HARVARD UNIVERSITY
Information Technology

Developer Resources: PIN2

Contents

Introduction

The Harvard University PIN System provides centralized authentication services for online applications affiliated with the University. This gives users a single login that can be used with many different applications, relieving users of the need to remember multiple passwords nor system administrators of the need to maintain them.

The PIN System provides an authentication service only, not an authorization service. It tells you who the user is, but not what their affiliation with the University is, whether their affiliation is current or not, or whether they should have access to your system. You are responsible for determining this yourself within your application.

This document serves as a guide to implementing PIN2. Also offered, is the newer PIN3. For details on PIN 3 please see www.pin.harvard.edu/developer

From the application's point of view, there are three possible situations, which should be evaluated in this order:

1. The user's web client presents state information generated by your application.

   Your application validates the information provided. If invalid (e.g. the session has expired), you may forward the user to the PIN System for re-authentication. Otherwise, the user may continue with the session.

2. The user's web client presents a PIN authentication token.

   Your application verifies the token's signature, then validates the contents of the token. If invalid (e.g. the token has been tampered with, or has expired, or is signed by a PGP key other than the PIN System's), you may forward the user to the PIN System for re-authentication. Otherwise, you must determine what access the user should have and, if they have access, set up whatever state information you need to maintain the session.

3. The user's web client presents no authentication information.

   You may forward the user to the PIN System for authentication.

Registering an Application

Developers of online applications who wish to use the PIN2 System for authenticating their users must register their applications with Directory Services, the administrators of the PIN System, by email at: directory_services@harvard.edu

Registration provides a measure of security by protecting users from unknowingly providing information to "rogue" applications. Application administrators provide their application's URL, to which authenticated users are redirected. Without this information, authentication tokens cannot be generated.

Registration also allows the University to ensure that PIN authentication services are provided only to applications developed by faculties, departments, and officially sanctioned student organizations.

Information Required for Registration

To register an application, please download, print, and fill out this form:

http://reference.pin.harvard.edu/sites/reference.pin.harvard.edu/files/PIN%20Service%20Registration%20Form%20ver15_pub_0001.pdf

The form asks for the following information:

- The school/unit and department responsible for the application
- The name and email address of the person requesting the registration, the application's technical practice owner, business owner, project coordinator, and main developer
- A generic email address for communicating PIN-related announcements that will not change with turnover of personnel
- The name, description, and general user population of the application
- The service provider (internal/external)
- The authorization method (LDAP/Internal/Authorization Proxy/Other)
- An indication of the desired Single Sign-On option.
- Indicator of which login types are allowed for the application (Harvard University ID number, XID login, HMS eCommons login, or Post.Harvard login)
- For each instance of the application (development, test, production, etc.):
    - The URL from which users are directed to the PIN System
    - The URL to which authenticated users are directed by the PIN System
    - The location of the application server hosting the instance
    - The date by which the instance is needed

You may return this form to us by mail or by faxing it to us at (617) 496-5653. (We regret that due to the need for a written signature, we cannot offer an online version of this form at this time.)

Information Returned by Directory Services upon Registration

Once an application is registered, Directory Services will provide the application administrator with the following:

- The PGP public key of the PIN authentication server.

PIN tickets are digitally signed using PGP. Your application uses the authentication server's public key to verify that the ticket has not been forged.

- Internally used, unique application ids for each instance of the application.

  The application id is used to identify your application when redirecting your users to the PIN login page. You should also provide the application id when asking for any future modifications to your application registration info (e.g. redirect URLs, application login timeouts, supported login types, etc.)

- The URL of the page to which your application should redirect your users for authentication.

Technical information on how to use PIN authentication in your application

Directing Users to the PIN Login Page

Your application must be able to direct unauthenticated users to the PIN System. It is recommended that you provide an introductory page for such users. If you choose not to provide an introductory page, you may automatically direct the user to the PIN login page.

If you choose to provide an introductory page, we suggest that it at least contain the following:

- A general explanation of your application and who may use it.
- An HTML link containing the PIN login page URL, indicating that user's may login to the application by clicking the link.
- An HTML link to the "Request a New PIN" page, for those users who do not have a PIN or have forgotten it. The URL for the "Request a New PIN" page is:

  https://www.pin.harvard.edu/self-service/authentication.jsp

The PIN Login Page URL

Unauthenticated users must be directed to the PIN login page. The URL for the PIN login page is:

https://www.pin1.harvard.edu/pin/authenticate

Included on the URL as a parameter of the URL query string must be the application id that you obtained when the application was registered with the PIN System. The application id is used by the PIN System to identify which application is requesting authentication, determine which login types to allow the user to choose, and determine where the user should be directed after a successful authentication. The application id is a single string containing words in all capital letters, separated by underscores.

The parameter name that must be used when including the application id is:

__authen_application

An example of a complete PIN login page URL is:

https://www.pin1.harvard.edu/pin/authenticate? __authen_application=HARVARD_PORTAL

Getting Users to the PIN Login Page

You must provide a way for unauthenticated users to access the PIN login page URL. This may be done by providing a link on a page where the user can click the link and be directed to the PIN login page. It may also be done programmatically by sending an HTTP redirect message to the user's web browser with the PIN login page URL.

Whichever mechanism you provide for your users to get to the PIN login page, it must abide by the following rules:

- The URL to the PIN login page must be a complete URL as described above in the PIN Login Page URL section.
- The mechanism must employ an HTTP GET request when directing the user to the PIN login page URL. This can be accomplished with both a standard HTML link and an HTTP redirect message to a web browser.

The only method for getting to the PIN login page is through an HTTP GET request. The PIN System does not allow HTTP POST requests, or any other type of HTTP request, to be made against the PIN login page URL.

Saving State Information

The PIN System allows you to save state information from your application. Of course, you may do this on your own through the use of cookies. But, if you do not use cookies within your application, you may pass state information along to the PIN System where it will be returned to your application upon a successful authentication along with the PIN token.

In order to pass state information through the PIN System, simply include the information on the PIN login page URL as additional URL parameters. The PIN System retains any additional URL parameters throughout the authentication process, but ignores them for the purposes of authentication. So, additional URL parameters are returned as URL parameters using the exact same parameter names and values.

An example of a complete PIN login page URL with additional state information would be:

https://www.pin1.harvard.edu/pin/authenticate?
__authen_application=HARVARD_PORTAL&stateParameter1=stateValue1

As long as the URL for the PIN login page is complete as described in the PIN Login Page URL section, you may include as many additional state parameters as you'd like. The only

restriction is that of the actual physical limit to the number of characters allowed in a URL by web browsers.

Verifying PIN Authentication Tokens

If authentication is successful, an authentication token is generated by the PIN System. This token consists of a PGP-signed string containing the user's id, the application for which the user was authenticated, the IP address of the user's computer, and the time of authentication.

Upon successful authentication, an authentication token is generated by the PIN System. This token contains a set of data that includes the user id, the id of application for which the user was authenticated, the IP address of the user's computer, and the time of authentication. The token data formatted into a string that is signed using PGP and then the data and generated signature are transmitted to your application. Transmitting the authentication token to your application occurs through a web browser redirect.

The PIN System redirects the user back to your application with a query string appended to a URL that you provide. For example:

https://www.huid.harvard.edu/ssl/pin_verify.asp?
__authen_huid=10329779
&__authen_ip=128.103.217.27
&__authen_time=Wed+Jan+12+16%3A24%3A11+EST+2000
&__authen_application=HUIDTEST
&__authen_pgp_signature=iQA%2FAwUBOHzw%2Bx3DN6W2uztjEQITIQCeNxpKVJbkY
swX5ez3uruUpPsHjiYAoMl2%0D%0Ao2or%2BvWBV2GBsPEh00AWRbNw%0D%0A%3D
UEQ8
&__authen_pgp_version=5.0

Please note that the lines are wrapped to fit the width of this page. The actual URL would appear on a single line with no spaces.

Your application must be able to reconstruct the original token string from the URL parameters, verify the token string against the PGP signature provided, and then validate the individual pieces of data. A successful PGP signature verification assures you of the following:

- That the token originated from the PIN System (i.e. was not forged by a third party)
- That the token has not been tampered with in transit.

PIN Token Versions

With the introduction of different login types into the PIN System, it became necessary to modify the original PIN token format. In order to include the information necessary for the new login types and still remain compatible with existing customers, a second token format was developed. Both of the PIN token versions are valid and will continue to be valid in the future.

The new PIN token version was developed so that applications could identify the user's login type. It can be used by any application to verify/validate the authentication of any type of user. The format was designed so that it could be extensible in the future and still be backward compatible.

The steps for processing version 2 PIN tokens is different than the steps for processing version 1 PIN tokens, which means that any current customer would have to modify code if the application was to allow multiple login types. However, because of the design of the version 2 PIN tokens, there would be no need for code changes if more login types are integrated into the PIN System. And, if more data is included in the version 2 token in the future, applications that already process the version 2 token would not have to change their implementations unless processing the added data was necessary for operation of the application.

Processing the Token: Version 1

Version 1 of the PIN token is the original version. It consists of several URL parameters that each contains a single piece of data from the PIN token. That data is extracted, constructed into a single string for PGP verification, and then validated individually.

URL Parameters

The URL query string contains the following parameters for the version 1 PIN token:

Parameter Name	Description
__authen_huid	The internal of the person being authenticated. For Harvard University ID login types, this is the 8-digit HUID of the person being authenticated. For XID login types, this is the 8-character internal identify of the XID account used by the person being authenticated.
__authen_proxy_id	This is a legacy parameter and has been deprecated in the new PIN System. However, it must still be accounted for when verifying the PGP signature in order to ensure backwards compatibility with older verification code. It is also being kept as a placeholder in the event that the proxy id concept is reintroduced in the future.
__authen_ip	The IP address of the computer from which the user was authenticated.
__authen_time	The time the user was authenticated. The time is transferred in CTIME format, in Eastern Standard Time. For example: Wed Jan 12 16:24:11 EST 2000

Parameter Name	Description
__authen_application	The name of the application for which the user was authenticated.
__authen_pgp_signature	A PGP signature based on combination of the above values, as described below. The signature is generated using the PIN System's PGP private key.
__authen_pgp_version	The version of PGP which signed the token, set to a value of "5.0". The digital signatures generated by the PIN System are supported in all PGP versions since 5.0. Older versions of PGP (e.g. 2.6.2), which use different algorithms for digital signatures, are not compatible with the PIN System.

Constructing the Token String

To verify the PGP signature, you must first reconstruct the string that was originally signed by the PIN System. This string contains the data from the above parameters concatenated together with each piece of data delimited by the "pipe" symbol, "|". The data is concatenated in the following order:

1. __authen_application
2. __authen_huid
3. __authen_proxy_id
4. __authen_ip
5. __authen_time

Thus, in the sample query string from the Introduction section above, we have the following parameters values:

Parameter Name	Value
__authen_huid	10329779
__authen_proxy_id	{blank}
__authen_ip	128.103.217.27
__authen_time	Wed Jan 12 16:24:11 EST 2000
__authen_application	HUIDTEST
__authen_pgp_signature	iQA/AwUBOHzw+x3DN6W2uztjEQITIQCeNxpKVJbkY swX5ez3uruUpPsHjiYAoMl2 o2or+vWBV2GBsPEh00AWRbNw =UEQ8

Parameter Name	Value
__authen_pgp_version	5.0

Note: All of the parameter values are URL encoded so that they can be successfully passed on the URL. If the programming language that you are using does not automatically URL decode the parameter values when you retrieve them, then you will have to do so manually.

From these values, we construct the token string:

HUIDTEST|10329779||128.103.217.27|Wed Jan 12 16:24:11 EST 2000

At this point, you would verify the PGP signature value against the token string. For more detail, see the Signature Verification section.

Processing the Token: Version 2

Version 2 of the PIN token is the new version. It consists of two URL parameters; one containing the entire token string and one containing the PGP signature. The two parameter values are extracted and the signature is verified. Then, the data is parsed from the token string and the individual data elements are validated.

URL Parameters

The URL query string contains the following parameters for the version 2 PIN token:

Parameter Name	Description
__authen_parameters	The entire concatenated, pipe "|" delimited, PGP-signed token string.
__authen_pgp_signature	A PGP signature based on the above token string. The signature is generated using the PIN System's PGP private key.

Note: All of the parameter values are URL encoded so that they can be successfully passed on the URL. If the programming language that you are using does not automatically URL decode the parameter values when you retrieve them, then you will have to do so manually.

The values from the parameters should be extracted from the query string. At this point, you would verify the PGP signature value against the token string. For more detail, see the Signature Verification section.

Parsing the Token String

Processing the version 2 token is almost the total opposite of processing the version 1 token. In version 2, the __authen_parameters parameter contains all of the token data values including, now, the user's login type.

Taking the values from the previous PIN token sample in the Introduction section above, the __authen_parameters value would look like:

HUIDTEST|10329779||128.103.217.27|Wed Jan 12 16:24:11 EST 2000||P

Once the PGP signature is verified, the above token string must be parsed to obtain the individual data elements. The data elements are concatenated in the following order:

Application Id
: This element corresponds to the version 1 token's __authen_application parameter.

User Id
: This element corresponds to the version 1 token's __authen_huid parameter.

Proxy Id
: This element corresponds to the version 1 token's __authen_proxy_id parameter, which has been depractated. It is kept as a placeholder in the event that the proxy id concept is reintroduced in the future. The value will always be blank (an empty string "").

IP Address
: This element corresponds to the version 1 token's __authen_ip parameter.

Timestamp
: This element corresponds to the version 1 token's __authen_time parameter.

{blank}
: This element is a placeholder for a legacy data element that has been depracated in the new PIN System. The value will always be blank (an empty string "").

Login Type
: This element indicates the login type of the user's id. Current values are "PIN" for HUID, "XID" for XID, "HMS ECOMMONS" for HMS eCommons, and "POST.HARVARD" for Post.Harvard.

Thus, once we parse the sample token string above, we get the following data element values:

Data Element	Value
Application Id	HUIDTEST
User Id	10329779
Proxy Id	{blank}
IP Address	128.103.217.27
Timestamp	Wed Jan 12 16:24:11 EST 2000
{blank}	{blank}
Login Type	PIN

PGP Signature Verification

The PGP signature value that is transmitted with the PIN token is called a "detached" signature. Verifying a detached PGP signature requires two components -- the PGP public key from the PIN System and the signed data, which is, in this case, the PIN token string.

If the PGP tool that you are using does not require a full PGP-signed message, then you may be able to process the verification using the token string and the detached signature as-is. If the PGP tool that you are using requires a full PGP-signed message structure (most command line PGP tools will), then you will have to construct the full message from the PIN token string and detached signature. Refer to the documentation of the PGP tool that you are using to determine exactly how to verify a PGP signature.

The format of a full PGP-signed message is as follows:

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

{token string}
-----BEGIN PGP SIGNATURE-----
Version: 5.0

{pgp signature}
-----END PGP SIGNATURE-----

Thus the format of the message for the sample token is:

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

HUIDTEST|10329779||128.103.217.27|Wed Jan 12 16:24:11 EST 2000
-----BEGIN PGP SIGNATURE-----
Version: 5.0

iQA/AwUBOHzw+x3DN6W2uztjEQITIQCeNxpKVJbkYswX5ez3uruUpPsHjiYAoMl2
o2or+vWBV2GBsPEh00AWRbNw
=UEQ8
-----END PGP SIGNATURE-----

Note: If you are using a PGP tool that requires the full PGP-signed message and signature as shown above, then the message *must* match this format exactly. There *must* be one blank line between the "Hash:" indicator and the token string and *no* blank line between the token and the start of the signature block. Extra white space will cause the signature verification to fail.

You must decrypt this message using PGP version 5.0 or greater or GNU Privacy Guard version 1.0 or later. The signing algorithms of earlier versions of PGP (such as version 2.6.2) are not compatible with later versions and attempting to verify the signature with these earlier versions will fail.

Validating Token Components

After you have confirmed that the PGP signature is valid, you must perform the following additional checks:

- Confirm that the value of the __authen_application parameter matches the id assigned your application when it was registered with the PIN System
- Confirm that the value of the __authen_ip parameter matches the user's current IP address

It is recommended that you consider tokens invalid (expired) if the value of the __authen_time parameter is more than a few minutes old. The actual number of minutes is left up to your discretion.

The PIN System clock is synchronized with the National Institute of Standards and Technology's atomic clock twice per day, so is always accurate to within a second or two. Users are redirected to the application's site immediately upon successful authentication, but heavy network traffic might delay the redirection for a short period of time. Furthermore, system clocks available to applications may have varying accuracy. A window of a few minutes allows some flexibility in the time comparisons.

It is also recommended that accepted tokens be preserved. By saving previously accepted tokens for as long as you consider them valid and comparing them with newly presented ones, you can detect and block "replay attacks," where an attacker tries to gain access by reusing a token. Saving them indefinitely gives you additional system auditing capabilities.

Additional Resources

Below are some additional resources that may help in the understanding and processing of PIN authentication tokens.

Java, Perl and VB Code for Token Processing

There is existing code that is free to use by all Harvard and Harvard affiliated application developers who are registered to use the PIN System. For more information, and actual code library downloads, please go to the Downloads page of the Developer's Resources.

Helpful Links

The following are links to online resources you may find useful in developing your application.

- PGP Corporation

  The PGP Corporation purchased the rights to PGP from Network Associates, the former distributors of commercial PGP software. However, because the PGP application simply uses widely-supported encryption algorithms and has a well-defined file specification, other

developers may continue to support the data formats, and it is even possible to develop your own implementation of the token verification process.

- www.pgpi.org

  The international PGP distribution site. Contains links to downloads, reference documents and source code.

- GNU Privacy Guard

  A free cross-platform alternative to PGP. Fully compatible with the signatures generated by the Harvard University PIN System.

- Perl Security Modules

  Perl Crypt Modules

  There is a vast collection of free Perl modules that you can search at the Comprehensive Perl Archive Network, including are several that interface with PGP and GnuPG.

- Federal Information Processing Standard (FIPS) 180-1: Secure Hash Standard

  Federal Information Processing Standard (FIPS) 186-2: Digital Signature Standard

  Internet Engineering Task Force (IETF) RFC 2440

  These three documents define the Secure Hash Algorithm, the Digital Signature Algorithm, and the OpenPGP message format. They contain all the information you need to develop your own implementation of the PIN token verification process.

Other Considerations

Once the authentication token has been verified, your application is responsible for the following:

- Determining what privileges, if any, the user has within the application.
- Maintaining session state, e.g. through cookies or randomly generated session keys stored in hidden form fields.
- Preventing "replay attacks", i.e. reuse of a token after it has been validated.
- Protecting sensitive information from third-party interception with SSL encryption.
- Expiring page content immediately so that sensitive information is not cached.
- Ending users' sessions securely.

It is also recommended that you provide a link within your application that will allow users to logout of PIN. The PIN logout page is responsible for ending a user's PIN automatic login session, in

effect, logging a user out of the PIN System. Some applications automatically redirect a user to the PIN logout page when the user logs out of the application. Some applications provide seperate links for logging out of the application and logging out of the PIN System. The PIN logout URL is:

https://www.pin1.harvard.edu/pin/logout

Downloads

Harvard Framework Authentication Service (Java)

The Harvard University Framework group designed and constructed a general authentication service API for use in web applications. The authentication service is intended to provide application developers with a standard library for processing authentication assertions from an authentication authority (such as the Harvard PIN2 System) and then maintaining a user's login state throughout a session within the application.

Two specific implementations of the Harvard Framework Authentication service have been constructed for J2EE applications - one for applications that use the Harvard PIN2 System and one for applications that use the Harvard Authorization Proxy (AuthZProxy) in conjunction with the PIN System to receive data in an encrypted format.

We have packaged up a set of files containing all of the Java libraries required to use the authentication service and a "how-to" document in MS Word format. These files are distributed freely for use by Harvard and Harvard affiliated applications. If you have questions regarding integration with PIN and/or the AuthZProxy, please feel free to contact Directory Services. However, please note that we do **not** provide development support outside of the information in the packages below.

Download here:

- PIN (version 1 token)
- PIN (version 2 token) - Coming Soon
- AuthZProxy

Harvard Framework Authentication Service (Perl)

The Harvard University Framework group designed and constructed a general authentication service API for use in web applications. The authentication service is intended to provide application developers with a standard library for processing authentication assertions from an authentication authority (such as the Harvard PIN System) and then maintaining a user's login state throughout a session within the application.

A Perl implementation of the Harvard Framework Authentication service exists for applications that use the Harvard PIN System. We have packaged up a set of files containing all of the files required to use the authentication service and a "how-to" document in MS Word format. These files are distrubuted freely for use by Harvard and Harvard affiliated applications. If you have questions

regarding integration with PIN and/or the AuthZProxy, please feel free to contact Directory Services. However, please note that we do not provide development support outside of the information in the packages below.

At this time, there are no implementations in Perl for the AuthZProxy if you wish to receive data in an encrypted format.

Download here:

- PIN (version 1 token)

Microsoft Visual Basic DLLs

An ActiveX DLL interface was developed for use with Microsoft Windows NT/Internet Information Server to process version 1 PIN tokens. This DLL can be easily accessed by Active Server Pages scripts, or CGI or ISAPI applications, and provides all basic signature verification functions simply and efficiently, requiring you to specify only the token and the signature. There is also a sample ASP script demonstrating the use of the DLL. The DLL uses components of the PGP Software Development Kit and is subject to the terms of the PGP SDK license. Developers wishing to use this software may download the distribution, including an instructive readme.txt file, here:

- PIN DLLs (version 1 token)

DISCLAIMER
While this library has been tested, is available for download, and is in use in several production environments, Directory Services makes no guarantees as to its use with newer technologies or technologies other than the ones specified in the distribution. The code was developed using VB 6.0, which has since gone unsupported by Microsoft. Directory Services will provide information for setting up and using the libraries, but the code is delivered as-is and is no longer being supported by Directory Services in terms of upgrades and/or further development. Directory Services also offers no support if any work is required to port the code to the Microsoft .NET framework.

Public Keys

The PGP public keys for use in verifying the Harvard PIN System and Harvard Authorization Proxy (AuthZProxy) signature can be downloaded here.

Some browsers may simply display the key contents when you click on a link below, which can cause the keys to get corrupted if you attempt to copy and paste the text into a seperate file. In order to maintain proper formatting, it is suggested that you right-click on the links below and choose "Save As..." in order to download the key file.

- PIN Public Keys for the production, test and AuthZProxy production environments.