NIST
National Institute of
Standards and Technology
Technology Administration
U.S. Department of Commerce

Interagency Report 6981

Policy Expression and Enforcement for Handheld Devices

Wayne Jansen
Tom Karygiannis
Vlad Korolev
Serban Gavrila
Michaela Iorga

NIST Interagency Report - 6981

Policy Expression and Enforcement for Handheld Devices

Wayne Jansen
Tom Karygiannis
Vlad Korolev
Serban Gavrila
Michaela Iorga

C O M P U T E R S E C U R I T Y

Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20988-8930

April 2003

U.S. Department of Commerce
Donald L. Evans, Secretary

Technology Administration
Phillip J. Bond, Under Secretary of Commerce for Technology

National Institute of Standards and Technology
Arden L. Bement, Jr., Director

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analysis to advance the development and productive use of information technology. ITL's responsibilities include the development of technical, physical, administrative, and management standards and guidelines for the cost-effective security and privacy of sensitive unclassified information in Federal computer systems. This Interagency Report Discusses ITL's research, guidance, and outreach efforts in computer security, and its collaborative activities with industry, government, and academic organizations.

National Institute of Standards and Technology Interagency Report 6981
19 pages (April 2003)

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

Table of Contents

Policy Expression and Enforcement for Handheld Devices

Abstract

The use of mobile handheld devices, such as Personal Digital Assistants (PDAs) and tablet computers, within the workplace is expanding rapidly. These devices are no longer viewed as coveted gadgets for early technology adopters, but instead have become indispensable tools that offer competitive business advantages for the mobile workforce. While providing productivity benefits, the ability of these devices to store and transmit corporate information through both wired and wireless networks poses potential risks to an organization's security. This paper describes a framework for managing user privileges on handheld devices. The approach is aimed at assisting enterprise security officers in administering and enforcing group and individual security policies for PDAs, and helping constrain users to comply automatically with their organization's security policy. Details of a proof-of-concept implementation of the framework are also provided.

Keywords: *Security Policy, PDA, Handheld Devices, Digital Certificates*

Introduction

Wireless handheld devices, such as PDAs, enhance employee productivity by enabling the mobile workforce to communicate with customers and colleagues, access enterprise data, and conduct electronic commerce transactions while away from their offices. Digital cellular phones, now almost a necessity for today's workforce, are incorporating PDA functionality, while at the same time, PDAs are assimilating cellular phone capabilities as add-on or integrated modules. As the capabilities of handheld devices converge and prices become more affordable, these devices are likely to continue to find new applications in many sectors such as medical, financial, manufacturing, defense, and law enforcement.

While mobile computing opens up countless new applications to enhance productivity, it also raises a number of security concerns [Bro, Joh01, Bea01, Gue01]. 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 many types of safeguards. Their small size and mobility also leads to greater exposure to theft or misuse in the field.

Many of the most serious security concerns stem from the variety of ways in which a PDA can interact with other computing resources. Typically, these devices are equipped with the capability to communicate wirelessly over limited distances to other devices using infrared or radio signals. Moreover, many users are unaware of the risks associated with the use of these devices within and outside an enterprise. For example, a business associate could unknowingly beam a Trojan horse application from her PDA to a colleague's PDA through an IrDA port, and contaminate it. The victim could subsequently spread the malware to the corporate network when synchronizing the PDA to a desktop or notebook computer. Given that the malware was not analyzed by the corporate firewall, the PDA may inadvertently serve as a back channel through which network vulnerabilities are exploited. Similarly, a user could browse the Internet using a third party wireless ISP, independent from the wired infrastructure of the organization, and download or upload data or applications in violation of the corporate security policy. In short, a PDA is exposed to multiple risks associated with external communications and interfaces over which the corporate security officer cannot exercise any control.

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

Overview

A key element of the proposed framework is the use of digital certificates called policy certificates, which convey policy settings assigned to a user by some policy-setting authority, such as an enterprise security officer. A policy certificate, as typical of any digital certificate,

contains elements identifying the issuer, its period of validity, and other relevant information, all cryptographically signed by the issuer. Policy certificates are linked to X509 identity certificates. Consequently, our approach relies on the availability of an X.509 Public Key Infrastructure for managing identity certificates. Other components of our framework include an enrollment station, a policy update station, a PDA equipped with a smart card reader and policy enforcement software, as illustrated in Figure 1. The approach is straightforward – smart cards are used to convey certificates, issued to a user at an enrollment or update station, to an appropriately equipped PDA where they are enforced. The PDA maintains a highly restrictive default policy in the absence of a smart card bearing a valid set of credentials. From the user's perspective, the smart card is simply a token that enables access to the PDA.

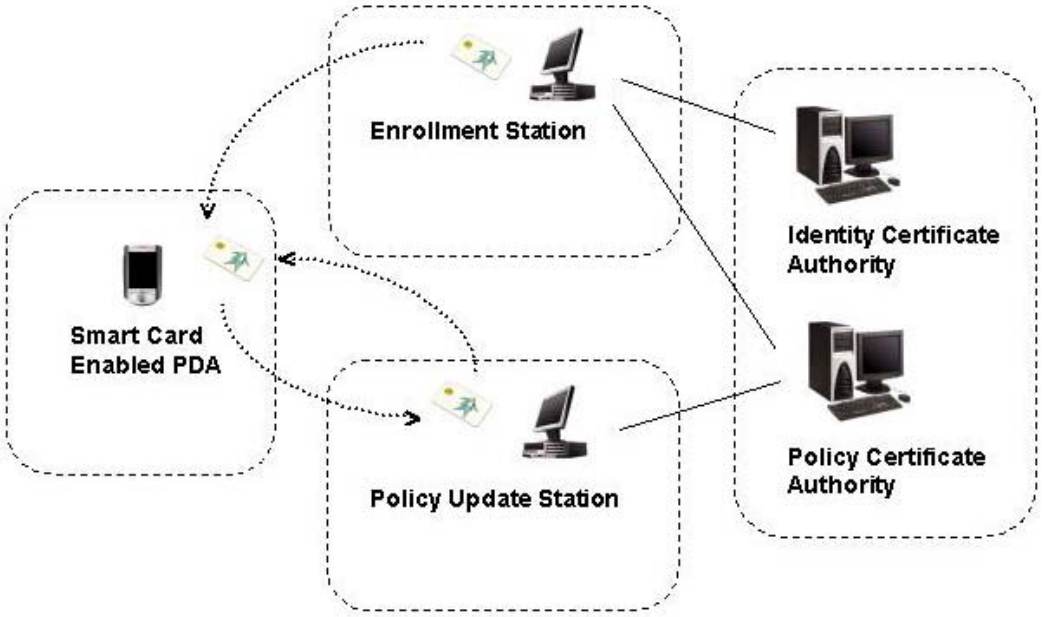

Figure 1: Security Policy Enforcement Components[1]

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. Though the certificates could be distributed through other means that would not require the PDA to be equipped with a smart card reader [Jan02], the focus here is on organizations where smart cards are already incorporated into the security infrastructure.

The enrollment station interacts with an X.509 Certificate Authority (CA) and a Policy Certificate Authority (PCA) to obtain respectively the identity and policy certificates for the user. Besides certificate handling, the enrollment station can also be used to create and manage policy templates for the organization. Policy templates simplify the security officer's task in defining and editing named sets of policy rules, and later, when policy certificates are issued, mapping those policies to individual or groups of users based on their assigned roles. Policy templates are maintained at the PCA.

[1] The HP product images are used with permission, courtesy of Hewlett-Packard Company.

After obtaining a certificate-bearing smart card, the user can take it to an appropriately equipped PDA, which reads the certificates from the smart card, validates them, and enforces the policy on the device. If no smart card is present in the device or if the smart card contains an invalid certificate, a default policy applies. The default policy restricts the use of any communications protocol, Personal Information Management (PIM) application, or interface module (e.g., PCMCIA or CF device), except for those needed to interact with the smart card. The underlying principle is that a policy enforcement mechanism, preloaded on the device, allows only default actions to be performed, until it holds a valid policy certificate. Thereafter, the enforcement mechanism allows only those actions specified by the policy settings within the certificate.

The identity certificate or policy certificate on the smart card are valid only for a limited period and eventually will expire. The validity period of the policy certificate is usually much shorter than that of its associated identity certificate and is always set to expire before the identity certificate does. As long as the identity certificate remains valid, a user can obtain a new policy certificate at a policy update station. The update station authenticates the user, reads and validates the identity certificate on the smart card, and uses the identity certificate to obtain a new policy certificate for the user. The new policy certificate replaces the old one on the smart card. The user can then take the updated smart card to an appropriately equipped PDA and resume processing under the newly issued policy. If both the identity and policy certificates have expired, the user must again go through the enrollment process, possibly recycling the original smart card.

Policy Certificates

A policy certificate is a structured set of information that conveys the policy assigned to an entity. Figure 2 illustrates the elements of the policy certificate, which closely follow the form and content of X.509 attribute certificates. Instead of using X.509 certificates and relying on an ASN.1 encoding, we use XML to define and generate the external representation for policy certificates. Many aspects of certificate handling are simplified by using a textual representation for the policy certificate rather than a binary representation. In addition to being easily readable and interpretable using common utilities, the associated overhead of having to decode the ASN.1 representation on a PDA is avoided.

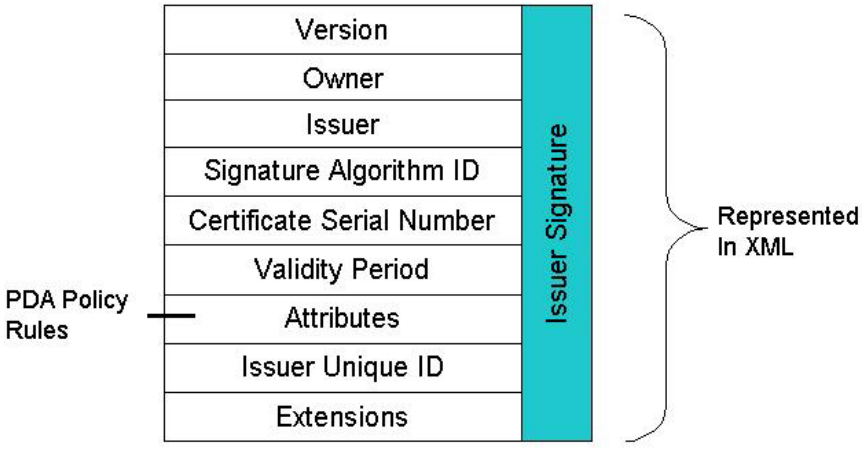

Figure 2: Policy Certificate Elements

The issued policy comprises a set of policy entries or rules conveyed within the Attributes element of the policy certificate. Besides the policy payload, other elements of the certificate indicate from whom (i.e., the issuer) and to whom (i.e., the owner) the certificate was issued, its period of validity, and supplemental information needed to establish its authenticity and apply the policy. A digital signature computed over the certificate elements protects the policy certificate from tampering and thwarts attempts to forge certificates. In order to verify the signature and establish the authenticity of the certificate, a device must hold the corresponding public key of the issuer of the certificate, the PCA.

For added flexibility, policies other than those following our policy specification style, which is designated as "PDAPolicy," may be conveyed within the Attributes 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. This would allow, for example, the ability to specify a policy for a Java or other virtual machine environment running on a mobile device, in addition to that for the PDA operating system.

Policy Expression

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 access privileges of the operating system are enabled, unless denied by a policy entry.

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 control points 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.

- 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; for example, to enable the device to use its serial ports for synchronizing the PDA with a workstation, beam data through the IR port, or accept organizationally supported PCMCIA cards. A specific target attribute value is used only with the PCMCIA or CF cards, to represent the device identifier. An asterisk 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; for example, to enable the device to communicate wirelessly with organizational or other approved sets of hosts. 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* actions. 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.

The policy statements are tied closely to the various policy enforcement mechanisms implemented for the PDA operating system, discussed later in the paper. For example, when a user attempts to beam data to another PDA, a policy enforcement mechanism must first check the parsed policy information to determine whether the action is allowed before letting the action proceed. If the security policy does not explicitly grant the beaming, the action is denied, the user is notified of the attempted policy violation, and the event is logged into the audit trail. Similarly, to prevent the enterprise firewall from being circumvented, any PDA connection to the Internet through a third party ISP using wireless modems or wireless LAN/WAN cards must be controlled by some enforcement mechanism.

Figure 3 illustrates a sample policy. The set of policy entries appears within the Attributes element, between its starting and ending tags, which are labeled accordingly. Each entry grants some 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 communications 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 critical PIMS, or enabling recognition of supported CF cards, which are then allocated to organizational roles.

```
<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 3: Sample Policy Expression

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 also run the enroller application, but under very restricted privileges (i.e., only able to update his or her policy certificate). As mentioned earlier, a user typically updates a policy certificate at a separate station, set up exclusively for this purpose. Though it would be possible to have the PDA update policy certificates directly over the corporate network in lieu of an update station, we have not yet implemented this feature for the PDA.

The various transactions performed by the enroller application during an initial user's enrollment are shown in Figure 4.

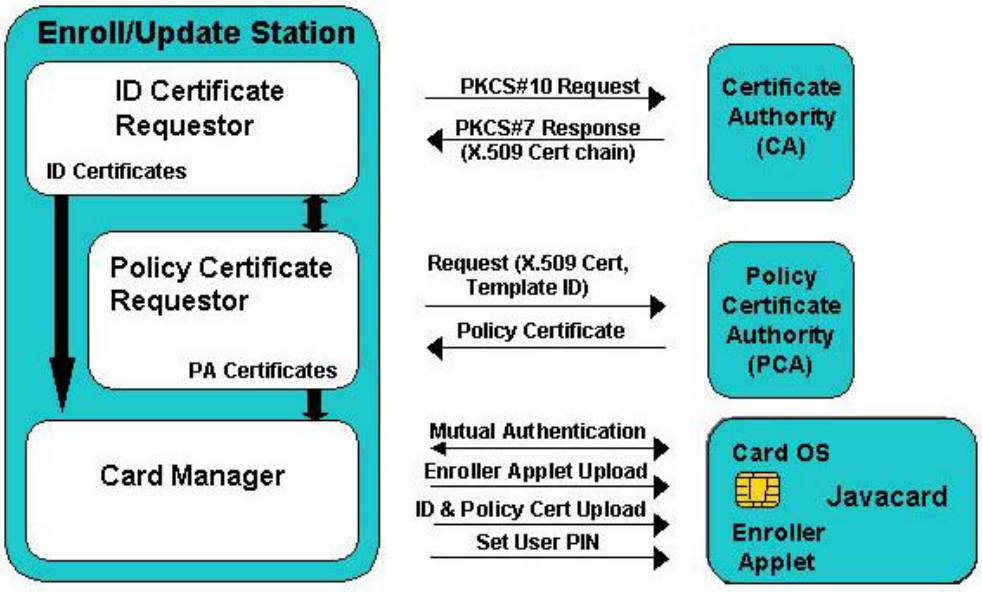

Figure 4: Enrollment Transactions

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. In our initial implementation, the CA was configured to issue the identity certificate immediately, allowing the enroller application to obtain the identity certificate in the same call to the CA. However, it is easy to implement a more realistic approach, where the CA sets the request to "pending" and waits for the CA administrator's approval, requiring the enroller application to return later and ask about the request's state.

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 his or her 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. 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 has only to place the certificates on the smart card and issue it to the user. We chose a Javacard type of smart card, compliant with both Javacard 2.1 and Global Platform 2.0.1. An on-card applet we developed stores the identity and policy certificates, protects the certificates by setting up a PIN, and retrieves the certificates after PIN verification. The Application Protocol Data Units (APDUs) used to communicate with the on-card applet once it is established on the smart card are given in Appendix A.

Storing the certificates on the smart card is a four-step operation. First, the enroller application and the Card Manager applet (i.e., part of the Card OS) establish a secure communication channel by mutual authentication. Second, our applet is loaded onto the card. Third, the enroller application sends the certificates to our on-card applet, which stores them onto the smart card's memory. Finally, the user-selected PIN is set onto the card. The PIN must be presented and verified, each time the certificates need to be read from or updated on the smart card.

When the policy certificate expires or become obsolete (e.g., the role-to-template mapping has changed), it must be updated. The policy certificate update process is similar to that used for an initial enrollment. The user can perform the update operation at an update station, provided that the identity certificate on the smart card is valid and the correct PIN for the card is known. 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 communicates with our on-card applet to replace the policy certificate on the smart card.

Except for an off-the-shelf CA, all the policy management software components were written in the Java programming language.

Policy Enforcement

The policy enforcement occurs on the PDA and ensures that the user adheres to the enterprise security officer's granted security policy settings. The various policy enforcement mechanisms implemented supplement, rather than replace, existing operating system security mechanisms. Policy enforcement is divided into two parts: a policy manager and the policy enforcer, which are illustrated in Figure 5.

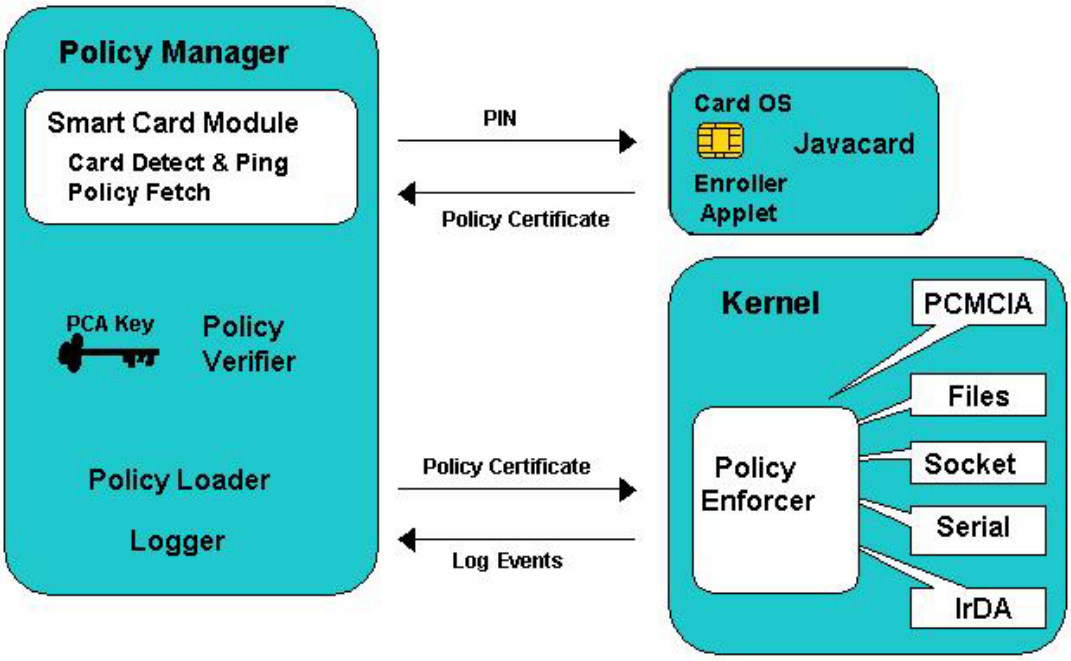

Figure 5: Policy Enforcement

The components of the PDA policy manager takes 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 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 kernel: 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 requires two additional components in 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 because of some error, the process monitor restarts it. The authorization module receives queries from the patched components and then decides if the operation represented by the query is allowed. If the entry is not allowed the authorization module generates an audit event. The authorization module also receives policies and commands from the policy manager.

The specific policy enforcement mechanisms are easily understood through the three main classes of policy statements with which they are associated, identified by the action attribute values: *interface*, *file*, and *socket*.

For the action attribute with a value of *interface*, policy entries pertaining to the serial and infrared port are enforced for each data transfer. However, PCMCIA or CF card handling occurs only during the initialization phase, when a card is inserted into the slot and the card services manager sends a query containing the device identifier to the authorization module. If the authorization module decides that current policy does allow this type of card, the card manager attaches the device driver to this card. Otherwise, the card initialization stops and all data going to and from this interface are discarded.

For the action attribute with a value of *file*, all policy entries are enforced when the file is being opened. Should a new policy be enabled that prohibits the use of a file that is already opened by an application, the application can still continue to access this file as before, until the file is closed and another attempt is made to open it.

For the action attribute with a value of *socket*, all policy entries are enforced for each data transfer. When the transport protocol layer (i.e., TCP or UDP) receives a packet from the network interface (incoming) or an application (outgoing), it sends the query containing the remote address and local and remote port numbers to the authorization module. Based on the current policy, the enforcement module decides if this packet is allowed to pass. Depending on the verdict the transport protocol layer receives from the enforcement module, it either lets the packet continue along or simply discards it.

The proof-of-concept implementation of policy enforcement was developed for an embedded Linux distribution for an iPAQ PDA. The Open Palmtop Integrated Environment (OPIE) [OPIE] was used as the desktop environment. Linux and OPIE were chosen for ease of development and availability of open source code. Moreover, the Linux kernel provides an equally secure environment for handheld devices as it does for workstations and servers.

The policy enforcer is realized within 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, because each process can execute without blocking the rest of the system, and if one of the processes terminates for some reason, 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 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. All communications between the modules are stateless and atomic. Any module can send a message to the other module at any time, and no module should block waiting for the message. Appendix B gives a detailed example scenario that illustrates the role the various modules play in the policy enforcement process.

Advantages and Disadvantages

Any security scheme has its advantages and disadvantages. While we have emphasized the advantages of our framework throughout the paper, in fairness, some of the potential disadvantages should be pointed out:

- The PDA's operating system is a critical factor in the effectiveness of our scheme, since it serves as the foundation for the policy enforcement mechanism. A weak security architecture or unresolved vulnerability within operating system undermines the policy enforcement mechanism. Similarly, any flaws in the policy enforcement mechanism may create new vulnerabilities in the operating system.

- The granularity of the policy may not be fine enough in some instances. For example, since the IR beaming protocol currently does not support identification or authentication of the peer, the beaming action from a particular application can be denied or granted but not confined to specific sets of PDAs.

- A user could circumvent the security policy by transferring information from the scope of one application whose features are disabled to another application having those features enabled. For example, a user's policy certificate may prohibit beaming information from one application, but allow it from the other.

- Not all handheld devices readily accept smart cards. In fact, for our implementation, we needed to modify an open source Linux PCMCIA smart card reader driver for the ARM architecture. Fortunately, those modifications have been incorporated into the open source driver, removing that technical hurdle for other implementers.

- Organization policy can be difficult to translate into PDA policy rules. The size of the rule may become lengthy and impose on the memory capacity of the smart card. If persistent memory is used on Java smart cards, this will affect the number of time a policy certificate can be updated before memory is exhausted.

- A privilege revocation mechanism for policy certificates is not currently supported. Therefore, validity periods must be set within typical organizational policy update cycles, to have policy updated consistently. Even so, on occasion it may be necessary to update policy out of cycle, in which case the organization is dependent on procedural or other means to have users perform the update.

Our experience to date has not shown any of these potential drawbacks to be a problem in practice. However, we are planning to improve some areas, such as extending the security mechanisms to allow for granting or denying certain actions to a peer that can be uniquely identified and authenticated, and to provide more granularity in wireless protocol policies. We also plan to add additional policy controls for Bluetooth interfaces.

Related Research

The area of trust management [Bla96, Bla99] aims toward a comprehensive approach to specifying and interpreting security policies, credentials, and relationships in order to authorize security-related actions within distributed systems. Trust management engines typically do not directly enforce policy, and instead provide a policy decision to the application that invokes it. Strictly speaking, the overall framework we employ is not a trust-management engine according to the precise use of the term, because rendering a decision is done in an application-dependent manner. However, the key components of a trust management system are addressed, which include a language for describing actions, a mechanism for identifying principals, a language for specifying application policies, a language for specifying and delegating credentials, and a compliance checker.

Marconi et al. disclose a system and method for maintaining security in a distributed computer network in United States Patent 6,158,010. The patent describes "a system and method for maintaining security in a distributed computing environment comprises a policy manager located on a server for managing and distributing a security policy, and an application guard located on a client for managing access to securable components as specified by the security policy. In the preferred embodiment, a global policy specifies access privileges of the user to securable components [Mor00]." This approach does not utilize policy certificates or signed policy objects. Thus, a secure implementation in general would require trusted distribution of the policy from a protected server to the application guard, in order to prevent attacks on the policy content. One drawback to using an online policy server is that a handheld device may not always be able to communicate with it.

Odyssey Research Associates have developed a policy-modeling tool to enable the modeling and representation of security policies for clinical workflow processes. This tool incorporated a user interface, support for policy modification, policy consistency checking, policy modeling and administration, and the translation of modeled policies into runtime enforcement rules for workflow engines [ORA01]. Our tools do not provide any modeling techniques for policy consistency, and focus primarily on expressing and assigning enforceable policy using a policy certificate.

Researchers at the Imperial College have investigated techniques and developed tools for specifying security policies, analyzing policies for inconsistencies and conflicts, and defining

obligation and authorization policies [Slo]. The goal of their project is to translate policy specifications into executable policy agents that can be used to install and enforce policies within a particular target environment. Our policy modeling language is currently a grant-style language for low-level PDA-specific information flows, which would benefit from a more robust and expressive policy language.

Conclusion

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, heath 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

[Bea01] Nelson Beach, "Handheld Security: A Layered Approach," SANS Institute June 25, 2001, <URL: http://rr.sans.org/PDAs/hand_sec.php>.

[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.

[Bro] Brown, Bruce and Marge. "Secure Your PDA," ZDNet HELP & HOW-TO <URL: <http://www8.zdnet.com/zdhelp/stories/main/0,5594,2403097-2,00.html>.

[Gue01] Susan Guerrero, "PDAs - A Security Primer," SANS Institute May 2, 2001, <URL: http://rr.sans.org/PDAs/sec_primer.php>.

[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.

[Joh01] M. Gregory St. John, "PDAs and Policy," SANS Institute, February 3, 2001, <URL: http://rr.sans.org/PDAs/PDA_policy.php>.

[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>.

[ORA01] Odyssey Research Associates, Policy Modeling for Healthcare Workflows, 2001, <URL: http://www.atc-nycorp.com/projects/past/healthcarePolicy.html>.

[Slo] 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>.

Appendix A: Enroller Applet's APDUs

Several APDUs are used by the enroller application to communicate with the enroller applet on the Javacard. They can be used by other applications to read the identity and policy certificates from the smart card.

In order to read the certificates, some conditions must be satisfied:

- The specialized Enroller applet must be present on the card (its AID is A0 00 00 00 1A FF 00 00 00 00 00 00 00 00 01 02);
- The certificates must be present on the card;
- The host application must present a valid PIN.

The certificates are stored on the card as byte arrays. To obtain a certificate, the host application first has to select the specialized Enroller applet, then get the certificate length (measured in bytes), and then repeatedly read blocks of desired length (but not bigger than 128 bytes) starting at appropriate offsets in the certificate array.

All the following APDUs except the last one (Get CPLC Data) are understood and executed by the Enroller applet. The last one must be addressed to the Card Manager applet (AID=A0 00 00 00 18 43 4D), and can be used to read the card serial number.

Select the Enroller Applet

CLA	INS	P1	P2	Lc	Data
0x00	0xA4	0x04	0x00	AID length	AID

Example:
00 A4 04 00 10 A0 00 00 00 1A FF 00 00 00 00 00 00 00 00 01 02

Status word returned:
9000: success
6283: applet blocked
6700: incorrect AID length
6999: select failed
6A82: applet not found
6A86: Incorrect P1, P2.

Verify PIN

CLA	INS	P1	P2	Lc	Data
0x90	0x16	0x00	0x00	PIN length	PIN

Example:
90 16 00 00 04 31 32 33 34 (verify PIN = "1234")

Status word returned:
9000: success
63Cx: failure (x = the remaining number of tries).

Get Policy Certificate Length

CLA	INS	P1	P2	Le
0x90	0x26	0x00	0x00	0x02

Example:
90 26 00 00 02

Status word returned:
9000: success
6986: command not allowed (e.g., applet or card blocked)
6982: security conditions not satisfied (e.g., failed to verify PIN before this command)
6985: conditions not satisfied (e.g., the certificate length is not yet set)

Get Identity Certificate Length

CLA	INS	P1	P2	Le
0x90	0x46	0x00	0x00	0x02

Example:
90 46 00 00 02

Status word returned:
9000: success
6986: command not allowed (e.g., applet or card blocked)
6982: security conditions not satisfied (e.g., failed to verify PIN before this command)
6985: conditions not satisfied (e.g., the certificate length is not yet set)

Get Policy Certificate Data

CLA	INS	P1	P2	Le
0x90	0x28	Offset msb	Offset lsb	Length (max 128)

Example:
90 28 04 80 78 (receive 0x78 bytes from the policy cert at offset 0x0480)

Status word returned:
9000: success
6986: command not allowed (e.g., applet or card blocked)
6982: security conditions not satisfied (e.g., failed to verify PIN before this command)
6985: conditions not satisfied (e.g., the certificate length is not yet set or the certificate is not present on the card)
6B00: wrong offset (e.g., the offset plus the length greater than certificate length).

Get Identity Certificate Data

CLA	INS	P1	P2	Le
0x90	0x48	Offset msb	Offset lsb	Length (max 128)

Example:
90 48 04 80 78 (receive 0x78 bytes from the identity cert at offset 0x0480)

Status word returned:
9000: success
6986: command not allowed (e.g., applet or card blocked)
6982: security conditions not satisfied (e.g., failed to verify PIN before this command)
6985: conditions not satisfied (e.g., the certificate length is not yet set or the certificate is not present on the card)
6B00: wrong offset (e.g., the offset plus the length greater than certificate length).

Get CPLC Data

CLA	INS	P1	P2	Le
0x80	0xCA	9F	7F	2D

The status word indicating success is 0x9000.
NOTE: This command must be sent to the card manager applet (AID = A0 00 00 00 18 43 4D), and not to the Enroller applet!
The card serial number is the 8 bytes starting at offset 13 in the returned buffer.

Appendix B: Startup Scenario for Policy Enforcement

The following example illustrates how the policy enforcement works by describing the flow of the data between different components of the system, some of which are illustrated in Figure 5 in the main body of this paper.

1. On system boot-up, the startup script tells the process monitor (through the /proc/policy file) the filenames of the policy manager and the smart card module.

2. The process monitor sees that the policy manager and smart card module are not running and starts them.

3. Upon initiation, the policy manager loads the default policy into the authorization module (through the /proc/policy interface). The default policy should contain entries that allow the smart card hardware module to function (i.e., PCMCIA smart card reader) and entries that allow only authorized programs to access the policy files on the file system and the /proc/policy interface, as well as other entries that the system administrator decides are necessary.

4. Since the user interface module, not shown in Figure 5, is implemented as a plug-in to the desktop environment, it does not need to be started explicitly.

5. At this point all the components of the system are running and the default policy having least privilege is loaded.

6. The user tries to beam an address book entry to another PDA. The patched IrDA driver notices the data transfer and sends the query to the policy authorization module asking if the use of the IR port is allowed.

7. The authorization module responds with the negative reply and adds an entry to the list of audit events. The IrDA driver receives a negative response from the authorization module and throws away the data packet.

8. The policy manager pulls the new audit event from the authorization module (via /proc/policyLog interface), records this event into the log file, and sends a command to the user interface module to display the “Beaming not allowed” message to the user.

9. The user inserts his smart card, containing the policy that allows beaming, into the smart card reader.

10. The smart card module detects the new smart card in the reader, it verifies that the smart card has the correct applet installed, and then sends a request for PIN to the user interface module. At this time, the smart card module also starts periodic monitoring of the smart card in the reader.

11. When the user interface module receives the request for the PIN, it displays the PIN keypad on the screen and waits for the user to enter the PIN number. When the user finishes entering the PIN, the user interface manager sends the PIN to the smart card module.

12. The smart card module receives the PIN and sends it to the smart card. The smart card verifies the PIN and responds with the appropriate return code.

13. When the smart card module receives the return code from the smart card, it sends a start request to the smart card to send the policy certificate. It then receives the certificate in APDU fragments and assembles them, restoring the certificate into its original form, as when originally generated by the enrollment application. The smart card module then sends the assembled certificate to the policy manager.

14. When policy manager receives the policy certificate, it uses the public key of the PCA it holds to verify the authenticity of the policy certificate. It also checks the validity period of the certificate. If the signature is valid and the certificate has not expired, the policy manager parses the certificate, loads it into the policy authorization module via the /proc/policy interface, and generates a "Policy Read" notification for the user interface module to display.

15. The user tries to beam an address book entry again. This time the authorization module allows the data to pass through the IR port, so the operation succeeds.

16. The user pulls out the card from the reader. The smart card module notices the card has been removed and notifies the policy manger that. When policy manager receives notification, it tells the authentication module to switch to the default policy. At this point, the system state is the same as in step 5 above.