Providing Security and Privacy through Context and Policy Driven Device Control

Anupam Joshi

IBM India Research Labs
New Delhi, India 110070
(on leave from UMBC)
anupam.joshi@in.ibm.com

Abstract.
This paper describes an approach that integrates declarative policies, context, OS level device control, and existing carrier authentication techniques to control access to features of mobile devices for web applications. The approach builds upon a body of existing work and adapts it to the mobile handset scenario.

1 Introduction and Background

As the price point for computing comes down, cell phones are becoming increasingly more sophisticated. Over half the phones sold in developed markets, for instance, have cameras, according to CEAs website (http://www.ce.org/Press/CEA_Pubs/2088.asp). Such phones also typically have Bluetooth and IrDA interfaces, as well as removable local stores such as SD cards. Newer phones also have GPS or aGPS functionality, accelerometers, and in some cases even physiological monitoring sensors. Specialized devices also have RFID or Barcode readers added on. With a variety of such devices on the cell phone, applications are emerging that access these functionalities. Recent approaches like the application markets associated with Apple and Google Android have allowed third parties to write such apps as well. While some platforms restrict these device functionalities to only applications "installed" on the phone, it is expected that applications running in the browser (or other elements of the client side web infrastructure) would also need access to these device capabilities. Given the somewhat restrictive application distribution model adopted by some providers such as Apple, a web based approach is naturally attractive to independent, third party application developers. In this whitepaper, we outline a scheme that allows declarative specification of policies that are compiled into access/noaccess decisions based on the context of the user and the device and leverages the carrier's authentication (signon) process for trust.

2 Related Work

Most of the current systems for device level security work on a per application basis, typically at install. For instance in Android [1], each application typically gets its own uid. All permissions it needs are predeclared (and signed) by the developer. At install time, these permissions are either granted to the uid or not based, amongst others, on interactions with the user. Clearly, this is very coarse grained control. Ideally, the decision on whether or not to grant a request to access device features from an application

should be based on the context of the device and the user. For instance, a building may have a policy that cell phones with cameras are not allowed to take pictures inside, or that they must be in vibrate mode when in a meeting room. These are all instances of the context of the device, as captured in the policy of the space it happens to be in. Similar context dependant policies arise from the user's perspective. A user might want to make his Bluetooth device discoverable when at home or office, but not otherwise. Finally, the telco (the home carrier, or where the user is visiting) might have their own restrictions. For instance, they might require that 802.11 connections be only made with base stations of approved networks. We posit that a context dependent, policy driven security mechanism is best suited for device capability security. We argue that large scale acceptance of web based, third party applications will only happen with such a security system. While this is almost a given in the enterprise space, even casual users would be reluctant to run apps that could access device features to find out their location, take pictures without them knowing, or even use the proposed phone based (micro) payment systems!

2.1. Access Control

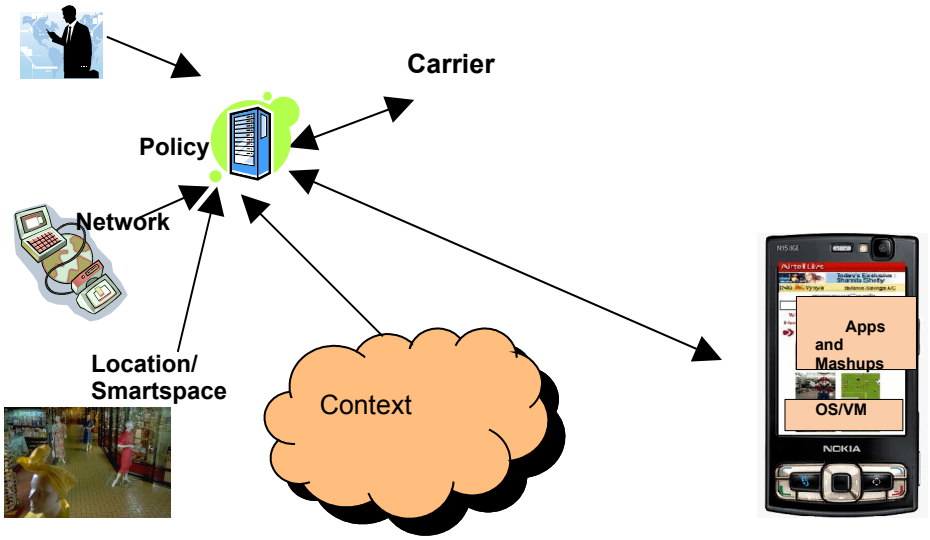
Access control is a very well studied topic in literature. We focus here on two works of immediate relevance. The first, by Jansen et al [2] suggests that devices such as PDAs be provided enterprise security policies (in XML) on a smart card. These policies would describe access/noaccess decisions for specific device features. The PDA would run a trusted kernel which would enforce these policies. The focus there was to provide policies, not base them on changing context. We extended this approach in our prior work [3] and showed that a pointer to such a policy could be sent over the 802.11 beacons, so that a "smart space" could point a device to its acceptable use policy. This work also suggested that policies be in richer declarative languages such as Rei[4]. Our proposed approach builds upon some of these ideas.

2.2. Policies

Policy driven security has been looked at by several academic research groups over the last decade. Ponder[5] is an OO approach to specifying policies. Researchers have also looked at using semantic web languages to declaratively specify policies, such as Rei[4], Rein[6] and KaOS[7]. There have also been efforts from industry and standards groups in this direction, especially in context of WS-Policy and EPAL[8].

3 Proposed Approach

Any solution to this problem needs three key components. *The first is an authentication and identity management system in which trust can be grounded*. We suggest that existing telco identity management and signon systems can be leveraged for this process. While such systems have known problems, the authentication of a mobile device to the telco's network is already a part of the system and so anything done of top will inherit its weaknesses (lack of mutual authentication, ability to clone subscriber identities, etc.). There are ongoing attempts in the research community to make the signon process more secure, including the use of biometrics.



A schematic of the Context and Policy driven access control system

We suggest that the normal signon process be modified to send back a pointer to the policy that the device should follow. The device (or rather, the SIM) should also host a restrictive user and carrier generated default policy that should be used to control the device in case a signon is not successful. Integrating the authentication/trust process with the carrier signon also allows us to leverage the cross network signons that many carries now deploy (e.g. GSM/GPRS/EDGE/UMTS and WiFi). The "base stations" of the various networks, as static, always connected entities, can then use standard PKI methods to establish trust amongst each other. The mobile device will then inherit this trust via delegation from its "home" telco.

The second key component is a policy system. There are a variety of candidate choices here, including the work being done by W3C researchers such as Weitzner and Kagal in the Policy Aware Web space. The exact choice of the system will be a function of the complexity/expressivity desired in the policy specification – at one end we have relatively simple XML based systems such as XACML. At the other we have systems expressive enough to capture arbitrary statements in first order logic. We believe that in the long term, more complex systems such as Rei [4] or Rein [6] are in fact needed to capture policies that are rich enough to express access decisions as a function of context. We also believe that grounding these systems in the semantic web languages such as RDFS or OWL allows a lot of domain specific knowledge that policies will need to be imported from ontologies being developed by semantic web researchers working with domain experts. For instance, the SOUPA ontology [10] defines a lot of terms that are needed to define policies in a smart space. Recent research has also shown that such languages are adequate to express well known security models such as Role Based Access Control [9].

As mentioned earlier, the policy will be provided to the mobile device by the network. Three elements contribute to this policy – the networks policy, the users policy, and (where applicable) the smart space's policy. This leads to the possibility of conflicts between these policies. However, most of the systems

described above provide mechanisms to handle this, such as priorities and modality preferences.

It is unlikely that in the near future, mobile handsets will have the capacity to reason over context events and expressive policies to decide access. So we suggest an approach where the policy system defines a set of context events it is interested in. These events could be user intent related (the user added a new meeting to her calendar), device status related (the location has changed from inside the building to outside the building), or network related (the device detects an inquiry page from a nearby Bluetooth device). The device sends updates about these context events to the policy server. The server reevaluates the access control decisions based on the new inputs, and sends back a (possibly) revised set of access triples (subject, object, permission). Incidentally, the same constraint holds at startup – when the device follows the policy pointer provided during signon, it should get access triples, not the actual policy statements.

Despite the fact that the policy is context dependent, it may not completely capture all possible situations. So at a very minimum, a user override provision should exist where a user can, if she wants, permit or deny access to a device feature no matter what the policy says. Some policy languages such as Rei permit such dynamic policy changes to be formally expressed via delegation mechanisms that are themselves constrained. For instance, the policy may permit the user to prohibit the GPS device to some application, but may not allow the user to override the networks policy on whether to associate the device with an untrusted WiFi/WiMAX network. Delegations, or more generally, speech acts which can dynamically modify a policy will be an important feature of the policy system for use in device feature control.

The third critical component of such a system will be the actual security mechanism on the device. This is highly dependant on the particular operating system. However, most OSs, including those used on mobile phones, share the idea of some generic i/o control mechanism such as ioctl. We can take the access triples provided by the policy server and create an in memory table that can be checked whenever an io request is made. Such an approach was demonstrated in [2] and [3]. Similar approaches can be used in VM based systems. Given that the number of devices to be controlled will be typically small, keeping this information in kernel memory will not provide any significant overhead. This assumes of course that the devices run some trusted kernel. This is not a very significant issue in the mobile phone domain since most phones run stock kernels installed by either the manufacturer or the network.

4 Discussion

In this position paper, we propose a system that leverages carrier authentication/signon process, and combines it with context dependent, policy driven security mechanisms to control access to device features from web applications. We also show that this can be combined with io control mechanisms in the OS kernel to enforce these policies.

We note that such mechanisms can also control access to the context related data on a device. For instance, consider a shopping scenario where a person entering a store has a shopping list. Sharing this information with a store might be related to what store it is, or perhaps what it offers (discounts) in exchange. Since a file can be treated as a device (and is, in most unix systems), the io control mechanisms can be used to share/show or hide a file based on context as well. Such a combination of integrated device and data control will be an important underlying mechanism for enabling sophisticated web based applications on mobile devices.

References

[1] Google Android Project, http://code.google.com/android/devel/security.html, Checked on Nov 4, 2008.
[2] Wayne Jansen, Tom Karygiannis, Michaela Iorga, Serban Gravila, and Vlad Korolev, Security Policy Management for Handheld Devices, The 2003 International Conference on Security and Management (SAM'03), June 2003.
[3] Anand Patwardhan, Vladimir Korolev, Lalana Kagal, and Anupam Joshi, Enforcing Policies in Pervasive Environments, International Conference on Mobile and Ubiquitous Systems: Networking and Services (Mobiquitous), August 2004.
[4] Lalana Kagal, Tim Finin, and Anupam Joshi, A Policy Language for A Pervasive Computing Environment, IEEE 4th International Workshop on Policies for Distributed Systems and Networks (Policy 2003), Jun 2003.
[5] Nicodemos Damianou, Naranker Dulay, Emil Lupu, Morris Sloman: The Ponder Policy Specification Language. IEEE Policy, 2001.
[6] Lalana Kagal, Tim Berners-Lee, Dan Connolly, and Daniel Weitzner, Using Semantic Web Technologies for Policy Management on the Web, 21st National Conference on Artificial Intelligence (AAAI), July 16 - 20, 2006.
[7] Andrzej Uszok, Jeffrey M. Bradshaw, Matt Johnson, Renia Jeffers, Austin Tate, Jeff Dalton, J. Stuart Aitken: KAoS Policy Management for Semantic Web Services. IEEE Intelligent Systems 19(4): 32-41 (2004)
[8] IBM: Enterprise Privacy Authorization Language (EPAL); Submission request to W3C, http://www.w3.org/Submission/EPAL/, November 2003.
[9] Ravi Sandhu, Edward Coyne, Hal Feinstein and Charles Youman, Role-Based Access Control Models, IEEE Computer, Volume 29, Number 2, February 1996.
[10] Harry Chen, Tim Finin, and Anupam Joshi, The SOUPA Ontology for Pervasive Computing, Ontologies for Agents: Theory and Experiences, 2005.