A Location-Based Mechanism for Mobile Device Security

Wayne Jansen
The National Institute of Standards and Technology

Vlad Korolev
Booz-Allen Hamilton

Abstract

While mobile handheld devices, such as cell phones and PDAs, provide productivity benefits, they also pose new risks. A vital safeguard against unauthorized access to a device's contents is authentication. This paper describes a location-based authentication mechanism that employs trusted servers called policy beacons, which are used to provide location data and control device behavior. Mobile devices determine their proximity to available policy beacons and upon validation assume the designated organizational policy. The mechanism is designed to take advantage of Bluetooth functionality built into many current handheld devices used in organizational settings.

1. Introduction

With the trend toward a highly mobile workforce, the use of handheld devices such as smart phones and Personal Digital Assistants (PDAs) continues to grow. These devices are relatively inexpensive productivity tools that have become a part of the computing infrastructure for many organizations. While such devices have limitations, they are nonetheless extremely useful in managing appointments and contact information, reviewing documents and spreadsheets, corresponding via electronic mail and instant messaging, delivering presentations, accessing remote corporate data, and handling voice calls. Over the course of use, significant amounts of sensitive information can accumulate on them. Remote access to organizational resources via wireless and wired communications capabilities may also be enabled. These conditions create a potential target of attack.

One of the most serious security threats to a computing device is unauthorized use. User authentication is the first line of defense against this threat. Authentication using passwords is perhaps the best-known example of a proof by knowledge mechanism. Other classes of authentication mechanisms include proof by possession (e.g., smart cards) and proof by property (e.g., fingerprints). Two additional factors that can apply to each class of authentication mechanism are location and time of day. They refer respectively to whether the authentication is being attempted at either an acceptable location or an acceptable time. The mechanism described in this paper involves location as a facet of user authentication. That is, the mechanism is intended to be used in conjunction with a proof by knowledge, proof by possession, or proof by property authentication mechanism.

Establishing location benefits user authentication in several important ways:

- If a user attempts to authenticate from an unauthorized location, an authentication mechanism can reject the attempt.
- If a user attempts to authenticate from a location outside a defined boundary, the authentication framework can require that additional authentication mechanisms are satisfied before granting access.
- If a user instantiates a new activity, such as accessing a specialized application, the authentication framework can require that access to the functionality and related data be conducted from within an appropriate location.
- If a user moves within or outside of a defined boundary, an authentication mechanism can be triggered automatically to grant or deny access.

This paper provides an overview of a location-based authentication mechanism involving policy beacons, which are designed to support handheld devices. The paper describes how the beacon is used to authenticate the user of a handheld device and provides other details of the solutions' design and implementation. [1]

2. Background

Physical location systems employ various types of sensors that come in many different shapes and sizes and use different techniques for determining position. Physical location systems under consideration

[1] Certain commercial products and trade names are identified in this paper to illustrate technical concepts. However, it does not imply a recommendation or an endorsement by NIST.

typically have two kinds of components: appliances and infrastructure. An appliance is the equipment associated with an entity (e.g., a Global Positioning System (GPS) receiver or cell phone), while the infrastructure is the set collection of sensor equipment, usually fixed, which needs to be in place for the appliances to function (e.g., GPS satellites or mobile phone towers) [1]. A communications medium through which the devices and infrastructure communicate is also implicitly required. Location systems do exist in which the user carries no appliance and the solution relies entirely on infrastructure components (e.g., infrared cameras or floor sensors), but they are outside the scope of this paper.

In general, location can be treated in two ways: by position, where geographical or other physical coordinates of a unit are resolved to some degree of accuracy, or by proximity, where a unit's presence, relative position, or absence within an area is determined. Determining positional coordinates typically requires an extensive sensor infrastructure able to cooperate with an appliance to estimate position algorithmically through monitored signals. Determining proximity, while less precise, typically requires a less extensive infrastructure.

Position sensors attempt to provide the coordinates of an appliance relative to a coordinate system. The coordinate system may be fixed and global (e.g., the latitude, longitude and altitude reported by a GPS receiver), or mobile and local (e.g., "3 meters to my right"). Proximity sensors are less exact (e.g., within close or distant range of a sensor) [1]. Proximity sensors with overlapping detection regions can be used to calculate position, using triangulation or trilateration. Different sensors can have different resolutions and associated errors, ranging from centimeters to tens of meters [1, 2, 3]. They may also operate over different ranges and be limited to indoor or outdoor use.

Two classes of solutions for resolving location exist. The first is where location information is initially known only by the appliance, but not the infrastructure. The second is the opposite by which location information is initially known only by the infrastructure, and then furnished to the appliance [4].

The first class of solutions makes the appliance more independent of infrastructure components and services. However, it requires the appliance to be compatible with the infrastructure and powerful enough to make the needed computations and access control decisions. The second class of solutions is less demanding on the appliance, since it does not have to be powerful enough to perform such computations and access control decisions, relying instead on infrastructure components (e.g., RFID or the Active Bat [5]). Pervasive systems fall into this latter category, since by their very nature they are context-aware, with one type of context information being location information gathered from a variety of location sources and sensors [4].

The authentication mechanism described in this paper follows the first class of solutions. Location is resolved by the appliance through proximity, which is determined using information from a small number of policy beacons that make up the infrastructure (i.e., as few as one). The authentication mechanisms are organizationally oriented, and require only that participating handheld devices, which function as the appliances, support a common personal area network standard for wireless communications (i.e., Bluetooth). The mechanism is designed to establish the relative location of a mobile device with respect to a trusted beacon that, once discovered, serves as a security token, which is contacted periodically to reconfirm presence and to verify authenticity.

3. Overview

The policy beacon is a small device placed in an area to establish a perimeter where a distinct organizational policy is in effect. To accomplish this, the policy beacon offers an area location service for discovery and use by mobile devices. One or more policy beacons define the area. Location is determined relative to a beacon. Mobile devices equipped with the policy beacon authentication mechanism sense policy beacons active in the area and, if relevant, adjust their security policy settings accordingly. A device is either in or out of the vicinity of a beacon as determined by the footprint of the beacon's communications signals. Policy setting changes may enable or disable functionality on the mobile device. For example, certain network applications may be allowed to execute and protected repositories on the mobile device may become accessible. Figure 1 illustrates the policy transition of a device as it moves into the range of a policy beacon.

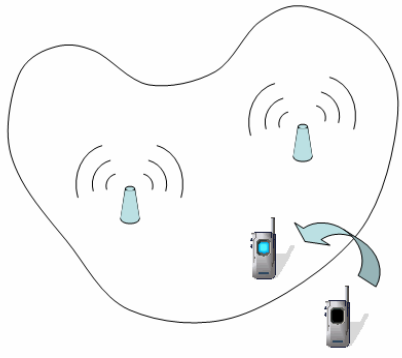

Figure 1: Device Entering the Footprint of a Policy Beacon

The policy beacon authentication mechanism on a mobile device checks periodically for proximity to a policy beacon. It reports successful authentication if a beacon is detected and able to be verified; otherwise, it reports failure. Multiple organizational beacons can be used to improve service levels above that of a single beacon, or arranged to service a larger area. An organizational beacon provides credential information for a device to verify using the Transport Layer Security (TLS) protocol over Bluetooth. Many mobile devices are manufactured with built-in Bluetooth radios, which allow short-range communication and have low power consumption. The solution could also be adapted for other types of wireless personal area network communications technologies.

4. Operation

The policy beacon authentication mechanism is designed as a client-server system. The policy beacon operates as a server to the client software on the mobile device, listening to inquiries from the client and responding as needed. The beacon proves its identity to mobile devices, but does not require mobile devices to do the same. The beacon establishes its identity via TLS using its private key and associated certificate. The beacon's server certificate must be valid and be issued by the organization's root certificate authority (CA) or by a CA having a valid certificate chain from the organization's root CA.

The mobile device plays an active role in the authentication mechanism. It must be configured to run the client software in the manner described below. The mobile device must also hold the public key of the organization's certificate authority to verify the authenticity of the beacon server certificate.

The policy beacon client on the mobile device operates in two distinct modes: unauthenticated and authenticated. In unauthenticated mode, the following steps occur:

- The mobile device periodically scans for the available policy beacons in the area.
- When the mobile device finds a potential beacon, it establishes a Bluetooth connection to it, and then attempts to set up a secure TLS connection over that physical channel, using the X.509 certificate supplied by the beacon.
- If the beacon is successfully authenticated and the TLS connection established, the mobile device enters a readiness exchange with the beacon to verify that it is indeed a functional policy beacon.
- Once the mobile device determines that the policy beacon is functional, the device enables the associated policy for that location and switches to authenticated mode.
- Otherwise, the mobile device blacklists the beacon for a period of time and retries the above steps.

Once in authenticated mode, the following steps occur:

- The mobile device periodically tries to reestablish a TLS connection over Bluetooth with the last beacon it previously used.
- If the beacon is again successfully authenticated and the TLS connection established, the mobile device verifies that the beacon is still functional.
- Once the mobile device determines that the beacon is functional, the device maintains the associated policy for that location and remains in authenticated mode.
- Otherwise, the mobile device retries the above steps, allowing for a momentary out of range condition.
- If the beacon cannot be authenticated and vetted within a preset time (approx. 2-3 minutes), the mobile device switches to unauthenticated mode and changes policy accordingly.

The Bluetooth device class identifier on the beacon is set to a specific value defined for beacon class devices. Using a customized device class improves performance, since the mobile device client can filter out other types of devices that may be present in an area (e.g., access points, or printers) and avoid unneeded and unwanted interaction.

Beacons support specific policies, denoted by an identifier in their credentials. A set of beacons may be configured to support distinct policies for different areas. A mobile device client for the policy beacon authentication mechanism is configured to observe a specific policy in the presence of an associated beacon. Doing so allows it to disregard beacons that identify other policies.

5. Mobile Device Implementation

The prototype mobile device authentication mechanism was implemented in C and C++ on an iPAQ Personal PDA, running the Familiar distribution of the Linux operating system and the Open Palmtop Integrated Environment (OPIE). Since the authentication mechanism is intended to operate in an environment that supports multi-mode authentication, the Familiar distribution was modified with MAF, a framework for multimode authentication [6]. The framework includes a policy enforcement engine that allows the behavior of code modules and device users to be governed via policy rules, and the facility to add new authentication mechanisms and have them execute in a prescribed order [7].

Table 1 illustrates the basic policy schema for MAF mechanisms. One or more authentication mechanisms can be configured to execute at levels 1 and above on a

mobile device. Successfully satisfying the mechanism at any level causes the associated security policy at that level to come into effect. The authentication levels are hierarchical, such that a lower-level mechanism must be satisfied before a higher-level mechanism can be encountered, while policies relationships can be independently specified.

Table 1 MAF Policy Schema

Authentication Level	Required Authentication	Effective Policy
Level 3	Zero or More Mechanisms	Policy C
Level 2	Zero or More Mechanisms	Policy B
Level 1	Zero or More Mechanisms	Policy A
Level 0	None—default at power on and boot up	Most Restrictive

A typical configuration corresponding to that on most devices would be a password mechanism at level 1. For the prototype implementation, the policy beacon authentication mechanism was configured to execute at level 2. The level 2 policy enabled more capabilities than at level 1, to represent entry into a trusted environment. This setup is characteristic of organizations, such as medical facilities, which need to ensure that access to specific applications on the mobile device and internal resources are limited to devices that are on-site.

MAF authentication mechanisms consist of two parts: an authentication handler, which embodies the procedure that performs the actual authentication, and a user interface, which performs all necessary interactions with the user. An authentication handler can be non-polling or polling to accommodate mechanisms that respectively grant total acceptance to the user once they are satisfied or instead require continual satisfaction to grant and maintain acceptance to the user.

The handler for the policy beacon is a polling mechanism, periodically awakening to perform the necessary operations. In unauthenticated mode, the handler periodically performs a Bluetooth inquiry to find potential beacons. If inquiry process results in finding a Bluetooth device with the beacon class identifier, the handler attempts to establish a Logical Link Control and Adaptation Protocol (L2CAP) connection to the predetermined Protocol Service Multiplex (i.e., a designator similar to an Internet port number) [8]. When the L2CAP connection is established, the handler tries to set up a TLS session over the connection and verify the server's X.509 certificate. If the verification succeeds, the handler switches to authenticated mode, where it periodically tries to establish a connection with the last known beacon and authenticate it using the same steps as above. If the handler eventually is unable to communicate and verify the last known beacon during the allotted interval, it switches back to unauthenticated mode.

The handler maintains a table of potential beacons to carry out its function. The table contains information about all Bluetooth devices in the vicinity of the mobile device. The table has the following fields: Media Access Control (MAC) Address, Last Seen, Last Contact and Status, as shown in Table 2.

Table 2: Potential Beacon Table

MAC Address	Last Seen	Last Contact	Status
00:02:92:21:AB:C8	20	20	Beacon
00:22:11:22:33:11	30	30	Not Beacon
00:22:99:11:11:11	20	20	Unknown

The MAC Address field contains the address of the Bluetooth device, while the fields Last Seen and Last Contact contain the time value of when the device was last seen and when the last successful communication with the device took place. The Status field contains the handler's idea of the device's purpose. The Status field can be one of the following: 'Beacon,' 'Not Beacon' and 'Unknown.' When the remote Bluetooth device is initially entered into the table, it is assigned the 'Unknown' status. Later, when a successful exchange with the remote device takes place, the device is assigned the 'Beacon' status. If the handler can establish a connection to the remote device, but the device does not follow the readiness protocol, the device is assigned the 'Not Beacon' status.

The handler populates the table by performing a Bluetooth inquiry process every 50 seconds. The inquiry discovers Bluetooth devices in the vicinity and returns a list of their MAC addresses. The handler looks up each MAC address received during the inquiry process to see if it already exists in the beacon table. If the address does not exist, it is entered into the table. For every MAC address received during the inquiry process, the handler updates the corresponding Last Seen entry in the handler table.

When the handler is not doing an inquiry, it tries to contact the devices in the beacon table whose Status entry contains either 'Beacon' or 'Unknown.' The devices with 'Beacon' status are contacted before the devices with 'Unknown' status. During the contact, the handler first tries to establish the L2CAP connection to the remote device. The Last Contact value is updated before every attempt to establish an L2CAP connection is made. If the connection

succeeds, the handler performs a TLS exchange. If a failure occurs after the L2CAP connection has been established, the handler sets the Status field of that beacon to 'Not Beacon,' which temporarily blacklists the beacon. If the TLS exchange results in successful authentication, the handler sets the Status to 'Beacon,' sets the lastAuthentication variable to the current time, and does not attempt further contact with the other devices in the table.

The lastAuthentication variable is used to determine whether the current authentication is still valid. If the time value stored in this variable is less then 120 seconds before the current time, the handler considers the state to be unchanged, remaining valid. When the kernel sends an authentication request to the handler, the handler checks the current time and the value of the lastAuthentication variable. It returns a positive response, if the value is within 120 seconds of the current time; otherwise it responds with negative authentication.

The handler periodically sweeps the beacon table for stale entries. If the handler sees an entry with the Last Seen value older than 60 seconds, the entry is removed from the table. The handler uses the Last Contact column in conjunction with the Status column to prevent permanent blacklisting of beacons that did not correctly follow the beacon readiness protocol previously. For example, it could be the case that the beacon was just booting up and not all the software was fully operational and able to complete the exchange. When the Status column for a particular entry contains a 'Not Beacon' value and the Last Contact time value is older than 20 seconds, the handler changes the Status value to 'Unknown.'

6. Policy Beacon Implementation

The prototype policy beacons were implemented using Intrinsyc CerfCube, embedded Linux devices. They come configured with the Familiar Distribution of Linux, including device drivers for all on-board peripherals. Peripheral support includes Ethernet and several serial ports. A Compact Flash slot supports Type I and II cards, which can be used to add Bluetooth or other wireless communications support. Other similarly-configured hardware platforms could be used in place of CerfCubes.

The policy beacon side of the implementation is less complicated than the mobile device side. The beacon software is a basic server that listens to incoming L2CAP connections. Once a connection occurs, it establishes a TLS protocol connection and observes its part of the readiness protocol, which involves a three-way handshake. The beacon can accept only one connection at the time. However, since the TLS exchange takes significantly less time than the Bluetooth connection time out, at least two devices can easily connect during that period.

The Bluetooth stack on the beacon is configured to respond to incoming inquires and connections, known respectively as inquiry scan and page scan modes. Both the mobile device and policy beacon manage the Bluetooth specific aspects of the communication, such as establishing and tearing down connections, determining the Message Transmission Unit (MTU) size, etc., as well as actual data transmission. Both also use the OpenSSL library to provide the TLS protocol functionality [9].

7. Safeguards

The authentication mechanism requires that the beacon is kept both physically and logically secure and situated at the correct location it identifies. When the authentication mechanism receives a message from a beacon, it must ensure that the message was created recently for the particular purpose intended and by the beacon claiming to have sent it. The mechanism must be able to detect when a message has been modified or forged by an attacker with access to the wireless network, or when a message issued previously (or for a different purpose) is being replayed on the network by an attacker. For these reasons, the organizational beacon handler uses the TLS protocol to authenticate potential beacons. The TLS protocol provides assurance that the beacon is genuine.

Besides the TLS protocol, the authentication mechanism relies on MAF for its protection. The substitution or overwrite of the authentication handler program is prevented by MAF functionality and the underlying operating system. The policy enforcement functionality of MAF is used also to protect the following security-related files and to grant the handler exclusive access:

- The X.509 certificate of the root CA used to validate the server's certificate – installed through security administration.
- The policy identifier observed by the authentication handler – installed through security administration.

Blocking access to the CA's public key certificate and the governing policy identifier prevents an attacker from substituting them with those from a different organization to gain unauthorized access to the mobile device.

8. Related Work

Little work on localization-based authentication for mobile devices appears in the literature. Localization efforts in sensor networks share some aspects, since size and cost concerns discourage the use of complex

hardware at the sensor nodes and power concerns limit transmission range. However, the goals are quite different with respect to using location for authentication purposes (e.g., [10]).

ZoneIT is a system designed to selectively control the functionality of mobile phones, including phone ringing and audio and visual recording capabilities, inappropriate for public settings, such as cinemas, museums, and concert halls [11]. The design relies on the wireless capabilities built into mobile phones and the use of a fixed beacon. Users would voluntarily download the required software to the mobile phone to communicate with beacon, which would be provided by the venue operator. The system was not designed for environments such as hospitals that would typically need a more robust means to disable functionality and enforce a broader range of policy controls. In the prototype implementation, a Bluetooth L2CAP connection is used to send simple control settings to mobile phones to disable services. A challenge-response exchange is used to authenticate a beacon.

SmartProfiles is a software application somewhat similar to ZoneIT insofar as it controls the ringing of an enabled smart phone [12]. It does not, however, control other device capabilities, and the settings are not determined through exchanges with a beacon, but instead by a predetermined schedule. The schedule can be set up by time of day (e.g., silent 11 PM-7 AM weekdays) and also through calendar entries (e.g., silent during 1 PM meeting).

9. Conclusion

Along with the productivity benefits provided by mobile devices also come new risks. This paper explains how proximity-based authentication can be implemented to reduce risks. The approach provides an organization with the ability to allow device users to perform their tasks within boundaries defined through configuration settings and policy controls, while limiting capabilities outside of those bounds. The authentication mechanism requires only a simple support infrastructure comprising one or more policy beacons. Although the mechanism was intended primarily for smart phones and PDAs, it would also be suitable for use with laptop and notebook computers and other transportable devices.

10. References

[1] J. Indulska, P. Sutton, "Location management in Pervasive Systems," Workshop on Wearable, Invisible, Context-Aware, Ambient, Pervasive and Ubiquitous Computing, Adelaide, Australia, February 2003.

[2] J. Hightower, G. Borriello, "Location Systems for Ubiquitous Computing," IEEE Computer, 34, 8, August 2001.

[3] M. Hazas, H.Scott, J. Krumm, "Location-Aware Computing Comes of Age," IEEE Computer, 37, 2, February 2004.

[4] M. Gruteser, G. Schelle, A. Jain, R. Han, D. Grunwald, "Privacy-Aware Location Sensor Networks," USENIX 9th Workshop on Hot Topics in Operating Systems (HOTOS IX), Lihue Hawaii, May 2003, pp. 163-167.

[5] A. Ward, A. Jones, A. Hopper, "A New Location Technique for the Active Office," IEEE Personal Communications, Vol. 4 (5), October 1997, pp 42-47.

[6] W. Jansen, V. Korolev, S. Gavrila, T. Heute, C. Séveillac, "A Framework for Multimode Authentication: Overview and Implementation Guide," NISTIR 7046, The National Institute of Standards and Technology, August 2003, http://csrc.nist.gov/publications/nistir/nistir-7046.pdf

[7] W. Jansen, V. Korolev, S. Gavrila, C. Séveillac, "A Unified Framework for Mobile Device Security," The International Conference on Security and Management (SAM'04), Las Vegas, USA, June 2004, http://csrc.nist.gov/groups/SNS/mobile_security/documents/mobile_devices/PP-UNIsecFramework-fin.pdf

[8] *Bluetooth Core Specifications*, Version 2.1 + EDR, Volume 3, Bluetooth SIG, Inc., July 26, 2007, http://www.bluetooth.com/NR/rdonlyres/F8E8276A-3898-4EC6-B7DA-E5535258B056/6545/Core_V21_EDR.zip

[9] "OpenSSL SSL/TLS Library," manual page, OpenSSL Project, http://www.openssl.org/docs/ssl/ssl.html

[10] N. Lane, H. Lu, A. Campbell, "Ambient Beacon Localization: Using Sensed Characteristics of the Physical World to Localize Mobile Sensors," ACM, Workshop on Embedded Networked Sensors, Cork, Ireland, June 2007.

[11] T. Moors, M. Mei. A. Salim, "Using Short-range Communication to Control Mobile Device Functionality," Personal and Ubiquitous Computing, Springer-Verlag, Vol. 12, Issue 1, January 2008.

[12] "SmartProfiles," SymbianWare, November 20002, http://www.symbianware.com/product.php?id=sprofiles&lang=en&pl=all&s=0