Network Working Group
Request for Comments: 3157
Category: Informational

A. Arsenault
Diversinet
S. Farrell
Baltimore Technologies

August 2001

Securely Available Credentials - Requirements

Status of this Memo

Copyright Notice

Abstract

This document describes requirements to be placed on Securely Available Credentials (SACRED) protocols.

Table Of Contents

1. Introduction

"Credentials" are information that can be used to establish the identity of an entity, or help that entity communicate securely. Credentials include such things as private keys, trusted roots, tickets, or the private part of a Personal Security Environment (PSE) [RFC2510] - that is, information used in secure communication on the Internet. Credentials are used to support various Internet protocols, e.g., S/MIME, IPSec and TLS.

In simple models, users and other entities (e.g., computers like routers) are provided with credentials, and these credentials stay in one place. However, the number, and more importantly the number of different types, of devices that can be used to access the Internet is increasing. It is now possible to access Internet services and accounts using desktop computers, laptop computers, wireless phones, pagers, personal digital assistants (PDAs) and other types of devices. Further, many users want to access private information and secure services from a number of different devices, and want access to the same information from any device. Similarly credentials may have to be moved between routers when they are upgraded.

This document identifies a set of requirements for credential mobility. The Working Group will also produce companion documents, which describe a framework for secure credential mobility, and a set of protocols for accomplishing this goal.

The key words "MUST", "REQUIRED", "SHOULD", "RECOMMENDED", and "MAY" in this document are to be interpreted as described in [RFC2119].

1.1 Background and Motivation

In simple models of Internet use, users and other entities are provided with credentials, and these credentials stay in one place. For example, Mimi generates a public and private key on her desktop computer, provides the public key to a Certification Authority (CA) to be included in a certificate, and keeps the private key on her computer. It never has to be moved.

However, Mimi may want to able to send signed e-mail messages from her desktop computer when she is in the office, and from her laptop computer when she is on the road, and she does not want message recipients to know the difference. In order to do this, she must somehow make her private key available on both devices - that is, that credential must be moved.

Similarly, Will may want to retrieve and read encrypted e-mail from either his wireless phone or from his two-way pager. He wants to use whichever device he has with him at the moment, and does not want to be denied access to his mail or to be unable to decrypt important messages simply because he has the wrong device. Thus, he must be able to have the same private key available on both devices.

The following scenario relating to routers has also been offered: "Once upon a time, a router generated a keypair. The administrators transferred the public key of that router to a lot of other (peer) routers and used that router to encrypt traffic to the other routers. And this was good for many years. Then one day, the network

administrators found that this particular little router couldn't handle an OC-192. So they trashed it and replaced it with a really big router. While they were there, the craft workers inserted a smart card into the router and logged into the router. They gave the appropriate commands and entered the correct answers and so the credentials (keypair) were transferred to the new, big router. Alternatively, the craft people could have logged into the router, given it a minimal configuration and transferred the credentials from a credential server to the router. They had to perform the correct incantations and authentications for the transfer to be successful. In this way, the identity of the router was moved from an old router to a new one. The administrators were glad that they didn't have to edit the configurations of all of the peer routers as well."

It is generally accepted that the private key in these examples must be transferred securely. In the first example, the private key should not be exposed to anyone other than Mimi herself (and ideally, it would not be directly exposed to her). Furthermore, it must be transferred correctly. It must be transferred to the proper device, and it must not be corrupted - improperly modified - during transfer.

Making credentials securely available (in an interoperable fashion) will provide substantial value to network owners, administrators, and end users. The intent is that this value be provided largely independent of the hardware device used to access the secure credential and the type of storage medium to which the secure credential is written. Different credential storage devices, (e.g., desktop or laptop PC computer memory, a 3.5 inch flexible diskette, a hard disk file, a cell phone, a smart card, etc.) will have very different security characteristics and, often very different protocol handling capabilities. Using SACRED protocols, users will be able to securely move their credentials between different locations, different Internet devices, and different storage media as needed.

In the remainder of this document we present a set of requirements for the secure transfer of software-based credentials.

1.2 Working Group Organization and Documents

The SACRED Working Group is working on the standardization of a set of protocols for securely transferring credentials among devices. A general framework is being developed that will give an abstract definition of protocols which can meet the credential-transfer requirements. This framework will allow for the development of a set of protocols, which may vary from one another in some respects. Specific protocols that conform to the framework can then be developed.

Work is being done on a number of documents. This document identifies the requirements for the general framework, as well as the requirements for specific protocols. Another document will describe the protocol framework. Still others will define the protocols themselves.

1.3 Structure of This Document

Section 1 of this document provides an introduction to the problem being solved by this working group. Section 2 describes requirements on the framework. Section 3 identifies the overall requirements for secure credential-transfer protocols, and separate requirements for two different classes of solutions. Section 4 identifies Security Considerations. Appendix A describes the relationship of the SACRED solutions and credential-mobility solutions involving hardware components such as smart cards. Appendix B contains some additional scenarios which were considered when developing the requirements.

2. Framework Requirements

This section describes requirements that the SACRED framework has to meet, as opposed to requirements that are to be met by a specific protocol that uses the framework.

2.1 Credential Server and Direct solutions

There are at least two different ways to solve the problem of secure credential transfer between devices. One class of solutions uses a "credential server" as an intermediate node, and the other class provides direct transfer between devices.

A "credential server" can be likened to a server that sits in front of a repository where credentials can be securely stored for later retrieval. The credential server is active in the protocol, that is, it implements security enforcing functionality.

To transfer credentials securely from one end device to another is a straightforward two-step process. Users can have their credentials securely "uploaded" from one device, e.g., a wireless phone, to the credential server. They can be stored on the credential server, and "downloaded" when needed using another device; e.g., a two-way pager.

Some advantages of a credential server approach compared to credential transfer are:

1. It provides a conceptually clean and straightforward approach. For all end devices, there is one protocol, with a set of actions defined to transfer credentials from the device to the server, and another set of actions defined to transfer credentials from the server to the device. Furthermore, this protocol involves clients (the devices) and a server (the credential server), like many other Internet protocols; thus, the design of this protocol is likely to be familiar to most people familiar with most other Internet protocols.

2. It provides for a place where credentials can be securely stored for arbitrary lengths of time. Given a reasonable-quality server operating under generally accepted practices, it is unlikely the credentials will be permanently lost due to a hardware failure. This contrasts with systems where credentials are only stored on end devices, in which a failure of or the loss of the device could mean that the credentials are lost forever.

3. The credential server may be able to enforce a uniform security policy regarding credential handling. This is particularly the case where credentials are issued by an organization for its own purposes, and are not "created" by the end user, and so must be governed by the policies of the issuer, not the user.

However, the credential server approach has some potential disadvantages, too:

1. It might be somewhat expensive to maintain and run the credential server, particularly if there are stringent requirements on availability and reliability of the server. This is particularly true for servers which are used for a large community of users. When the credential server is intended for a small community, the complexity and cost would be much less.

2. The credential server may have to be "trusted" in some sense and also introduces a point of potential vulnerability. (See the Security Considerations section for some of the issues.) Good protocol and system design will limit the vulnerability that exists at the credential server, but at a minimum, someone with access to the credential server will be able to delete credentials and thus deny the SACRED service to system users.

Thus, some users may prefer a different class of solution, in which credentials are transferred directly from one device to another (i.e., having no intermediary element that processes or has any understanding of the credentials).

For example, consider the case where Mimi sends a message from her wireless phone containing the credentials in question, and retrieves it using her two-way pager. In getting from one place to another, the bits of the message cross the wireless phone network to a base station. These bits are likely transferred over the wired phone network to a message server run by the wireless phone operator, and are transferred from there over the Internet to a message server run by the paging operator. From the paging operator they are transferred to a base station and then finally to Mimi's pager. Certainly, there are devices other than the original wireless phone and ultimate pager that are involved in the credential transfer, in the sense that they transmit bits from one place to another. However, to all devices except the pager and the wireless phone, what is being transferred is an un-interpreted and unprocessed set of bits. No security-related decisions are made, and no actions are taken based on the fact that this message contains credentials, at any of the intermediate nodes. They exist simply to forward bits. Thus, we consider this to be a "direct" transfer of credentials.

Solutions involving the direct transfer of credentials from one device to another are potentially somewhat more complex than the credential-server approach, owing to the large number of different devices and formats that may have to be supported. Complexity is also added due to the fact that each device may in turn have to exhibit the behavior of both a client and a server.

We believe that both classes of solutions are useful in certain environments, and thus that the SACRED framework will have to define solutions for both. The extent to which elements of the above solutions overlap remains to be determined.

This all leads to our first set of requirements:

F1. The framework MUST support both "credential server" and "direct" solutions.
F2. The "credential server" and "direct" solutions SHOULD use the same technology as far as possible.

2.2 User authentication

There is a wide range of deployment options for credential mobility solutions. In many of these cases, it is useful to be able to re-use an existing user authentication scheme, for example where passwords have previously been established, it may be more secure to re-use these than try to manage a whole new set of passwords. Different devices may also limit the types of user authentication scheme that are possible, e.g., not all mobile devices are practically capable of carrying out asymmetric cryptography.

F3. The framework MUST allow for protocols which support different user authentication schemes

2.3 Credential Formats

Today there is no single standard format for credentials and this is not likely to change in the near future. There are a number of fairly widely deployed formats, e.g., [PGP], [PKCS#12] that have to be supported. This means that the framework has to allow for protocols supporting any credential format.

F4. The details of the actual credential type or format MUST be opaque to the protocol, though not to processing within the protocol's peers. The protocol MUST NOT depend on the internal structure of any credential type or format.

2.4 Transport Issues

Different devices allow for different transport layer possibilities, e.g., current WAP 1.x devices do not support TCP. For this reason the framework has to be transport "agnostic".

F5. The framework MUST allow use of different transports.

3. Protocol Requirements

In this section, we identify the requirements for secure credential-transfer solutions. We will begin by listing a set of relevant vulnerabilities and the requirements that must be met by all solutions. Then we identify additional requirements that must be met by solutions involving a credential server, followed by additional requirements that must be met by solutions involving direct transfer of credentials.

3.1 Vulnerabilities

This section lists the vulnerabilities against which a SACRED protocol SHOULD offer protection. Any protocol claiming to meet the requirements listed in this document MUST explicitly indicate how (or whether) it offers protection for each of these vulnerabilities.

V1. A passive attacker can watch all packets on the network and later carry out a dictionary attack.

V2. An attacker can attempt to masquerade as a credential server in an attempt to get a client to reveal information on line that allows for a later dictionary attack.

V3. An attacker can attempt to get a client to decrypt a chosen "ciphertext" and get the client to make use of the resulting plaintext - the attacker may then be able to carry out a dictionary attack (e.g., if the plaintext resulting from "decryption" of a random string is used as a DSA private key).
V4. An attacker could overwrite a repository entry so that when a user subsequently uses what they think is a good credential, they expose information about their password (and hence the "real" credential).
V5. An attacker can copy a credential server's repository and carry out a dictionary attack.
V6. An attacker can attempt to masquerade as a client in an attempt to get a server to reveal information that allows for a later dictionary attack.
V7. An attacker can persuade a server that a successful login has occurred, even if it hasn't.
V8. (Upload) An attacker can overwrite someone else's credentials on the server.
V9. (When using password-based authentication) An attacker can force a password change to a known (or "weak") password.
V10. An attacker can attempt a man-in-the-middle attack for lots
V11. User enters password instead of name.
V12. An attacker could attempt various denial-of-service attacks.

3.2 General Protocol Requirements

Looking again at the examples described in Section 1.1, we can readily see that there are a number of requirements that must apply to the transfer of credentials if the ultimate goal of supporting the Internet security protocols (e.g., TLS, IPSec, S/MIME) is to be met. For example, the credentials must remain confidential at all times; it is unacceptable for nodes other than the end-user's device(s) to see the credentials in any readable, cleartext form.

These, then, are the requirements that apply to all secure credential-transfer solutions:

G1. Credential transfer both to and from a device MUST be supported.
G2. Credentials MUST NOT be forced by the protocol to be present in cleartext at any device other than the end user's.
G3. The protocol SHOULD ensure that all transferred credentials be authenticated in some way (e.g., digitally signed or MAC-ed).
G4. The protocol MUST support a range of cryptographic algorithms, including symmetric and asymmetric algorithms, hash algorithms, and MAC algorithms.

G5. The protocol MUST allow the use of various credential types and formats (e.g., X.509, PGP, PKCS12, ...).
G6. One mandatory to support credential format MUST be defined.
G7. One mandatory to support user authentication scheme MUST be defined.
G8. The protocol MAY allow credentials to be labeled with a text handle, (outside the credential), to allow the end user to select amongst a set of credentials or to name a particular credential.
G9. Full I18N support is REQUIRED (via UTF8 support) [RFC2277].
G10. It is desirable that the protocol be able to support privacy, that is, anonymity for the client.
G11. Transferred credentials MAY incorporate timing information, for example a "time to live" value determining the maximum time for which the credential is to be usable following transfer/download.

3.3 Requirements for Credential Server-based solutions

The following requirements assume that there is a credential server from which credentials are downloaded to the end user device, and to which credentials are uploaded from an end user device.

S1. Credential downloads (to an end user) and upload (to the credential server) MUST be supported.
S2. Credentials MUST only be downloadable following user authentication or else only downloaded in a format that requires completion of user authentication for deciphering.
S3. It MUST be possible to ensure the authenticity of a credential during upload.
S4. Different end user devices MAY be used to download/upload/manage the same set of credentials.
S5. Credential servers SHOULD be authenticated to the user for all operations except download. Note: This requirement can be ignored if the credential format itself is strongly protected, as there is no risk (other than user confusion) from an unauthenticated credential server.
S6. It SHOULD be possible to authenticate the credential server to the user as part of a download operation.
S7. The user SHOULD only have to enter a single secret value in order to download and use a credential.
S8. Sharing of secrets across multiple servers MAY be possible, so that penetration of some servers does not expose the private parts of a credential ("m-from-n" operation).
S9. The protocol MAY support "away-from-home" operation, where the user enters both a name and a domain (e.g. RoamingStephen@baltimore.ie) and the domain can be used in order to locate the user’s credential server.

S10. The protocol MUST provide operations allowing users to manage their credentials stored on the credential server, e.g., to retrieve a list of their credentials stored on a server; add credentials to the server; delete credentials from the server.
S11. Client-initiated authentication information (e.g., password) change MUST be supported.
S12. The user SHOULD be able to retrieve a list of accesses/changes to their credentials.
S13. The protocol MUST support user self-enrollment. One scenario calling for this is where a previously unknown user uploads his credential without requiring manual operator intervention.
S14. The protocol MUST NOT prevent bulk initializing of a credential server's repository.
S15. The protocol SHOULD require minimal client configuration.

3.4 Requirements for Direct-Transfer Solutions

The full set of requirements for this case has not been elucidated, and this list is therefore provisional. An additional requirements document (or a modification of this one) will be required prior to progression of a direct-transfer protocol.

The following requirements apply to solutions supporting the "direct" transfer of credentials from one device to another. (See Section 2 for the note on the meaning of "direct" in this case.)

D1. It SHOULD be possible for the receiving device to authenticate that the credential package indeed came from the purported sending device.
D2. In order for a sender to know that a credential has been received by a recipient, it SHOULD be possible for the receiving device to send an acknowledgment of credential receipt back to the sending device, and for the sending device to authenticate this acknowledgment.

4. Security Considerations

4.1 Hardware vs. Software

Mobile credentials will never be as secure as a "pure" hardware-based solution, because of potential attacks through the operating system of the end-user device. However, an acceptable level of security may be accomplished through some simple means. In fact the level of security may be improved (compared to password encrypted files) through the use of SACRED protocols.

The platforms to which credentials are downloaded usually cannot be regarded as tamper-resistant, and it therefore is not too hard to analyze contents of their memories. Further, storage of private keys, even if they are encrypted, on a credential server, will be unacceptable in some environments. Lastly, replacement of installed or downloaded SACRED client software with a Trojan horse program will always be possible, such a program could email the username and password to the program's author.

4.2 Auditing

Although out of scope of the SACRED protocol development work, implementations should carefully audit events that may be security relevant. In particular credential server implementations should audit all operations and should include information about the time and source (e.g., IP address) of the operation, the claimed identity of the client (possibly masked - see below), the type and result of the operation and possibly other operation specific information. Implementations should also take care not to include security sensitive information in the audit trail, especially not sensitive authentication information.

It may be sensible to mask the claimed identity in some way in order to ensure that even if a user enters her password in a "username" field, that that information is not in clear in the audit trail, regardless of whether or not it was received in clear.

Similar mechanisms which should be supported, but which are out of scope of protocol development include alerts and account locking, in particular following repeated authentication failures.

4.3 Defense against attacks

Credential servers are major targets. Someone who can successfully attack a credential server might be able to gain access to the credentials of a number of users, unless those credentials are sufficiently protected (e.g., encrypted sufficiently that they cannot be read or used by someone who gains access to them). Attackers might also be able to substitute credentials of users, to carry out other system attacks (e.g., an attacker could provide a user with a "trusted root" credential that the attacker controls, which would later allow the attacker to have some other certificate accepted by the user counter to policy).

In addition, a credential server is a major target for denial of service attacks. Ensuring that a credential server is unavailable to legitimate users can be of great assistance to attackers. Users who were not able to retrieve needed credentials might be forced to

operate insecurely, or not operate at all. Credential servers are especially vulnerable to denial of service attacks if they do lots of expensive cryptographic operations - it might not take very many operations for the attacker to bring service to an unacceptable level.

Thus, great care should be taken in designing systems that use credential servers to protect against these attacks.

References

[PGP] Callas, J., Donnerhacke, L., Finney, H. and R. Thayer, "OpenPGP Message Format", RFC 2440, November 1998.

[PKCS12] "PKCS #12 v1.0: Personal Information Exchange Syntax Standard", RSA Laboratories, June 24, 1999.

[CMS] Housley, R., "Cryptographic Message Syntax", RFC 2630, June 1999.

[PKCS15] "PKCS #15 v1.1: Cryptographic Token Information Syntax Standard," RSA Laboratories, June 2000.

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2277] Alvestrand, H., " IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998.

[RFC2510] Adams, C. and S. Farrell, "Internet X.509 Public Key Infrastructure Certificate Management Protocols", RFC 2510, March 1999.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frysyk, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol - HTTP/1.1", RFC 2616, June 1999.

Acknowledgements

The authors gratefully acknowledge the text containing additional use cases in Appendix B that was supplied by Neal Mc Burnett (nealmcb@avaya.com).

Authors' Addresses

Alfred Arsenault
Diversinet Corp.
P.O. Box 6530
Ellicott City, MD 21042
USA

Phone: +1 410-480-2052
EMail: aarsenault@dvnet.com

Stephen Farrell,
Baltimore Technologies,
39 Parkgate Street,
Dublin 8,
IRELAND

Phone: +353-1-881-6000
EMail: stephen.farrell@baltimore.ie

Appendix A: A note on SACRED vs. hardware support.

One way of accomplishing many of the goals of the SACRED WG is to put the credentials on hardware tokens - e.g., smart cards, PCMCIA cards, or other devices. There are a number of types of hardware tokens today that provide secure storage for sensitive information, some degree of authentication, and interfaces to a number of types of wireless and other devices. Thus, in the second example from section 1.1, Will could simply put his private key on a smart card, always take the smart card with him, and be assured that whichever device he uses to retrieve his e-mail, he will have all of the information necessary to decrypt and read messages.

However, hardware tokens are not appropriate for every environment. They cost more than software-only solutions, and the additional security they provide may or may not be worth the incremental cost. Not all devices have interfaces for the same hardware tokens. And hardware tokens are subject to different failure modes than typical computers - it is not at all unusual for a smart card to be lost or stolen; or for a PCMCIA card to physically break.

Thus, it is appropriate to develop complementary software-based solution that allows credentials to be moved from one device to another, and provides a level of security sufficient for its environments. While we recognize that the level of security provided by a software solution may not be as high as that provided by the hardware solutions discussed above, and some organizations may not consider it sufficient at all, we believe that a worthwhile solution can be developed.

Finally, SACRED protocols can also complement hardware credential solutions by providing standard mechanisms for the update of credentials which are stored on the hardware device. Today, this often requires returning (with) the device to an administrative centre, which is often inconvenient and may be costly. SACRED protocols provide a way to update and manage credentials stored on hardware devices without requiring such physical presence.

Appendix B: Additional Use Cases

This appendix describes some additional use cases for SACRED protocols. SACRED protocols are NOT REQUIRED to support all these use cases, that is, this text is purely informative.

B.1 Home/Work Desktop Computer

Scenario Overview

A university utilizing a PKI for various applications and services on-campus is likely to find that many of its users would like to make use of the same PKI-enabled services and applications on computers located in their residence. These home computers may be owned either by the university or by the individual but are permanently located at the residence as opposed to laptop systems that may be taken home. The usage depicted in this scenario may be motivated by formal telecommuting arrangements or simply by the need to catch up on work from home in the evenings. The basic scenario should apply equally well to the commercial, health care, and higher education environments.

Assumptions

This scenario assumes that the institution has not implemented a hardware token-based PKI mobility solution

The home computer has a dial-up as opposed to a permanent network connection.

The PKI applications, whenever practical, should be functional in both on-line and off-line modes. For example, the home user signing an email message to be queued for later bulk sending and the reading of a received encrypted message may be supported off-line while composing and queuing of an encrypted message might not be supported in off-line mode.

Applications using digital signatures may require "non-repudiation".

The institution prefers that the user be identified via a single certificate / key-pair from all computers used by the individual.

The home computer system can not be directly supported by the institution’s IT staff. Hardware, operating system versions, and operating system configurations will vary widely. Significant software installation or specialized configurations will be difficult to implement.

Uniqueness of Scenario

vThe PKI mobility support needed for this scenario is, in general, similar to the other mobility scenarios. However, it does have several unique aspects:

1. The home-user scenario differs from the general public workstation case in that it provides the opportunity to permanently store the user's certificate and key-pair on the workstation.

2. Likewise the appropriate CA certificates and even certificates for other users can be permanently stored or cached on the home workstation.

3. Another key difference is the need to support off-line use of the PKI credentials given the assumed dial-up network connection.

4. The level of hardware and software platform consistency (operating system versions and configurations) will vary widely.

5. Finally, the level of available technical support is significantly less for home systems than for equivalent systems managed by the IT staff at the office location.

B.2 Public Lab / On-campus Shared Workstation

Scenario Overview

Many colleges and universities operate labs full of computer systems that are available for use by the general student population. These computers are typically configured with identical hardware and an operating system build that is replicated to all of the systems in the lab. Many typical configurations provide no permanent storage of any type while others may offer individual disk space for personal files on a central server. Some scheme is generally used to ensure that the configuration of the operating system is preserved across users and that temporary files created by one user are removed before the next user logs in. Students generally sit down at the next available workstation without any clear pattern of usage.

The same basic technical solutions used to operate public labs are often also used in general environments where several people share a single workstation. This is often found in locations with shift work such as medical facilities and service bureaus that provide services to multiple time zones.

Assumptions

1. This scenario assumes that the institution has not implemented a hardware token-based PKI mobility solution.

2. The computer systems are permanently networked with LAN connections.

3. The configuration of the computer system is centrally maintained and customizations are relatively easy to implement. For example it would be easy to load enterprise root certificates, LDAP server configurations, specialized software, and any other needed components of the PKI on to the workstations.

4. Applications using digital signatures may require "non-repudiation" in some of the anticipated environments. Examples of this might include homework submission in a public lab environment or medical records in a health care environment.

5. The institution prefers that the user be identified via a single certificate / key-pair from all computers used by the individual.

6. Many anticipated implementations of this scenario will not implement any user authentication at the desktop operating system level. Instead, user authentication will occur at during the startup of networked applications such as email, web-based services, etc. Login at the desktop level may be with generic user names that are more targeted at matching printouts to machines than identifying users.

7. Users, with almost ridiculous frequency, will walk away from a system forgetting to first logout from running authenticated applications.

Uniqueness of Scenario

The PKI mobility support needed for this scenario is, in general, similar to the other mobility scenarios. However, it does have several unique aspects:

1. Unlike situations with personal workstations, there is no permanent storage available to hold user key pairs and certificates.

2. Appropriate CA certificates and custom software are easily added and maintained for these types of shared systems.

3. The workstations are installed in public locations and users will frequently forget to close applications before permanently walking away from the workstation.

B.3 Public Kiosk Mobility

Overview

This scenario describes the needs of the traveler or the shopper. This person is traveling light (no computer) or is burdened with everything but a computer. It recognizes the increasing availability of Internet access points in public spaces, such as libraries, airports, shopping malls, and "cyber cafes".

The Need

In our increasingly mobile society, the chances of needing information when away from the normal computing place are great. One may need to look up a telephone number. Have you tried to find a phone book at a public phone lately? It may become necessary to use a data device to find the next place to rush to. With the proliferation of wireless devices (electronic leashes), others have the ability to create a need for quick access to electronic information. A pager can generate a need to check the email inbox or address book. A cell phone can drive you to your database to answer a pressing question.

The ability to quickly access sensitive or protected information or services from publicly available devices will only become more necessary as we become more and more "connected".

The Device

The access device is more a function of the best discount or marketing effort than of design. Any number of hardware platforms will be encountered.

Since these devices are open to the public I/O ports are not likely to be. In order to protect the device and its immediate network environment, most devices will be in some sort of protective container. Access to serial, parallel, USB, firewire, SCSI, or PCMCIA connections will not be possible. Likewise floppy, zip, or CD drives. Therefore, any software "token" must be obtained from the network itself.

The Concerns

1. Getting the "token". Since it will be necessary to obtain the token (key, certificate, credential) from across the network. How can it be protected during transit?

2. Where did you get it? One of the primary controls in PKI is protection of the private key. Placing the key on a host that is accessible from a public network means that there is an inherent exposure from that network. The access controls and other security measures on the host machine are an area of concern.

3. How did you get it? When you obtained the token from the server, how did it know that you are you? Authentication becomes critical.

4. What happens to the token when you leave? You've checked your mail, downloaded a recipe from that super-secure recipe server, found out how to get to the adult beverage store for the... uh... accessories... for the meal, and you're off! Is your token? Or is it still sitting there on the public kiosk waiting for those youngsters coming out of the music store to notice and cruise the information highway on your ticket?

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.