PWM Administrator's Guide

Updated for v1.5.2

Overview and Requirements

Welcome to PWM Administrator's Guide. PWM provides a feature rich, easy to manage, password self service web application for ldap directories.

Always check the PWM website for the latest version. The PWM website also has useful FAQs, support lists, and other helpful information as well as the most current version of this document.

http://code.google.com/p/pwm

This document describes the overall installation and setup of PWM. The PWM ConfigManager interface has detailed information about each of the many PWM configuration options.

PWM is distributed under the terms of the GNU General Public License (GPL). The full license text is available in the file "license.txt" included with the distribution.

Requirements

PWM will work on any platform that meets the following minimum requirements:

- Java J2SE v1.5 (5.0) or greater.
- Java Servlet Container v2.3 (Tomcat v4x or greater).
- 64 MB of Java heap memory. Heavily utilized sites may require larger heap sizes.
- 1 GB of disk space for the pwmDB for default configurations.

The full Java JDK is required to both run and compile PWM. The JRE is not sufficient.

PWM is developed and tested on the following platforms:

- SUSE Linux 11 and Windows 7
- Sun Java v1.5 and v1.6
- Tomcat v5.0.28 and v6.0.20
- Internet Explorer 6,7,8 and Firefox 3.6 and Google Chrome Browser, iPhone Mobile Safari

PWM does not require a web server. Some administrators may chose to integrate tomcat with Apache, IIS or other web server, but that is beyond the scope of this document. PWM is very lightweight and there will probably not be much performance difference with or without a web server, particularly if used with proxy server.

PWM has been known to work on JBoss and WebSphere. For non-tomcat containers such as these, PWM works best using an exploded archive directory. Alternatively, you can modify the web.xml to use a configuration file anywhere on the file system, and the configuration can in turn use a pwmDB directory that is anywhere on the file system.

Some operating systems (such as SLES) come equipped with a pre-installed version of tomcat and Java. Some administrators prefer to use the operating system instance of tomcat while others prefer a standalone tomcat instance for PWM. Keep in mind that the default tomcat settings found on tomcat downloaded directly from Apache's website are used to test and develop PWM.

Installation

Installation of PWM follows standard Java servlet guidelines.

Most of the difficulty administrators have installing PWM is usually not with PWM itself, but with Java, Tomcat, or an optional web server.

Server Setup

On your platform of choice, ensure that Java v5.0 or better is installed. You can do this by typing "java -version" at the operating systems command prompt. If the response is something other than the java version, visit http://java.sun.com and download the latest Java v1.5.x or better J2SE JDK. PWM works even better with Java 6 (also known as 1.6). Make sure you have the JDK installed, the JRE does not contain enough libraries for JSP compilation.

The next step is to install tomcat. Tomcat can be found at http://jakarta.apache.org. Instructions for installing Tomcat vary from platform to platform, follow the documentation on apache's site. In order to proceed you should be able to access tomcat's welcome page from a web browser.

Finally, find tomcat's /webapps directory, usually located directly under the main tomcat directory. Copy the included pwm.war file into the /webapps directory and restart tomcat. Newer versions of tomcat provide web-based managers that allow servlet deployment from a browser and without restarts.

When tomcat starts up, it will deploy the war file into /webapps/pwm directory. Next, visit the pwm application with a browser. Assuming your browser and tomcat are on the same machine and your using the default tomcat port numbers, the url will be:

```
http://locahost:8080/pwm
```

Upon the initial PWM installation, a web-based configuration editor is available at:

```
http://locahost:8080/pwm/config/ConfigManager
```

Once a configuration is saved, PWM will operate in "Configuration Mode", which allows continued configuration changes to be made while you test PWM functionality. When all configuration changes are completed, you can Finalize the PWM configuration, which will lock the configuration and prevent any further changes via the web based ConfigManager. If you wish to make further changes to the configuration once the config is Finalized, you can follow the instructions on the ConfigManager screen. These steps require access to the server file system.

Beyond configuring PWM, you may also need to make changes to your LDAP directory schema and rights/permissions. Information about schema and rights modifications for particular directories can be found below, or by visiting the pwm-general discussion group.

Making a new WAR file

For a variety of reasons, it may be desirable to deploy a customized WAR file, you can use the included ant script to repackage a new WAR file after you have modified any of the PWM files.

Even if you can modify the servlet files after they are deployed, it's still a good idea to modify the original source and build your own pwm.war. That way, it will be available as a backup.

With this process, you can have a pwm.war file that is completely customized for your environment. You do not need ant on your server, you can follow these steps on your workstation, and then deploy the resulting war file on your server. It is even okay if your workstation and server are different operating systems.

To build a new pwm.war file, follow the following steps:

1. Make sure you have Java v1.5 SDK or better installed.
2. Set an environment variable for JDK_HOME to the JDK install directory.
3. Download and install Apache Ant
4. Make sure ant is in your path. Running "ant -version" should produce results.
5. Download PWM and unzip the pwm.zip directory
6. Make any changes to PWM desired (jsp edits, configuration changes, etc)
7. At the root of the pwm directory (where the build.xml file is located), run the following commands:
   ```
   ant clean
   ant makeWAR
   ```
8. Deploy the resulting pwm.war file. On tomcat this means deleting the webapps/pwm directory (if any) placing pwm.war file in the webaps directory, and restarting tomcat. (Newer versions of tomcat allow deploying war files through a web interface without restarting tomcat.

Novell eDirectory Integration

PWM was robust support for Novell eDirectory. The following features are supported:

- Read Universal Password policies and traditional password settings
- Correctly handle intruder lockout scenarios
- Read Universal Password challenge set policies, including localized policies.
- Write forgotten password responses to NMAS for compatibility with Novell forgotten password clients

PWM is only able to utilize existing stored NMAS responses for forgotten passwords when Novell UserApp (RBPM) is available. PWM utilizes web services available in IDM UserApp to validate user responses. This feature is optional. If UserApp is not available, PWM will use it’s own saved challenge/responses for user response validation.

eDirectory Schema

PWM uses eDirectory attributes to store data about users, including last password change time, last time PWM sent email notices about password expiration, and secret question/answer.

PWM includes a "edirectory-schema.ldif" file in the schema directory that has the standard PWM schema extensions. The schema included uses an auxiliary class that is added to users as they use PWM, so the auxiliary class and attributes are removable from eDirectory in the future.

Using the ICE command line the schema file can be imported with a command something like this:

```
ice -SLDIF -f edirectory-schema.ldif -DLDAP -s 192.168.75.132 -d
cn=admin,ou=ou,o=o -w password
```

The ldif file can be imported using the ConsoleOne wizard, iManager, the ICE command line, and standard "ldapmodify" tools.

If you do not wish to use the standard PWM schema, all the attributes used by PWM can be changed in the PWM configuration to attributes that are already available in the directory.

eDirectory Rights

PWM requires permission to perform operations in eDirectory.
PWM uses two different eDirectory logins, one is a generic proxy user that is used for certain operations, preAuthenticaton operations.
Once the user is authenticated most operations will be performed with the user's connection and permissions.

The ldif file eDirectoryRights.ldif is included that offers a sample basic configuration of a proxy user and also sets ACL for users that are required by PWM.

For a default configuration, the following rights are required for the proxy user to the user container(s):

- Browse rights to [Entry Rights]
- Read and Compare rights to pwmResponseSet and CN (or otherwise configured naming attribute)
- Read and Compare and Write rights to objectClass, passwordManagement, pwmEventLog and pwmLastPwdUpdate
- Read and Compare rights to any attribute used by ActivateUser servlet

For a default configuration, the following rights are required for each user to their own user entry:

- Browse rights to [Entry Rights]
- Read and Compare and Write rights to pwmResponseSet
- Read and Compare and Write rights to any attributes used in the UpdateAttributes servlet

To assign rights to each user, it is best to use the [this] security entry. Assigning rights to a parent level container to modify pwmResponseSet will allow any user in the container to modify the value of this attribute for any other user in the container, and thus allow a password reset by any user. See the eDirectoryRights.ldif file for an example of the [this] security entry.

Optionally, PWM should also have rights to read the password. This is configured as part of the eDirectory password policies. Normally, when a user uses PWM's Forgotten Password recovery feature, pwm will set the user's password to a randomly generated value during the recovery process. It does this so that when the user actually does type set a new password, the

PWM can authenticate as the user using the ramdom password, and then change the password using the user's credentials.

This process allows the directory to apply normal change password effects such as correctly setting the password expiration. However, if PWM is able to read the password of the user, it will not set an intermediary temporary password on the user.

PWM provides extensive logging to help troubleshoot installation and configuration issues. For best results during installation, set the log levels to "trace" in the log4jconfig.xml file, and monitor the output. See Logging

eDirectory Interaction

With the default configuration, PWM performs all operations against eDirectory using generic LDAP calls unless NMAS support is enabled.

Enabling NMAS allows for better error reporting and integration with eDirectory.

Many operations are preformed using the proxy user specified in the PWM Configuration. However, if the Always Use Proxy is set to true, then PWM will not bind as the user. Authentications will be handled using ldap compare.

If the option to store NMAS responses is enabled, then whenever a user saves their responses using PWM, they will also be stored in NMAS. This allows for Novell forgotten password clients to use the same responses. However, PWM itself can not directly use these responses for forgotten password.

OpenLDAP Integration

There are a few modifications that may be needed to the OpenLDAP configuration file, /etc/ldap/slapd.conf. Note that these modifications here are suggested as a template and may need to be customized to your own requirements.

For example, if pwm is configured to enable New User Registration and use the cn attribute to test for object name uniqueness, then the following configuration edit would allow PWM to perform the ldap equality check from PWM.

```
# Index cn to allow equality checks from the pwm webapp. This is
# used by the new user registration module to check whether a given
# username (cn) already exists in the LDAP directory.
index           cn eq
```

Access control rules

Configuring access control rules properly for pwm can be challenging. The following ruleset can be used as a reference. It has been tested to verify that users (or the PWM Admin) does not have any more privileges to the directory than absolutely required, but this configuration should be examined closely to ensure it does not allow any excess privileges in your environment.

```
# The userPassword by default can be changed
# by the entry owning it if they are authenticated.
# Others should not be able to see it, except the
# admin entry below
#
# NOTE: this is mostly standard OpenLDAP configuration.
# The pwmadmin line can probably be omitted.
access to attrs=userPassword,shadowLastChange
        by dn="cn=admin,dc=domain,dc=com" write
        by dn="cn=pwmadmin,dc=domain,dc=com" write
        by anonymous auth
        by self write
        by * none

# Grant access to subtree "ou=Accounts, dc=domain, dc=com"
# which contains the accounts created by pwm. Change
# this path to match your LDAP directory.
#
# Note that here we have two "classes" of pwm users. The
# dn="cn=pwmadmin,dc=domain,dc=com" user is used during
# the initial phases of new user registration to create
# the user into the directory. This is why the pwmadmin
# account needs to have write access to this subtree.
#
# Right after the user is created, pwm binds to OpenLDAP
# as the newly created user, so "by anonymous auth" is
# required. The user also needs to be able modify it's own
# data, so "by self write" is also needed. If it's missing,
# the user's password and/or personal information cannot be
# created/edited. This also prevents the new user
# registration from working properly.
access to dn.subtree="ou=Accounts,dc=domain,dc=com"
        by dn="cn=admin,dc=domain,dc=com" write
        by dn="cn=pwmadmin,dc=domain,dc=com" write
        by anonymous auth
        by self write
        by * none

# NOTE: this is standard OpenLDAP stuff
access to dn.base="" by * read

# The admin dn has full write access, pwmadmin has full read access.
# The pwmadmin entry may not be required. Feel free to experiment.
access to *
        by dn="cn=admin,dc=domain,dc=com" write
        by dn="cn=pwmadmin,dc=domain,dc=com" read
        by * none
```

Schema extensions

Pwm contains a few schemas extensions which are distributed as LDIF files, however some of the settings are Novell eDirectory-specific, so these templates can instead be used on for OpenLDAP. The following are changes made to the */etc/ldap/schema* file.

```
# /etc/ldap/schema/pwm.schema
#
# We try to define OID's "correctly" as outlined here:
#
# http://www.openldap.org/doc/admin23/schema.html
#
# 1.3.6.1.4.1   base OID
# 591242        organization idenfifier
# 1             if an objectclass
# 2             if an attribute
# yyyy.mm.dd    date of creation
# n             extra identifier
#
attributetype ( 1.3.6.1.4.1.591242.2.2010.04.16.1
        NAME 'pwmEventLog'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.40 )

attributetype ( 1.3.6.1.4.1.591242.2.2010.04.16.2
        NAME 'pwmResponseSet'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.40 )

attributetype ( 1.3.6.1.4.1.591242.2.2010.04.16.3
        NAME 'pwmLastPwdUpdate'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 )

objectclass ( 1.3.6.1.4.1.591242.1.2010.04.16.1
        NAME 'pwmUser'
        AUXILIARY
        MAY ( pwmLastPwdUpdate $ pwmEventLog $ pwmResponseSet )
```

This allows pwm to track pwm-specific attributes for each user.

Active Directory Integration

PWM has support for standard change password functionality against active directory. Forgotten Password support also functions correctly with PWM, however the exact steps to create the schema and Active Directory rights have not yet been documented.

Web Integration and Page Flow

PWM has been designed and tested to work well with portals and web access gateways.

PWM uses a very simple page flow model, with limited opportunity for configuration or changes. However, the settings that are available combined with some creative HTTP redirecting can allow for very customized scenarios.

It is helpful to not think of PWM as a standard web application with meaningful user navigation; instead the general intent is to place the user on a PWM page, complete a function, then redirect the user elsewhere. This keeps with the notion that password functions are typically not regarded as something the user desires to do, but rather an interruption or “security-wall” around the desired content.

PWM uses two configurable URLs, the **logoutURL** and **forwardURL**. These URLs are configured as part of PWM's general configuration. However, they can be overridden for any particular session by including the HTTP parameters *forwardURL* or *continueURL* on any request during the session.

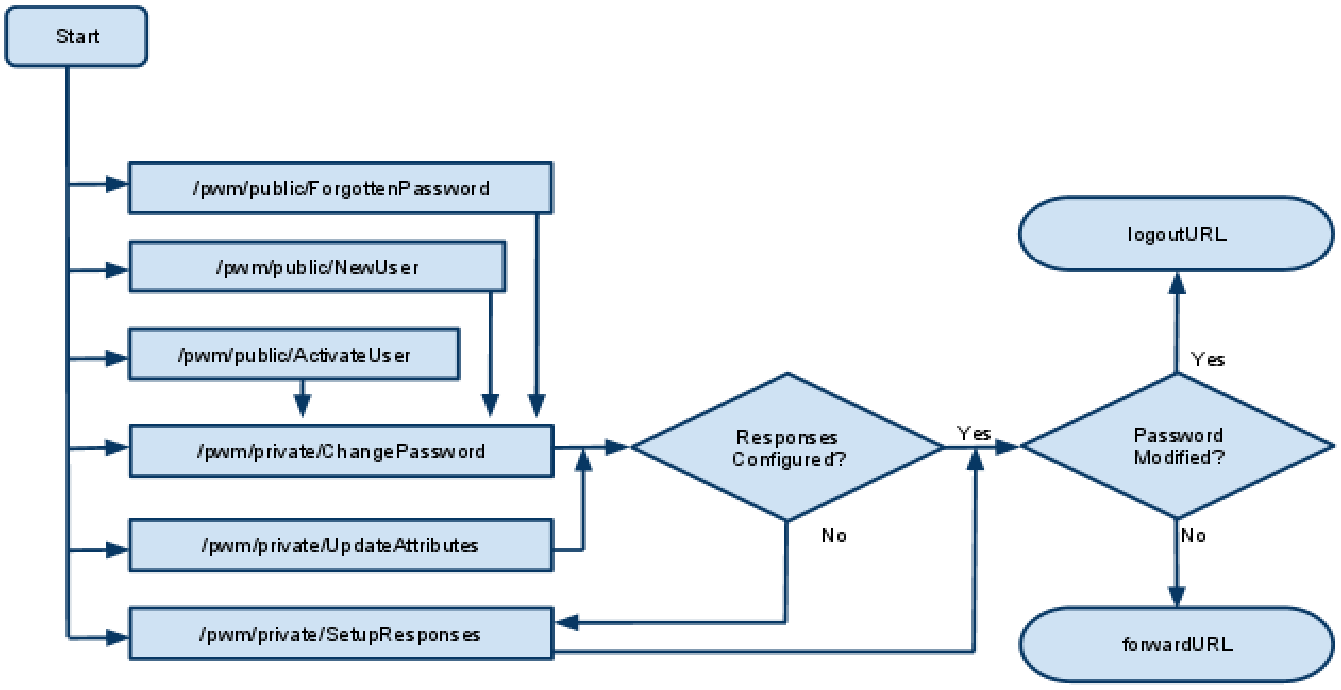
After completing a function, the user will be redirected to the *forwardURL*, except if the password has been modified and the Logout After Password Change setting is set to true. In that case, the user will be redirected to the *logoutURL* instead.

There are two exceptions where a user is not immediately redirected to the *forwardURL*.

The first is when the Check Expiration During Authentication setting is set to true and the user's password has been determined to be expired. If this is the case, then the user is redirected to the change password screen (or possibly the password expiration warning page if the expiration is within the Expire Warn Time window. After the password change is completed, the user is then redirected back to the *forwardURL*/*logoutURL* except if:

The second exception is when Force Setup of Challenge Responses setting is set to true, the user matches Challenge Response Queiry Match and the user does not have valid pwm responses configured. In this case, the user is redirected to the setup responses module. Once compete, the user is then redirected back to the *forwardURL*/*logoutURL*.

Standard PWM Page Flow



Access Gateways

PWM Supports basic authentication. If an http "Authorization" header is present, PWM will use the credentials in the header to authenticte the user. It is best practice to use the entire user DN as the username, but simple usernames will also be accepted and PWM will attempt to search the directory for the correct user.

Some parts of PWM need to be publicly accessible, such as forgotten password modules and new user registration. To support this, configure the following urls as public or restricted by your proxy or gateway configuration

Assuming PWM is setup so the user enters the following url to access PWM:

```
http://password.example.com/pwm
```

Add the following protected URLs:

URL	**Mode**
password.example.com/*	Public
password.example.com/pwm/private/*	Restricted
password.example.com/pwm/admin/*	Restricted
password.example.com/pwm/config/*	Restricted

If your access gateway supports it, you should configure it to redirect to PWM if the password is expired.

For expired password URL use:

```
http://password.example.com/pwm/private/ChangePassword?
passwordExpired=true
```

Optionally you can modify your login page to add links to the public NewUser page or Forgotten Password page.

Request Parameters

A variety of commands to PWM can be specified as parameters on URLs. Parameters are case sensitive. An example follows:

```
http://password.example.com/pwm/private/ChangePassword?
passwordExpired=true&forwardURL=http://www.example.com
```

Parameter	Effect
passwordExpired=true	Setting this parameter will make PWM override the state of the user's password expiration. This can be used to make PWM behave as though the user's password is expired, even if it is not.
forwardURL	Specifies a URL for PWM to send the user to after they are complete with PWM. If the user password is changed however, the user is sent to the logoutURL. This setting overrides the setting in the PWM Configuration file for the life of the session. Value must be URLEncoded
logoutURL	Specifies a URL to use as the PWM logout url. Generally, PWM only sends users to this url if the password has changed at some point in the user session. This setting overrides the setting in the PWM Configuration file for the life of the session. Value must be URLEncoded

Command Servlet

The CommandServlet allows you to redirect a user to PWM and have it perform some specific command. Typically, you would use the CommandServlet functions during a user's login sequence to a portal or some other landing point.

Ideally, these functions work best when used with iChain or some other device that will auto-authenticate the user. Otherwise, the user will have to authenticate to PWM during every login.

CommandServlet calls can be combined with any of the request parameters described above, such as the "forwardURL" parameter.

For example, the user login redirect sequence may proceed as follows:

http://portal.example.com	Initial request from browser
http://portal.example.com/Login	access gateway redirects to login page
http://portal.example.com/	access gateway redirects back to portal root
http://portal.example.com/index.html	web server redirects to index.html
http://password.example.com/pwm/private/CommandServlet?processAction=checkAll&forwardURL=http%3A%2F%2Fportal.example.com%2Fportalpage.html	index.html has meta redirect to PWM checkAll CommandServlet with a URLEncoded forwardURL value.
http://portal.example.com/portal/main.html	PWM redirects back to the actual portal URL

# 'http://portal.example.com/portal/main.html' PWM redirects back to the actual portal URL

The index.html described above would have the following content:

```
<html>
  <head>

<meta http-equiv="REFRESH" content="0; URL=http://
password.example.com/pwm/private/CommandServlet?

processAction=checkAll&forwardURL=http%3A%2F%2Fportal.example.com%2Fpo
rtalpage.html"/>
  </head>
  <body>
   <p>If your browser doesn't automatically load, click
    <a href="http://password.example.com/pwm/private/CommandServlet?
processAction=checkAll&
       forwardURL=http://portal.example.com/portal/main.html">here</
a>.
   </p>
  </body>
</html>
```

Commands

checkExpire *http://password.example.com/pwm/private/CommandServlet?processAction=checkExpire*

Checks the user's password expiration. If the expiration date is within the configured threshold, the user will be required to change password.

checkResponses *http://password.example.com/pwm/private/CommandServlet?processAction=checkResponses*

Checks the user's challenge responses. If no responses are configured, the user will be set them up.

checkAttributes *http://password.example.com/pwm/private/CommandServlet?processAction=checkAttributes*

Checks the user's attributes. If the user's attributes do not meet the configured requirements, the user will be required to set their attributes.

checkAll *http://password.example.com/pwm/private/CommandServlet?processAction=checkAll*

Calls checkExpire, checkResponses and checkAttributes consecutively.

Internationalization

PWM is fully internationalized, and comes with several localized languages. Only pages and configuration options that affect end users are internationalized. Most administrator screens are not localized or internationalized.

PWM uses the Java PropertyResourceBundle strategy for managing localized text. The locale of the user is determined by examinging the first HTTP request sent by the browser. Any locale information in ldap is not evaluated. Thus, PWM responds in the configured locale of the browser.

Support for internationalization falls into three categories: Display, Configuration, and Challenge/Response.

Display Pages

Localized display values are managed by *.properties* files. The *.properties* files are located in the PWM src directory along with the Java class files that use them. Localized fields referenced using the <pwm:Display> jsp tag are part of the password.pwm.config.Display class

Locales are indicated by copies of the file appended with the locale code. For example, the French version of *Display.properties* is *Display_fr.properties*. The locale can be further specified by a sub-locale, such as for Canadian French, the *Display.properties* file would be

Display_fr_ca.properties. If a file or particular key in the file is missing, the default version will be used.

Once deployed, these language files reside in:

```
/WEB-INF/classes/password/pwm
```

It is probably better to change these files in the source directory and rebuild a new WAR.

As the content of these files are generally not site-specific, administrators who modify or add to the localization bundles are encouraged to share these updates with the PWM developers for future inclusion in a shipping PWM version.

Configuration

PWM allows several configuration values in the PWM Configuration to be localized. Generally, any configuration option which includes text visible to the end user is eligible for localization.

Challenge Questions

PWM takes special consideration for Challenge/Response settings. Different challenges can be configured for each potential locale. Not only can the challenges be different, but the number of challenges, and number of random challenges can be different as well.

PWM will determine which set of challenges to use based on the locale of the browser during the SetupResponses process. Responses stored in ldap are tied to the locale of the challenge sets. Forever after, PWM will display and check the challenges of the user based on the locale they were entered in. For example, if a user sets up their responses in French, and then logs into pwm to recover their passwords from an English web browser, the challenges will display in French, and the user must enter the French responses.

If the user then sets up responses in a different language, the new language responses will overwrite the old responses.

Wordlists

PWM is capable of checking user entered passwords and dis-allowing those found in a predefined password dictionary wordlist. The dictionary in use is configured by settng the "password.WordlistFile" setting in pwmSetting.properties.

The wordlist is a ZIP file containing one or more plain text files with one word per line. To facilitate the speedy checking of the dictionary during password changes, PWM compiles the wordlist into a local embedded database (pwmDB). By default, PWM uses the BerkelyDB embedded database, and stores its files in the META-INF/pwm-db directory. With logging set to trace, PWM will output the status of the wordlist compile to the log. The compile process only needs to run once unless the wordlist ZIP file is modified.

The default distribution of PWM contains a wordlist that includes about 8 million words of common passwords found in many systems. Larger worldlists containing tens of millions of words with a variety of languages are available on the PWM website. PWM has been tested with wordlists over 50 million words, however the initial wordlist compile time can be excessive.

Seedlists are used by pwm to generate random passwords for users. The seedlists are used as a basis for a new random password, but are sufficiently modified to gaurentee randomness and also to meet the configured policy for the user. Seedlists are specified with the "password.SeedlistFile" setting.

Global Password History

A seperate wordlist dictionary is used to provide a global password wordlist history. This wordlist is not pre-populated by any file, instead it is populated with a user's old password during any password change.

This has the effect of preventing any two users from using the same password over time. For security reasons, only the user's old password is stored. Administrators should evalute this feature closely before enabling it. This setting is controlled with the Shared Password History Age setting in the PWM Configuration. PWM will periodically purge the global password history of any passwords older than this age. A value of several months or years is appropriate. The unit for the setting is in seconds. A value of "0" will disable the feature.

PWM Database (pwmDB)

To store wordlists, event logs and statistics, pwmDB uses an on-disk embedded database. Several different database types are available. The default is the BerekelyDB embedded database. The default settings are intended to be self-sufficient, and should not require any administrator interaction or maintenance. The pwmDB location and other settings are configured in the advanced section of the PWM Configuration.

Policies

PWM uses a matrix of configurations for determining the correct password and challenge/ response policies for determining a policy for a given set. PWM gives detailed information about discovered policies during authentication time at the TRACE log level.

Password Policy

Each password policy setting is available in the PWM Configuration. These password policies represent "minimum" policies that will be applied to the user. If the setting the directory type is Novell eDirectory and the Configuration setting Read eDirectory Password Policy is set to true, then PWM will attempt to locate a Universal Password policy configured for the user. If one is found then the policy is merged with the settings in the policies set in the PWM Configuration. The most restrictive of any two settings are used.

When using eDirectory, PWM will attempt to read the legacy password policy settings directly from the user object in the case where the user does not have a Universal Password policy assigned.

For example, if the PWM Configuration contains a setting of Password Minimum Length set to 5 and the Universal Password policy has a setting of 4, then the minimum password length for the user will be 5.

Challenge Policy

For challenge questions, pwm can read both the PWM Configuration, as well as the configuration stored in eDirectory Universal Password policies. The behavior is disable in the eDirectory configuration settings.

Challenge policies can be localized for the the users language.

If localized policies are discovered in the ldap directory, PWM will honor them.

PWM can also be configured to require that all random password questions be required to be entered at setup time. In this case, the questions are presented randomly to the user when trying to reset a forgotten password. This is the standard behavior with eDirectory challenges.

Alternatively, PWM can allow the user to provide responses for only the minimum number of required random responses at setup time. Then, only those responses will be required when resetting a forgotten password. This behavior reflects more common forgotten password scenarios encountered on the Internet, but is less secure.

Logging

PWM uses Apache Log4j. Log4j is a common logging java engine for Java. Log4j allows logging to a variety of destinations such as files, syslog, nt event log, databases and others.

Logging levels available (in order of severity):

1. trace
2. debug
3. info
4. error
5. warn
6. fatal

For normal operations, "info" level should be sufficient. "trace" level is recommended during initial configuration. Logging is controlled by editing the PWM Configuration. Alternatively, PWM can be configured to register a log4j xml configuration file. A default sample configuration *lof4jconfig.xml* is included.

The default logging configuration logs info level to stdout. When used with tomcat, this will log to tomcat's logs/catalina.out file.

PWM also stores a temporary log of recent events in the pwmDB. This log is accessible in the administrator screens and is useful for checking recent pwm activity.

Captchas

PWM has integrated support for Captcha protection. The captcha makes it more difficult for a hacker to launch an automated attack against PWM. PWM uses the online reCaptcha service for captcha generation and validation. You will need to configure a reCaptcha account to use the service. reCaptcha accounts are free.

The online service approach insures that the captcha technology is continuously improved to make it difficult to be compromised. The reCaptcha account parameters are configured in the PWM Configuration.

Java Security Manager

Pwm can be used with Java Security Manager. For those not familiar with it, it's like SELinux for Java. It ensures that the application (pwm) does not do anything it's not supposed to, even if it has security problems. As Java Security Manager is a pain to setup, the usual advise is to turn it off. Fortunately there are full integration instructions available here.