

Chapter 28



External Authentication

28.1 Overview

You can override the standard NetVigil authentication methods by creating your own plugin Java classes or scripts. This allows the use of custom authentication databases or other site-specific authentication methods to control access to NetVigil. Note that although it is possible to use an external authentication source, the authorization information (permissions, limits, etc.) are still stored in NetVigil's provisioning database. So it will be necessary to create the login ID on NetVigil database even though that login is authenticated from an external database.

Additionally, NetVigil provides the facility for integrating with a Web portal and using the Web portal's authentication mechanism.

NOTE: If you change the authentication mechanism, the changes are applied only to users created after the change was made. To use the new authentication method for users created before the authentication mechanism changed, you must change their passwords using either the web application (see the chapter “Managing Actions and Notifications” in the Web User Guide) or the BVE TCP server (see “User.update” on page 305). Otherwise, older users will continue to be authenticated by the old mechanism (e.g., NetVigil's internal password database).

28.2 Authentication Plugin Java Class

If you want to create a java class for authentication, your plugin class must implement the `NetvigilPluginAuthentication` interface. See the javadoc for `NetvigilPluginAuthentication` for more information.

Your implementation of the Java class

`NetvigilPluginAuthentication.getAuthenticationString` should take the user login name and password, and create an authentication string, such as an encrypted password, from this information. The `Properties` argument to `getAuthenticationString` is reserved for future use. When a user logs in, your implementation of `NetVigilPluginAuthentication.authenticate` will be given the user login name, the authentication string for that user that was created by `getAuthenticationString`, and the password the user gave when he or she tried to log in. As with `getAuthenticationString`, the `Properties` parameter to `authenticate` is reserved for future use. Your version of `authenticate` should use this information to determine whether or not the user should be allowed to log in. If the user should be allowed to login, your version of `authenticate` should return `true`.

As an example, for a Java class which does authentication using `rot13` to encrypt a given password, you should choose a unique string to identify the authentication method for your class. This string will be stored in the database to indicate the type of authentication method to use with a given authentication string. The strings `clear`, `des`, and `script` are reserved, but you can use any other unique string to identify your plugin authentication method.

Finally, you must use the unique string along with the name of your plugin class in the `authentication` section of `NETVIGIL_HOME/etc/netvigil.xml` to identify your plugin class as the class to use for authentication. For example, if you wanted to use the `rot13` class mentioned above for authentication, you could choose the string “`rot13`” as the identifier, and modify the `authentication` section in `netvigil.xml` so that it looks like this:

```
<authentication
  method="rot13"
  class="Rot13Authentication"
  execute=""
  parameters=""
/>
```

Leave the `execute` and `parameters` attributes in the `authentication` section empty. They're reserved for plugin authentication using scripts (described in Section 28.3, “Authentication Plugin Script” on page 353).

Once you're done writing your class, create a `.jar` file for it and any other required classes and place them in `NETVIGIL_DIR/plugin/auth` directory. Your plugin class, and any third party `.jars` you've included, must work with the Sun 1.4 JVM.

28.3 Authentication Plugin Script

You can also specify a script, program or batch file to use for authentication. NetVigil will run this program and pass it the user's login name and password as arguments. Following the convention of using a zero return code for successful program execution, your script must return a zero value to indicate that authentication was successful. You can specify the format that the arguments are passed to your program.

Here's an example Perl script (`auth.pl`) that will only let a user named "jane" log in, and only if she gives the password "secret".

Sample login authentication script

```
.....
#!/usr/bin/perl
if($#ARGV != 1) {
    print STDERR "not enough arguments!\n";
    # exit with a non-zero
    exit 2;
}

# get the username and password from the arguments
#
# we've set up our parameter string so that username
# is the first argument, and password is the second
#

$username = $ARGV[0];
$password = $ARGV[1];

if($username eq "jane" && $password eq "secret") {
    # return 0 so that jane can log in to NetVigil
    exit 0;
} else {
    # return a non-zero failure code, since the username
    # and/or password was wrong.
    exit 1;
}
.....
```

Once you're done with your script, place it in the NetVigil plugin authentication directory (`NETVIGIL_DIR/plugin/auth`). To instruct NetVigil to use your script for authentication, you'll need to modify `NETVIGIL_DIR/etc/netvigil.xml`. Update the authentication element, which initially may look like this:

```
<authentication
  method="des"
  class=""
  execute=""
  parameters=""
/>
```

Change this so that the method attribute is `script`. This will tell NetVigil that you want to do authentication with a script. Leave the class attribute empty, since that's only used for plugin authentication using a Java class (described in Section 28.2, “Authentication Plugin Java Class” on page 351). Place the name of your script in the `execute` attribute. Use the `parameters` attribute to specify the order that the user name and password should be passed to your script, along with any other flags you want passed. You can use the special variables `${username}` and `${password}` as placeholders for the username and password respectively. For example, you may want your script to take GNU-style long parameters, so you could set the `parameters` attribute to something like this:

```
--username=${username} --password=${password}
```

Since our example script doesn't use any flags for the username and password, we'll use `${username} ${password}` for the parameters. The authentication section of `netvigil.xml` would look like this after we're done:

```
<authentication
  method="script"
  class="class=""
  execute="auth.pl"
  parameters="${username} ${password}"
/>
```

Note that any existing NetVigil users will still continue to be authenticated using the older authentication method (since the authentication method is stored with each user entry in the database). To switch them to the new scheme, simply change their password once.

This allows you to keep the password for “superuser” tied to the local authentication scheme and not dependent on an external resource or database.

WARNING On Linux or Solaris, any parameters passed to the plugin script you specify may be viewed by anyone on your system with the `ps` command during the time it takes the script to execute.

28.4 Web Portal Authentication

You can bypass the initial login page in NetVigil by directly encoding the username and password information in the URL and encrypting this information using a shared key. This mechanism allows a user to access NetVigil via some other portal where he/she has already been authenticated.

1. Edit `NETVIGIL_HOME/tomcat/webapps/ROOT/WEB-INF/web.xml` and change the shared key in `<param-name>externalLoginKey</param-name>`
2. Copy `NETVIGIL_HOME/utils/externalWebLogin.cgi` to your web portal.
3. Edit this script and set the shared key, as well as the mechanism to get the department, username, password (can be changed to extract from the HTTP environment depending on your setup).
4. Set `maxPages` to 1 to limit the user to only view the one page that the URL connects to, else leave as -1 for full access.

This allows displaying just one page (e.g making one report publicly available) and not allowing a full login.

28.5 Architectural Description

If you've specified a plugin java class to use for authentication, when a user password is created or changed, your implementation of the `NetvigilPluginAuthentication.getAuthenticationString` method will be called. The authentication string this method returns will be stored in the provisioning database, along with the unique string you picked to identify the authentication method. NetVigil will use the unique string as a key to find and load your plugin authentication class. When the user tries to login to a NetVigil application, this authentication string will be retrieved from the database, and it, along

with the password the user gave and the user login name will be passed to your `NetVigilPluginAuthentication.authenticate` method. If your `authenticate` method returns `true`, the user will be allowed to login. Conversely, if `authenticate` returns `false`, the user won't be able to login.

If you've told NetVigil to use a plugin script, when the user logs in, NetVigil will take the user login name, password and the `parameters` attribute from `netvigil.xml`, and will replace the placeholders in the `parameters` attribute with the login name and password. It will then look in the authentication scripts directory for the script named in the `execute` attribute in `netvigil.xml`, and will execute the script with the updated `parameters` attribute. If the script runs successfully, and returns a zero exit code, NetVigil will allow the user to log in. If NetVigil can't run the script, or the script returns with a non-zero exit code, the user will not be allowed to login.