On Securing the Public Health Information Network Messaging System

Barry Rhodes
Associate Director For Public Health System Development
Information Resources Management Office
Centers for Disease Control and Prevention
www.cdc.gov

Rajashekar Kailar
Chief Technology Officer
Business Networks International, Inc.
www.bnetal.com

Abstract

The Public Health Information Network Messaging System (henceforth, PHINMS) is the Centers for Disease Control and Prevention's (CDC) implementation of the ebXML 2.0 messaging standards [EbXML]. This system was developed for the purpose of secure and reliable messaging over the Internet. This software has been widely deployed by CDC and its public health partners, including state and local health departments, and healthcare providers. PHINMS is designed to leverage X.509 digital certificates issued by public key infrastructures, but does not require a single, universal PKI. In this paper we discuss some of the security aspects of PHINMS.

Introduction

The Public Health Information Network Messaging System (PHINMS) is a CDC developed implementation of existing standards for the secure and reliable transmittal of messages across the Internet.

The PHINMS relies on ebXML, XML encryption [XMLENC], XML Digital Signature [XMLDSIG], SOAP [SOAP] and other standards. PHINMS is the primary message transport system for the National Electronic Disease Surveillance System [NEDSS], the Laboratory Response Network [LRN], National Health Safety Network [NHSN] and various other public health preparedness programs within CDC.

By design, PHINMS is message data (payload) independent; hence it can be used to transport any type of data (e.g., text, binary).

PHINMS is operating system neutral since it is implemented using Java and J2EE standards.

Further, it provides language neutral, queue based interfaces for sending and receiving messages. The preferred queue implementation is an ODBC/JDBC compliant database table, but support for queues based on XML file descriptors also exists. PHINMS supports peer-to-peer messaging, as well as messaging via a third party using a send and poll model.

Message data security is accomplished using a combination of encryption, end-point authentication, and access control techniques. Transport reliability is accomplished using message integrity verification, transmission retries and duplicate detection on message receipt.

Since PHINMS is used to transport sensitive data over public un-trusted networks (e.g., Internet), it is important to make sure that end-points trust each other, are able to identify and authenticate each other, and that communication channels preserve data confidentiality and integrity. Further, access to data sent and received should be controlled.

The balance of this paper will focus on some of the security considerations that went into the design and implementation of PHINMS.

Security Considerations

Several security considerations went into the design, implementation and deployment of PHINMS. The following is a brief description:

Trust[1]

Secure messaging over public un-trusted networks requires messaging parties to be able to identify, authenticate and trust each other. For this, firstly, real world trust relationships need to be established between messaging organizations. This may include establishing written agreements on service levels, liabilities, etc., pursuant to OMB guidance on the Government Paperwork Elimination Act (GPEA) as well as the Electronic Signatures in Global and National Commerce Act (E-SIGN). Further, business processes for creating and handling messages at each end of the messaging pipe need to be put in place. Once trust and business relationships are established in real world terms, electronic collaboration agreements can be setup for message transport and processing. This includes setting up relationships to trust certification authorities and the identity of the sending and receiving components (e.g., using access control lists).

In the centralized trust scenario, a central node performs identity binding and security credentialing, and all nodes establish trust relationships with a central node. In this case, assuming *n* nodes, only $O(n)$ trust relations are needed. However, in a heterogeneous environment where trust is de-centralized, with *n* nodes, each node may need to establish a trust relationship and security credentials with every other node, and in the worst case scenario $O(n^2)$ trust relationships may be needed. Since messaging nodes typically belong to autonomous organizations and realms, establishing a globally accepted central identity and trust authority may not be politically acceptable. In a purely PKI based authentication framework, a trust bridge such as the Federal

Bridge CA could be used to address this problem. However, while PHINMS supports PKI based authentication, it also supports other modes of authentication, such as basic or custom authentication.

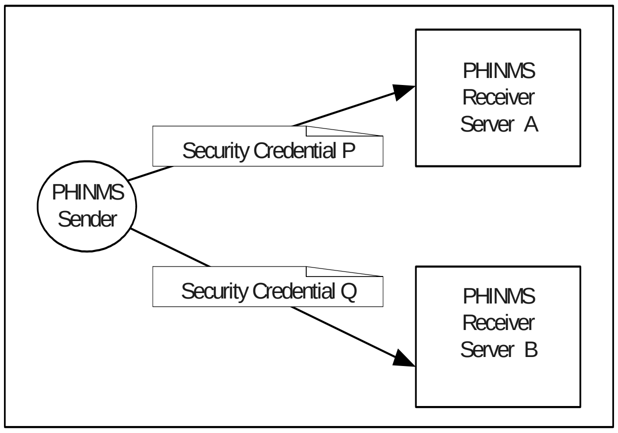
[1] "Trust" in this context is more generic than what is involved in PKI based certificate chain validation. In particular, it may involve other (non-PKI) authentication mechanisms (e.g., basic or form based authentication).

PHINMS is designed to support both centralized and de-centralized trust models. Decisions on identity binding and security credentialing are made by the deploying organizations. Decisions on trusting the identity and security credentials are made mutually between messaging parties at the time when electronic collaborations are created.

Identification, Authentication and Authorization

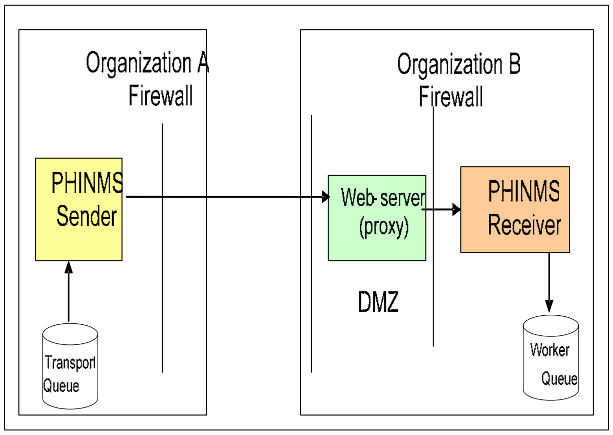
Identification and authentication in messaging is a difficult topic and is one that is far from mature. Since the message is typically sent by a process and not necessarily triggered by an individual, the authentication dialog must be scriptable. That is, the sending application must be able to negotiate the exchange of credentials without human intervention. This is only possible for certain security tokens (e.g., hardware based one time passwords and biometric identities don't lend themselves to this kind of scripted authentication exchange).

PHINMS supports automated authentication dialogs for client-certificate based authentication over SSL, basic authentication, and form based authentication. The method used for mutual and automated identification and authentication between messaging parties is part of the electronic agreement between them, and should be established upfront, after the real world trust relationship has been established.



Each messaging node in the Public Health Information Network (PHIN) is identified by a globally unique identifier. As shown in the diagram above, a messaging node (i.e., PHINMS sender) may contain one or more security credentials that allow it to conduct automated authentication dialogs with other messaging nodes. In the absence of a universally trusted authority to issue security identities and credentials, potentially, a different security identity and set of credentials may be needed for the purpose of authenticating to each message destination. The security credentials may include client certificates (key-stores), passwords, etc. Managing these security credentials can be a daunting task for the messaging administrator in the face of expiring certificates, password renewals etc. Of course, certificates can be issued with an expiration period of years, whereas passwords typically must be changed every 90 days, so the problem with the latter is far more daunting.

The recommended architecture for PHINMS messaging is one where the PHINMS receiver components are protected from direct access from the Internet, by web-server proxies as shown in the diagram at the top of the next column:



The web-server proxies typically reside in the organization’s DMZ, and mediate all inbound traffic for the PHINMS receiver server, authenticating the sending process. SSL with client-certificate based authentication is the preferred method of authentication for PHINMS, since it is a well established standard and is widely implemented by web-server proxies.

Once the message sender is authenticated, it is the responsibility of the receiving organization’s web-server proxy to ensure that an authenticated sender only gains access to the receiver URL. At this time, PHINMS does not provide support for attribute certificates which can be used for authorization decisions. Authorization information is stored on the receiver server, and enforced by the web-server proxy based on the authenticated identity of the PHINMS senders.

Authentication Factors

For interactive authentication dialogs over the Internet, generally, two factor authentication[2] is considered stronger and more secure than single factor authentication.

[2] Authentication mechanisms typically use secrets such as what a user knows (e.g., password), what a user has (e.g., hardware token) and what a user is (e.g., thumbprint). These are called authentication factors. For strong authentication, a combination of two of these three factors is used.

However, in the case of B2B automated security dialog, the security value of two-factor authentication is significantly diminished, since there is no real user behind the authentication dialog.

All user factors required for the authentication dialogs would need to be pre-configured into the software that initiates the authentication handshake. Further, at the time of this writing, there are no published and accepted Internet standards for two factor authentication in B2B transactions. While it is possible to use hardware based security modules (sometimes called HSM) to emulate additional authentication factors for B2B exchanges, such mechanisms require additional hardware and management complexity.

Confidentiality

Since communication is over un-trusted public networks, protecting its confidentiality is important. PHINMS uses payload level asymmetric encryption for end-to-end persistent confidentiality. The XML encryption standard The XML encryption standard [XMLENC] is used for encrypting the payload.

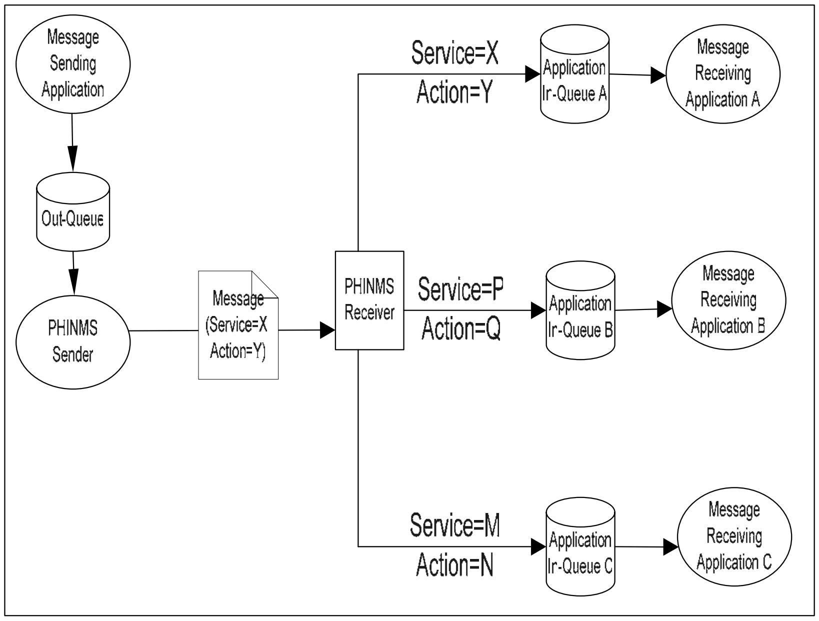
In the case of store and forward messaging, data is protected from being read by intermediaries by using asymmetric encryption using the public key of the message recipient to encrypt a random symmetric key, which in turn encrypts the data. Additionally, communication is typically conducted over a Secure Sockets Layer (SSL) channel, ensuring that the message meta-data is also protected. To ensure end-to-end confidentiality, the channel between the web-server proxy and the application server is also over SSL.

Integrity and non-repudiation

PHINMS supports the use of XML digital signatures [XMLDSIG] for message integrity and non-repudiation of message data. Signing certificates can be sent as part of the signature meta-data facilitating verification of the signature, alternatively, signing certificates can be statically pre-configured at the receiving node. Additionally, communication is typically conducted over SSL with client-certificate based authentication, which provides further message integrity and non-repudiation assurances.

Access control

The ebXML messaging standard supports message labels called “Service” and “Action”. These XML tags are part of the message envelope, and can be mapped to a service on the receiving node These XML tags are part of the message envelope, and can be mapped to a service on the receiving node.



In the PHINMS implementation, messages that are received using the receiver server are stored in database tables (queues) based on their Service and Action tags. These queues are the equivalent of an application “inbox”, and each application can only access its own inbox.

Public key infrastructure (PKI)

PHINMS is designed to leverage a PKI, but it does not require a universal PKI. For instance, a PHINMS sending client can use a client certificate issued by one certification authority (CA) to authenticate itself to a PHINMS receiver server, and use a client certificate issued by a different CA to authenticate itself to a different PHINMS receiver server. Currently, PKI trust relations are statically defined at the time when collaboration is established and configured between messaging entities. This is sometimes called the “Certificate Trust List” model. Ideally, public key certificates are published in an LDAP directory (need not be centralized), but PHINMS also supports a web-service interface to publish and retrieve certificates. As a third alternative, encryption public key certificates can be distributed out of band and pre-configured at the message sending nodes.

Public key certificates can be published in de-centralized LDAP directories as well.

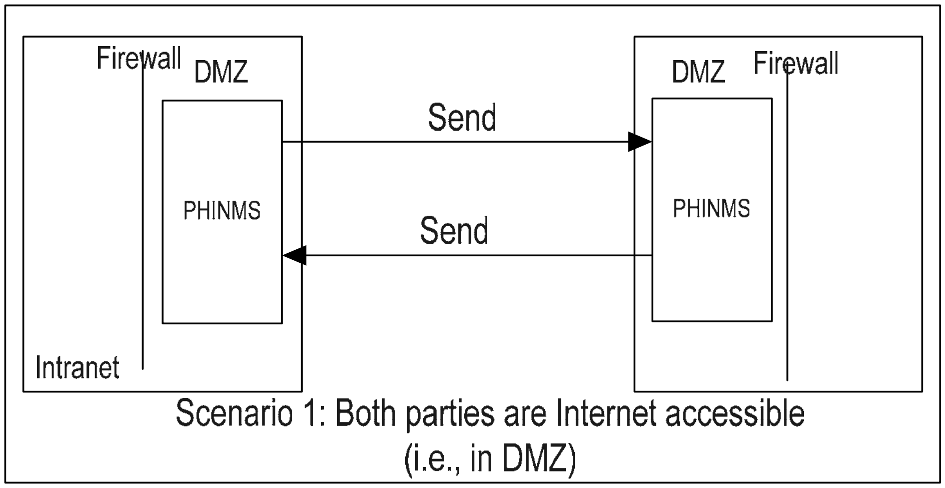
Firewalls

Though firewalls are necessary for the protection of resources within an enterprise, they complicate matters for a messaging system trying to send messages across enterprise boundaries. PHINMS uses two independent pieces of code, a client capable of sending messages and receiving real time (synchronous) responses, and a server receiver that can receive messages at any time. These two components may be used in three possible scenarios. These examples assume that the parties are in different organizations with separate firewalls.

1. Both parties are located outside their respective firewalls (i.e., in their DMZ)
2. One party is outside the firewall and the other is inside a firewall.
3. Both parties are inside their respective firewalls.

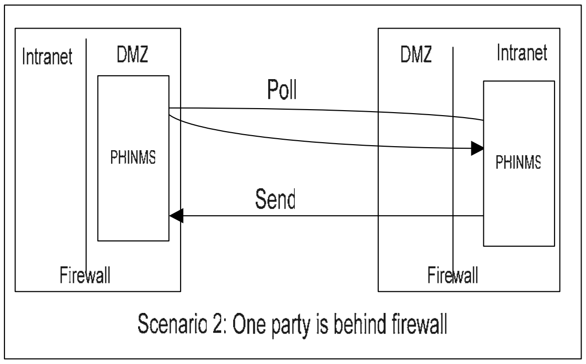
In the case where both parties are located outside their respective firewalls, messages may be sent and received at any time and acknowledgements send either synchronously or asynchronously. This requires that both parties have sending and receiving components installed.

Basically a poll is where a client sends a message to a server with some meta-data which maps to a piece of functionality that looks to see if the server has something for the client.
If so, the server can return the file as a response to the send. Because the client is not really receiving messages, the complexity of the software is reduced and therefore the platform requirements are reduced as well.

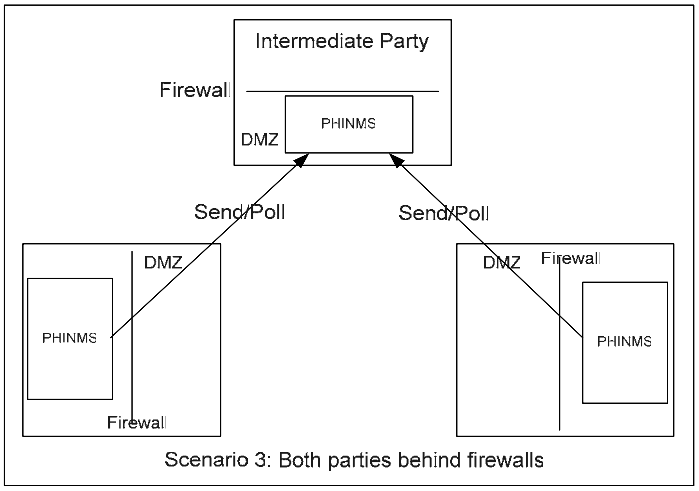

Firewall
DMZ
PHINMS
Intranet
Send
Send
DMZ
Firewall
PHINMS
Scenario 1: Both parties are Internet accessible (i.e., in DMZ)

For the situation where one party is behind a firewall and the other party has a server receiver located in the public Internet space, message sending options are slightly reduced. The client piece behind the firewall can send data much like a typical browser to a receiver and receive synchronous acknowledgements back.

Because it sits behind a firewall, the client cannot receive messages as firewalls typically block this type of "push" of information. What it can do is poll for messages.



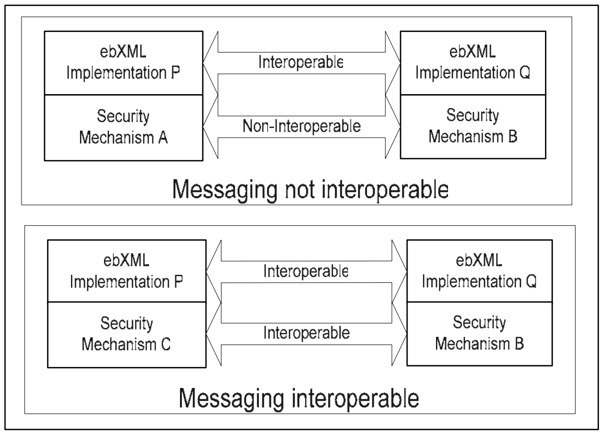
Typically the client can reside on a workstation capable of hosting a Java application.



In the case where both parties are behind firewalls, a third party server with Internet presence is required to broker the exchange. For example let's say party A is located behind a firewall in enterprise 1 and A wants to send a message to party B in enterprise 2, where B is also behind a firewall. Then A must send a message to an intermediary server on the Internet with a service action that states that the server should hold the message in a queue for B. Then when B polls the server, it will find the message from A in its queue and request it.

Authentication Interoperability

The ebXML messaging standard specifies the structure and semantics of message meta-data and addressing information, but for the most part, leaves the messaging security (identification and authentication) aspects to the implementers.



As shown in the above diagram, for interoperability, in addition to the message structure and semantics, the security mechanisms also need to interoperate. XML digital signatures can be used to support message non-repudiation (the strength of which is dependent upon legal elements that transcend the technology), but using them may not be sufficient for authentication, since digital signatures can be replayed[3].

When used, XML digital signatures should be combined with a handshaking protocol such as SSL, which mitigates the threat of replay attacks and provide freshness assurances. The alternative is to use SSL with client certificate based authentication. This provides per-link assurance of identity and authentication, as well as confidentiality. Since SSL is the most widely accepted standard, this is the recommended mode of authentication for PHINMS.

Acknowledgement

The authors would like to thank the Public Health Information Network Messaging System team: Michele Bowman, Thomas Brinks, Dongtao Jiang, Vaughn McMullin, Thomas Russell, and John Thomas.

Summary

The security design, implementation and deployment considerations of CDC's Public Health Information Network Messaging System (PHINMS) were discussed herein.

[3] A digital signature does not necessarily provide freshness evidence unless it is cryptographically bound to a freshness token, requiring time synchronization or nonce based handshakes. Without adequate freshness assurances use of DSIG in authentication may not be adequate for some applications.

References

[EbXML] Message Service Specification

Version 2.0, OASIS ebXML Messaging Services Technical Committee (http://www.ebxml.org/specs/ebMS2.pdf)

[LRN] The Laboratory Response Network Partners in Preparedness http://www.bt.cdc.gov/lrn/

[NEDSS] National Electronic Disease Surveillance System, The Surveillance and Monitoring Component for the Public Health Information Network. (www.cdc.gov/nedss/)

[NHSN] National Healthcare Safety Network (NHSN)(http://www.cdc.gov/ncidod/hip/NNIS/members/nhsn.htm)

[SOAP] SOAP Version 1.2 Part 0: Primer

(http://www.w3.org/TR/2003/REC-soap12-part0-20030624/)

[XMLENC] XML Encryption Requirements

(www.w3.org/TR/xml-encryption-req)

[XMLDSIG] XML-Signature Syntax Processing (www.w3g.org/TR/xmldsig-code