototype
  – Due to scripting techniques, composition analysis is easy and fast
  – The analysis stage is time consuming
    – aim to develop techniques to distribute analysis amongst multiple devices
The End

Thankyou for your time

www.cms.livjm.ac.uk/PUCsec