Security Policy Management for Handheld Devices

Wayne A. Jansen,
Tom Karygiannis
The National Institute of Standards and Technology, {Wayne.Jansen,Tom.Karygiannis}@nist.gov

Michaela Iorga,
Serban Gavrila
VDG Inc., {Michaela.Iorga, Serban.Gavrila}@nist.gov

Vlad Korolev
Booz-Allen Hamilton, Vlad.Korolev@nist.gov

Abstract

The adoption of wireless technologies and handheld devices is becoming widespread in business, industry, and government organizations. The use of handheld devices introduces new risks to existing enterprise computing resources. Therefore, organizations require new strategies to mitigate the security risks associated with the integration of wireless technologies into existing computing environments. In this paper, we describe a framework for managing user privileges on handheld devices. Our framework aims at assisting enterprise security officers in creating, distributing, and enforcing group and individual security policies for Personal Digital Assistants, and helping users to automatically comply with their organization's security policy. Details of a proof-of-concept implementation of the framework are also provided.

Keywords

Security Policy, PDA, Trust Management

1. Introduction

Wireless handheld devices, such as Personal Digital Assistants (PDAs), enable mobile ad-hoc networking of the workforce and provide flexible enterprise data access and electronic-commerce capabilities. While mobile computing opens new application areas, its characteristics introduce vulnerabilities to attacks varying from tunneling-like, inadvertent actions to deliberate, aggressive interferences with corporate operations [Bro, Jan01, Kar02]. PDAs increasingly retain corporate information, but unlike their desktop counterparts, they lie at the periphery of organizational controls and oversight. Limited computing power, memory, interfaces, and battery life impose constraints on the practicality of applying standard safeguards. The PDA's small size and mobility also leads to greater exposure to theft or misuse in the field. Serious security concerns stem from the variety of ways in which a PDA can interact with other computing resources. These devices can inadvertently transfer malicious applications from one PDA onto another, or throughout the corporate network. Since PDA-enabled, application-level malware cannot typically be blocked by corporate firewalls, a PDA may serve as a back channel through which network vulnerabilities can be exploited. In short, a PDA is exposed to multiple risks associated with external communications and interfaces over which corporate security officers only exercise limited control.

To reduce or eliminate common risks associated with handheld devices, an enterprise security officer should have the means to express, monitor, and enforce corporate security policy effectively, particularly over external communications and interfaces. In this paper, we describe a general framework for managing and enforcing PDA security policies. The aim of this framework is twofold: first, to assist enterprise security officers in setting, monitoring, and enforcing PDA security policies, and second, to help users comply with assigned policies. The work builds on earlier research in this area [Jan02]. The description includes details of a proof-of-concept implementation of the framework using an embedded Linux distribution for an iPAQ PDA.

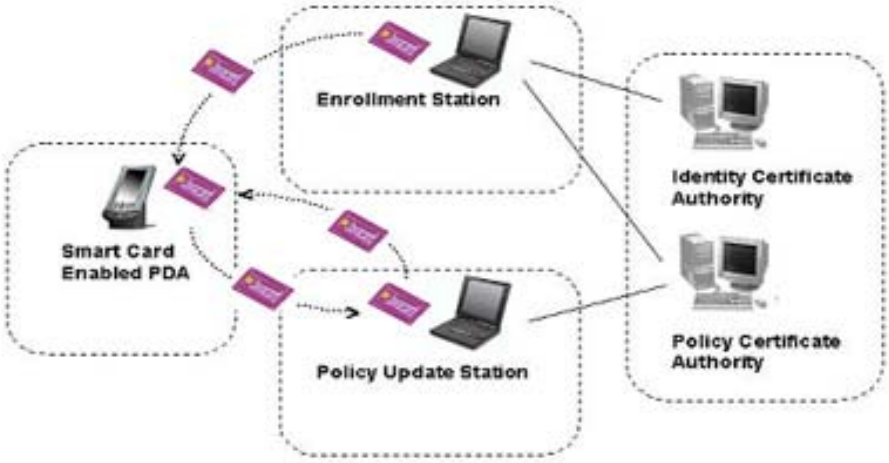

Figure 1: Security Policy Enforcement Components.

The Open Palmtop Integrated Environment (OPIE) [OPIE] is used as the desktop environment. Linux and OPIE were chosen for ease of development and availability of open source code.

2. Overview

Our framework addresses the main components of a trust management system [Bla96, Bla99, Mor00, Slo99.] The key element is the definition of a language (outlined below) for describing actions and specifying application policies through the use of digital certificates called *policy certificates*. These certificates encode the policy settings assigned to a user by the designated policy-setting authority. Policy certificates are linked to *X.509 identity certificates* that are used to establish the identity of the user. In order to prevent attacks on the policy content, a trusted distribution of the policy from a protected server is secured by using *smart cards.* The smart cards convey the X.509 identity certificates and the policy certificates from the back-end, protected server, represented by the *enrollment station* or *policy update station* to the front-end, smart-card-enabled PDA. The PDA is loaded with the guarding software that incorporates a mechanism for identifying principals and enforcing the security policy, called the *policy enforcement engine*.

Our framework focuses primarily on expressing and assigning enforceable policies through policy certificates that contain elements identifying the owner, the issuer, the period of validity, and the policy assigned to an entity, all cryptographically signed by the issuer. The approach is illustrated in Figure 1. From the user's perspective, the smart card is simply a token that enables access to the PDA.

The policy enforcement process begins with enrollment of a user. During enrollment, a security officer uses the enrollment station to generate an identity certificate and policy certificate for the user. The certificates are then stored on a smart card issued to the user. The enrollment station interacts with an X.509 Certificate Authority (CA) and a Policy Certificate Authority (PCA) to obtain the identity and the policy certificates for the user. Detailed explanations of this process are provided below in the Enrollment Station and Policy Management section.

After obtaining a certificate-bearing smart card, the user can take it to a smart card-enabled PDA, which reads the certificates from the smart card, validates them, and enforces the policy on the handheld device. If no smart card is present in the device or if the smart card contains an invalid certificate, a restrictive, default policy is applied. Detailed explanations of the enforcement mechanism are provided in the Personal Digital Assistants and Policy Enforcement section.

3. Policy Expression and Representation

A policy certificate is a structured set of information that conveys the policy assigned to an entity. The elements of the policy certificates are illustrated in Figure 2. They closely follow the form and content of the X.509 attribute certificates. For the sake of brevity, we have chosen for the policy certificates representation a grant-style language for low-level PDA-specific information flows, formatted in eXtended Markup Language (XML). In addition to being easily readable and interpretable using common utilities, the associated overhead of having to decode the ASN.1 representation of an X.509 certificate on a PDA is avoided.

```
<Attributes syntax="PDAPolicy">
        <policyEntry action="interface"  source="irda" target="*" />
        <policyEntry action="interface"  source="serial" target="*" />
        <policyEntry action="socket"  source="in:inet:22:127.0.0.1/0:*" target="*">
</Attributes>
```

Figure 2: Sample Policy Expression

The issued policy comprises a set of policy entries or rules conveyed within the XML Attributes element of the policy certificate. For protection against tampering and forgery attempts, certificates are digitally signed. In order for a device to verify the signature and establish the authenticity of a policy certificate, it must hold the corresponding public key of the PCA, the issuer of the certificate.

For added flexibility, policies other than those following our policy specification style, which is designated as "PDAPolicy," may be conveyed within the Attributes XML element. This allows transition to or support for other forms of policy expression should the need arise. The policy certificate itself is also structured to be able to convey multiple policies within the same certificate.

The policy language for expressing policy rules within a policy certificate follows a grant-style form of specification, whereby security-relevant actions are denied on a device unless enabled by a policy entry. Thus, each policy entry selectively enables increased privilege over the device's computational resources. The one exception to this principle, made to avoid large sets of rules, is that normal file system privileges of the operating system are enabled, unless denied by a policy entry. This particular formulation was possible with the chosen platform for our implementation.

Policy entries are a set of related *action, source*, and *target* attribute/value pairs of a single XML element of the form <policyEntry action="value" source="value" target="value" />, whereby:

- the *action* attribute specifies controls that the PDA is permitted to use and which the policy enforcement mechanism is able to mediate,
- the *source* attribute specifies named objects or applications on the PDA, such as interface names, IP addresses, or file path names that further refine the action attribute, and
- the *target* attribute specifies points of interface or reference to complete an action.

The *source* and *target* attribute values of a policy entry depend entirely on the value of the *action* attribute. Three values for the *action* attribute are currently supported: ***interface***, ***socket***, and ***file***.

For an *action* attribute having a value of ***interface***, the *source* attribute values supported pertain to the serial and infrared device ports and to PCMCIA or CF cards. The policy entries of this type are system-wide and applied equally to all applications. A specific *target* attribute value is used only with the PCMCIA or CF cards, to represent the device identifier. An asterisk symbol denotes any possible value.

For the *action* attribute having a value of ***socket***, the source attribute value specifies the pattern of allowed sockets, which is used to govern network socket communications. Currently, both TCP and UDP socket types are supported. The *target* attribute value specifies one or more applications that are allowed to use this socket, allowing greater granularity than with ***interface*** action values. An asterisk denotes any application. *Source* attribute values are represented using the following syntax for allowed sockets:

<Direction>:<Family>:<Local Port>::<Remote Address>:<Remote Port>

The <Direction> component denotes the direction of the traffic from the device's perspective; possible values are "in" and "out." The <Family> component specifies the socket family. In the present implementation, the supported family is "INET." The <Local Port> and <Remote Port> components indicate the Internet port numbers (e.g., 53 for Domain Name Server) to and from which the traffic is allowed to pass. An asterisk indicates any port. The <Remote Address> components specify the range of addresses to which a user is allowed to connect, in a format compatible with the WHOIS Internet database.

The address syntax is <nnn.nnn.nnn.nnn/bits>, where "nnn.nnn.nnn.nnn" is a numeric IP address and "bits" give the number of significant bits in this address (i.e., the leftmost) to apply.

For the *action* attribute with a value of ***file***, the *source* attribute value specifies one or more files, and the *target* value specifies one or more applications that are allowed to use this file. All other applications are barred from accessing the files specified. In addition, access to the applications indicated in the policy entry is limited only to read and execute. Besides enabling an application's use of files, such policy entries can be applied selectively; for example, to control access to networking libraries or prevent the reinstallation or overwriting of critical software.

Figure 2 illustrates a sample policy. The set of policy entries appears within the Attributes XML element, between its starting and ending tags, which are labeled accordingly. Each entry grants a privilege. The first entry allows beaming "in" and "out" information through the IrDA interface on the device, while the second entry allows the serial port of the device to be similarly enabled. The third entry allows anyone to login remotely via a specific communication socket through either of those interfaces. In practice, we tend to manage privileges by groups of related policy entries that fulfill a particular objective, such as allowing Web access to approved domains, enabling the use of Personal Information Management (PIM) applications, or enabling recognition of supported CF cards, which are then allocated to organizational roles.

4. Enrollment Station and Policy Management

The enrollment station allows the enterprise security officer to specify the security privileges of a user, request and verify policy and identity certificates issued to a user, place user certificates onto smart cards, and manage templates of predefined sets of privileges. It also provides the functionality to update an expired or obsolete policy certificate for an enrolled user. The security officer, after being successfully authenticated, runs the enroller application with full privileges. A user may update a policy certificate at a separate station, set up exclusively for this purpose.

The various transactions performed by the enroller application during an initial user's enrollment are shown in Figure 3. To request and issue the identity certificate, the enroller application first generates a pair of public-private cryptographic keys for the user being enrolled. It then generates a PKCS #10 certificate request, populated with the user's information entered by the security officer and the newly generated public key. The enroller signs the certificate request with the newly generated private key and sends the request to the CA. The CA responds with a PKCS #7 formatted identity certificate and certificate chain for the user.

Once the user's identity certificate is obtained from the CA, the enroller application requests a policy certificate. The PCA maintains a mapping between the various roles in the organization and a set of policy templates, which is used in generating a policy certificate for a user based on user's role within the organization. The policy templates contain an appropriate set of policy entries for some organizational unit. Both the mapping and the templates can be managed either locally by the PCA administrator or remotely by the security officer using the enroller application.

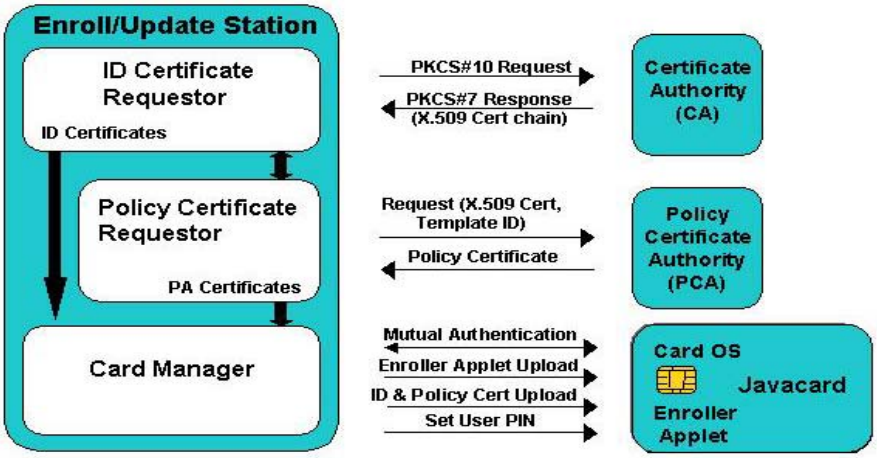

Figure 3: Enrollment Transactions

For secure remote administration, the enroller application and PCA communicate through the SSL v3 protocol.

To request a policy certificate from the PCA, the enroller application generates a policy certificate request containing the user's identity certificate and a policy template identifier. The latter is determined from a role-to-template mapping table, maintained by the PCA, and from the user's role, carried in the identity certificate as part of the user's distinguished name (i.e., the organizational unit). The request is sent to the PCA, which verifies the identity certificate and issues the policy certificate containing the privileges from the requested template to the owner of the identity certificate. The issued policy certificate is signed with the PCA's private key.

To complete the process, the enroller application places the certificates on the smart card and issues it to the user. For this proof-of-concept implementation we used a Java® smart card, compliant with both Javacard 2.1 and Global Platform 2.0.1. The identity and policy certificates are stored and protected by a custom on-card applet we developed. The security access rules are set such that read operations from the card are PIN protected while write and update operations require mutual authentication and a secure communication channel established with the card. If the policy certificate expires or becomes obsolete (e.g., the role-to-template mapping has changed) and the identity certificate is still valid, the policy certificate can be updated at a policy update station. The updater application reads both the identity and policy certificates from the smart card and presents them to the PCA in an update request. The PCA verifies the identity certificate and the policy certificate against its database, and issues a new policy certificate corresponding to the current role-to-template mapping. The updater application then updates the policy certificate on the smart card.

5. Personal Digital Assistants and Policy Enforcement

The policy enforcement occurs on the PDA and ensures that the user adheres to the granted enterprise policy. The various policy enforcement mechanisms implemented supplement, rather than replace, existing operating system security mechanisms. The policy enforcement engine comprises of two logical components: a policy manager and a policy enforcer, as illustrated in Figure 4.

The components of the PDA policy manager take responsibility for obtaining certificates from the smart card via the smart card module, validating them, extracting the policy entries from the policy certificate, and passing the entries to the policy enforcer to enact. A simple user interface module displays a message to the user and obtains the PIN code of the smart card.

The smart card module is responsible for detecting smart card insertion and removal, authenticating the user to the smart card (via PIN), downloading the policy from the smart card, and forwarding all of these events to the policy manager. The policy manager logs event information provided from the smart card module and

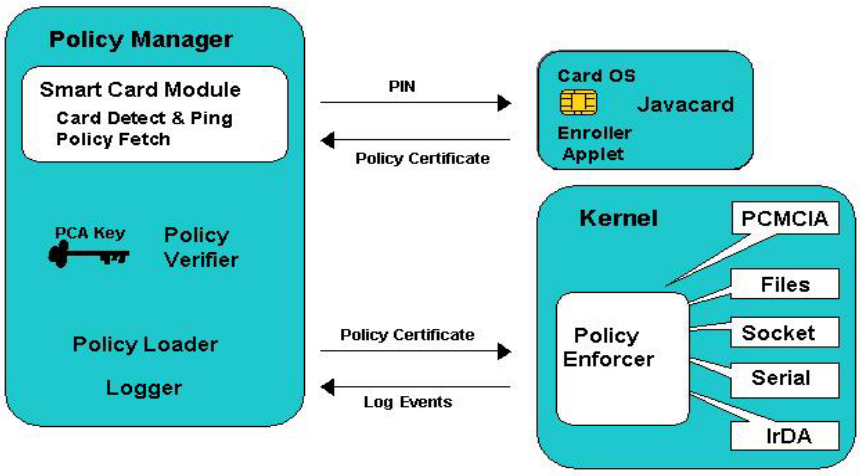

Figure 4: Policy Enforcement Engine

from the policy enforcer, and reinstates the default policy if the smart card is removed.

The policy enforcer is a set of kernel-resident policy enforcement mechanisms that mediate actions to resources and, as appropriate, impose either the default policy or the smart card resident policy. When the user attempts to perform security-relevant actions, the policy enforcer checks the policy information, and grants or denies the user permission to perform the requested action. The policy enforcer is implemented through a collection of kernel patches that are applied to several parts of the Linux kernel, such as: the serial port driver, IrDA protocol stack, TCP/IP network stack, file system, socket manager, and PCMCIA card manager.

In addition to these patches, the policy enforcer adds two more components to the kernel: the process monitor and the authorization module. The process monitor makes sure that all components of the system are running and, if one of the components is terminated due to an occurred error, the process monitor restarts it. The authorization module receives the queries from the patched components and decides if the requested operation is allowed based on the policies and the commands received from the policy manager. If an entry is prohibited, the authorization module generates an audit event.

The policy enforcer shares the kernel space and the user interface module is implemented as a plug-in module for OPIE, while the rest of the modules are user space processes. Having multiple independent processes allows for a simplified design. Each process can be executed independently, and if one of the processes erroneously terminates, the rest of the system is still functional. This design also allows the system to be tailored to work with different kinds of computational environments and different kinds of tokens. All user space modules are stateless and atomically communicate with each other via UNIX domain sockets. Communication between the policy enforcer and the rest of the system is done via Linux /proc interface.

6. Conclusions

The ability for a policy-setting authority, such as a security officer, to control information flow and other policy settings on a handheld device is an area that holds promise for improved security, yet has not received much attention. The approach we took is one that is relatively straightforward and flexible, and one that we believe is suitable for many organizational environments, particularly those where smart cards are a facet of the security infrastructure. Organizations, such as military, health care, and law enforcement, where PDAs regularly retain highly critical information, are considered prime candidates. The approach mitigates external threats by specifying the conditions under which information can be exchanged with the handheld device, and mitigates internal risks by not only specifying, but also enforcing the corporate handheld security policy.

References

[Bla96] Matt Blaze, Joan Feigenbaum, and Jack Lacy, "Decentralized Trust Management," IEEE Conference on Privacy and Security, Oakland, 1996.

[Bla99] Matt Blaze, Joan Feigenbaum, John Ioannidis, and Angelos D. Keromytis, "The Role of Trust Management in Distributed Systems Security," In Secure Internet Programming: Issues in Distributed and Mobile Object Systems, Springer-Verlag Lecture Notes in Computer Science, pp. 185 - 210, Berlin 1999.

[Bro99] Brown, Bruce and Marge. "Secure Your PDA," PC Magazine, December 1, 1999. <URL:http://cma.zdnet.com/texis/techinfobase/techinfobase/+zwq_qr+6XvXKs/zdisplay.html>.

[Jan02] Wayne Jansen, Tom Karygiannis, Serban Gavrila and Vlad Korolev, "Assigning and Enforcing Security Policies on Handheld Devices," Proceedings of the Canadian Information Technology Security Symposium, May 2002.

[Kar02] Tom Karygiannis and Les Owens, "Special Publication 800-48: Wireless Network Security: 802.11, Bluetooth, and Handheld Devices," National Institute of Standards and Technology, November 2002, <URL:http://csrc.nist.gov/publications/nistpubs/800-b48/NIST_SP_800-48.pdf>.

[Mor00] Mark Moriconi, Shelly Qian, United States Patent 6,158,010, System and method for maintaining security in a distributed computer network, December 5, 2000, <URL:http://www.uspto.gov>.

[OPIE] OPIE User Manual, The OPIE Team, 2002, <URL:http://opie.handhelds.org/usermanual/index.htm>.

[Slo99] M. Sloman, N. Dulay, and B. Nuseibeh, "SecPol: Specification and Analysis of Security Policy for Distributed Systems," Imperial College, <URL:http://www-dse.doc.ic.ac.uk/projects/secpol/SecPol-overview.html>.