ORNL/TM-2020/1324

An Integrated Two-Factor Authentication Solution Using Pulse Connect Secure and Apache HTTP Server

P.B. Nance
A.S. Bengston

January 20, 2020

Approved for public release.
Distribution is unlimited.

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE US DEPARTMENT OF ENERGY

DOCUMENT AVAILABILITY

Reports produced after January 1, 1996, are generally available free via US Department of Energy (DOE) SciTech Connect.

Website www.osti.gov

Reports produced before January 1, 1996, may be purchased by members of the public from the following source:

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
Telephone 703-605-6000 (1-800-553-6847)
TDD 703-487-4639
Fax 703-605-6900
E-mail info@ntis.gov
Website http://classic.ntis.gov/

Reports are available to DOE employees, DOE contractors, Energy Technology Data Exchange representatives, and International Nuclear Information System representatives from the following source:

Office of Scientific and Technical Information
PO Box 62
Oak Ridge, TN 37831
Telephone 865-576-8401
Fax 865-576-5728
E-mail reports@osti.gov
Website http://www.osti.gov/contact.html

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM-2020/1324

Data System Sciences and Engineering

An Integrated Two-Factor Authentication Solution Using Pulse Connect Secure and Apache HTTP Server

P. Bradford Nance
Adam S. Bengston

January 20, 2020

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831-6283
managed by
UT-BATTELLE, LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

ACRONYMS

2FA	two-factor authentication
ACL	Access Control List
CN	Common Name
DC	Domain Component
HTTP	HyperText Transfer Protocol
IdP	Identity Provider
LDAP	Lightweight Directory Access Protocol
PCS	Pulse Connect Secure
RADIUS	Remote Authentication Dial-In User Service
SAML	Security Assertion Markup Language
TOTP	time-based one-time password
URL	Uniform Resource Locator

ABSTRACT

Complex system requirements can often lead to a situation wherein no single product or platform can provide a solution. In these cases, a solution integrating the features of multiple products and platforms is needed. Recently, a program sponsor required discontinued use of a third-party Security Assertion Markup Language Identity Provider that was being used to implement a two-factor authentication (2FA) solution that fully integrated with Pulse Connect Secure (PCS). PCS alone could not provide the features necessary to meet all the requirements of a new 2FA solution. The challenge was to leverage the features of additional platforms to design an integrated solution to meet all requirements. This paper presents a solution that integrates the features of PCS, Apache HyperText Transfer Protocol Server, and a custom application that meets all requirements, including a requirement to defend against Uniform Resource Locator manipulation as means to gain unauthorized access to backend applications.

1. INTRODUCTION

Complex system requirements often can lead to a situation wherein no single product or platform can provide a solution. In these cases, a solution integrating the features of multiple products and platforms is needed. Recently, a program sponsor required discontinued use of a third-party Security Assertion Markup Language (SAML) Identity Provider (IdP) that was being used to implement a two-factor authentication (2FA) solution that fully integrated with Pulse Connect Secure (PCS). The requirements for the new solution were the following:

1. Provide access to a secure web portal using 2FA that would include a username/password combination as the first factor, and an email passcode challenge as the second factor.
2. Continue using PCS as the primary platform.
3. Implement a solution without using the existing third-party SAML IdP.
4. Implement a solution where the backend applications leverage a HyperText Transfer Protocol (HTTP) request header variable for authentication.
5. Prevent unauthorized access, which is usually achieved using Uniform Resource Locator (URL) manipulation[1], to backend applications for users who have not completed the email passcode challenge.

PCS alone could not provide the features necessary to implement a solution to meet all the requirements. The challenge was to leverage the features of additional platforms to design an integrated solution that met all requirements including the one to prevent unauthorized access through URL manipulation.

2. PCS–ONLY 2FA CONFIGURATION

A PCS–only 2FA configuration can be implemented to satisfy all the requirements (Section 1) except the one to prevent unauthorized access through URL manipulation. The primary components of such a configuration are as follows:

- PCS as the secure web application server that provides a point of presence on the Internet.
- Apache HTTP Server[2] acting as the internal (backend) web server.
- Oracle WebLogic Server acting as the internal (backend) application server.

For a PCS–only configuration, Apache simply acts as the HTTP listener for the applications running on the Oracle WebLogic Server. The authentication flow for this configuration is shown in Figure 1.

[1] Wikipedia refers to URL manipulation as a "Semantic URL attack" (https://en.wikipedia.org/wiki/Semantic_URL_attack) where "a client manually adjusts the parameters of its request by maintaining the URL's syntax but altering its semantic meaning".
[2] The Apache HTTP Server Project (https://httpd.apache.org) "is an effort to develop and maintain an open-source HTTP server for modern operating systems." Its power and feature set can be expanded through the use of modules (https://httpd.apache.org/docs/2.4/mod/) that can be installed and custom configured.

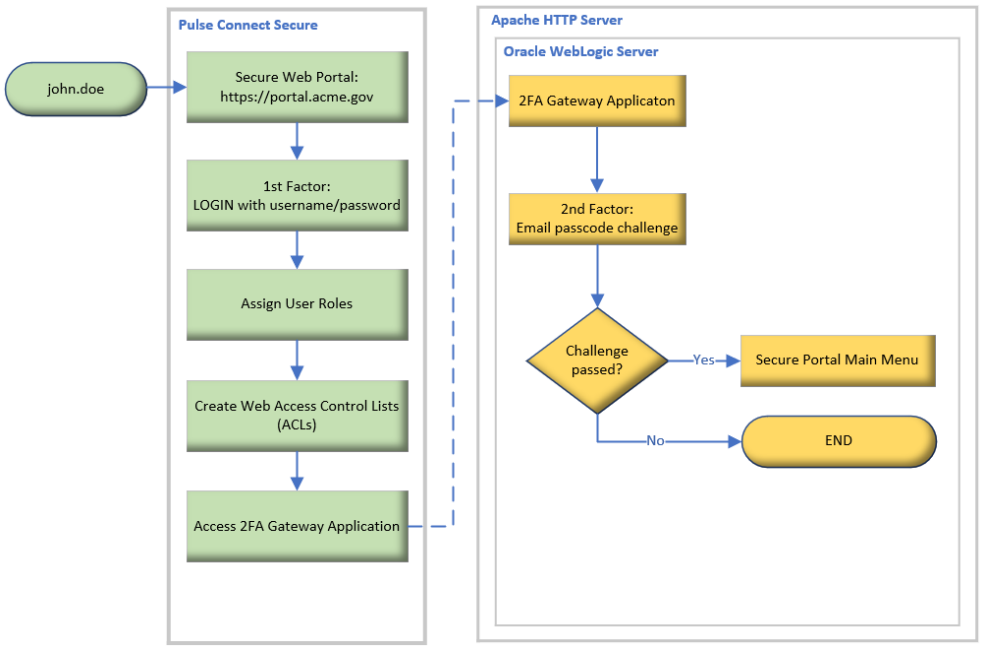

Figure 1. PCS–only configuration authentication flow.

2.1 PCS CONFIGURATION COMPONENTS

To gain a deeper understanding of the PCS–only 2FA configuration and the integrated solution, an understanding of the relevant components of a PCS configuration is necessary. The relevant components are as follows:

- Sign-in Policy – Defines the URL used to access the secure web portal.
- Authentication Realm – Defines the security realm associated with the sign-in policy and includes the following components:
  - Authentication Server – Primary server used as the basis for validating the identity of the user. The supported authentication server types are described later in this section.
  - Additional Authentication Server – A secondary authentication server used to implement a PCS–integrated 2FA solution.
  - User Directory/Attribute Server (Authorization Server) – A secondary authentication server that is not an active part of the authentication process that provides user attributes to the user realm that are used for defining the role mapping rules.
  - Role Mapping Rules – Rules that determine the user roles that are assigned. Usually based upon user attribute or group membership.
- User Roles – Assignment based upon the role mapping rules.
- Web Access Control List (ACL) – Allows access to backend resources based upon assigned user roles.
- Custom Headers – Sends defined HTTP request headers to the backend application based upon assigned user roles. PCS enhances security by preventing custom headers from appearing in the end user's browser.

There are several authentication server types supported by PCS. The relevant server types are as follows:

- Lightweight Directory Access Protocol (LDAP) Server
- Active Directory
- Remote Authentication Dial-In User Service (RADIUS) Server
- Certificate Server
- SAML Server
- Time-based one-time password (TOTP) Server

2.2 PCS–ONLY DETAILED AUTHENTICATION FLOW

The following list describes the authentication flow for the PCS–only configuration:

- User john.doe initiates access to the secure web portal by accessing the URL.
- User john.doe authenticates using the username and password of account on the LDAP server.
- Upon successful authentication, the PCS configuration assigns user roles including those needed to access the 2FA Gateway Application and the secure portal main menu.
- Based upon the assigned user roles, the PCS configuration also does the following:
  - Creates ACLs that allow the user to access the 2FA Gateway Application and the secure portal main menu.
  - Redirects the authenticated portal user to the landing page of the 2FA Gateway Application.
- User john.doe accesses the landing page of the 2FA Gateway Application, then the application generates the random passcode and emails it to the end user.
- User john.doe submits a valid passcode and is redirected to the secure portal main menu.

2.3 URL MANIPULATION VULNERABILITY OF PCS–ONLY CONFIGURATION

The authentication flow displayed in Figure 1 and described in Section 2.2 is vulnerable to URL manipulation. User roles and web ACLs are assigned by PCS at the conclusion of the authentication process. In this example, this is right after a valid username/password is provided but before the email passcode challenge is completed. A PCS configuration does not include the capability to circle back and refresh the user roles after the email passcode challenge is completed. In addition, since the 2FA Gateway Application is a backend application, access cannot be an integrated part of the authentication process.

The following scenario illustrates unauthorized access to the secure portal main menu using URL manipulation:

1. User john.doe accesses URL for secure portal (https://portal.acme.gov).
2. User john.doe authenticates with username/password.
3. User john.doe is assigned user roles for accessing 2FA Gateway Application and portal main menu.
4. User john.doe is redirected to 2FA Gateway Application, which triggers an email to john.doe@acme.gov containing a passcode.
5. Prior to submitting the valid passcode, john.doe manipulates the URL to access the secure portal main menu (https://portal.acme.gov/mainmenu).

3. INTEGRATED SOLUTION

The integrated solution uses existing features of PCS and Apache, along with the custom 2FA Gateway Application, to establish communication between system components to prevent URL manipulation. With communication mechanisms in place, Apache can be configured to implement authorization rules requiring an end user to complete the email passcode challenge before allowing access to backend applications. The mechanisms used to establish communication among PCS, Apache, the LDAP server, and the 2FA Gateway Application include HTTP request headers and LDAP server attributes.

Two different views for the authentication flow of the integrated solution are provided in this section. The first view (discussed in Section 3.1) emphasizes the constructs used for component communication. The second view (discussed in Section 3.2) emphasizes the constructs used to implement the Authorization Rules. More detailed information for the implementation of the Authorization Rules also is included in Section 3.2. Prevention of URL manipulation within the integrated solution is discussed in Section 3.3.

3.1 COMPONENT COMMUNICATION

The component communication required to implement the integrated solution is shown in Figure 2. The communication flow among the components of the integrated solution includes the following:

- User john.doe accesses the sign-in page of the secure web portal.
- User john.doe authenticates using username and password against the john.doe account in the LDAP server user directory.
- PCS is configured to pass the following HTTP request headers on to the Apache HTTP Server:
  - enable_apache_authorization (true|false) – If true, communicates to Apache that the authorization check is active.
  - Authorization (Basic <Base64 encoded username:password of 2fa service account>) – Basic Authentication header that establishes the user context for the LDAP authorization in Apache.
  - application_username – Used to pass the authenticated portal username to the backend applications utilizing HTTP header variable authentication.
- PCS redirects john.doe to the 2FA Gateway Application, which is allowed by the Authorization Rules.
- Access to the 2FA Gateway Application triggers an email to john.doe@acme.gov containing a passcode.

- User john.doe submits the passcode from the email, which triggers the 2FA Gateway Application to add the 2fa_service account to the john.doe group in the 2FAGroups container.
- User john.doe is redirected to the secure web portal main menu, which is allowed by the Apache Authorization Rules.

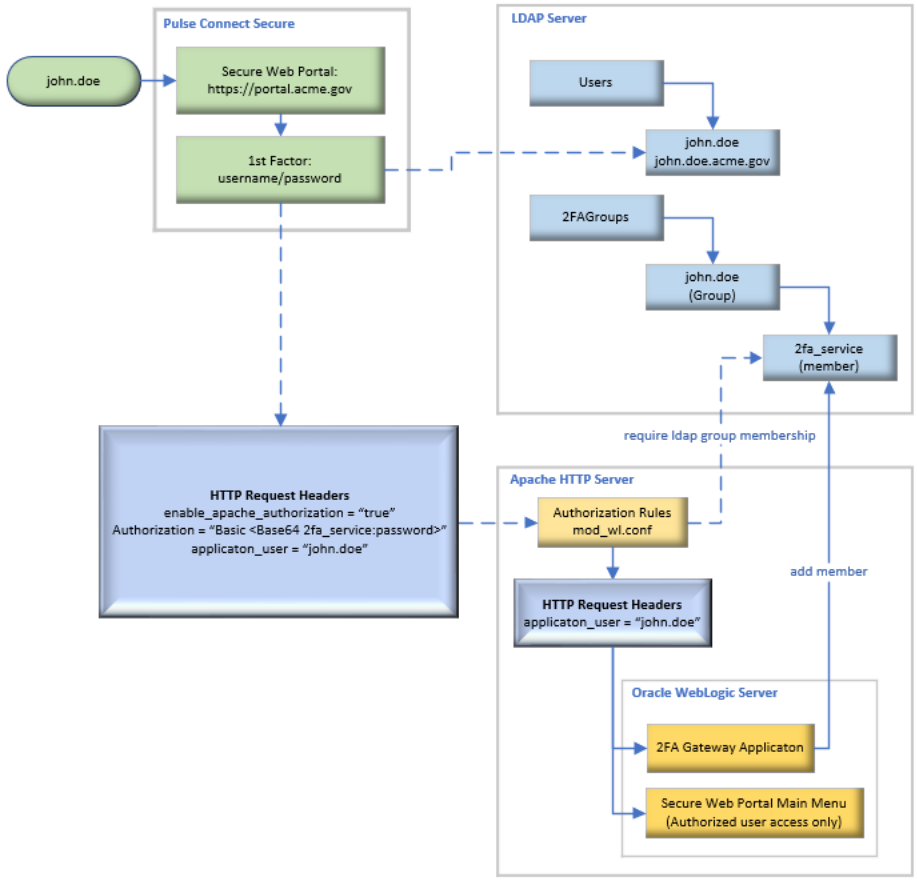

Figure 2. Component communication.

3.2 AUTHORIZATION RULES

This section delineates a view of the authentication flow for the integrated solution with emphasis on the implementation of the Apache Authorization Rules, as shown in Figure 3.

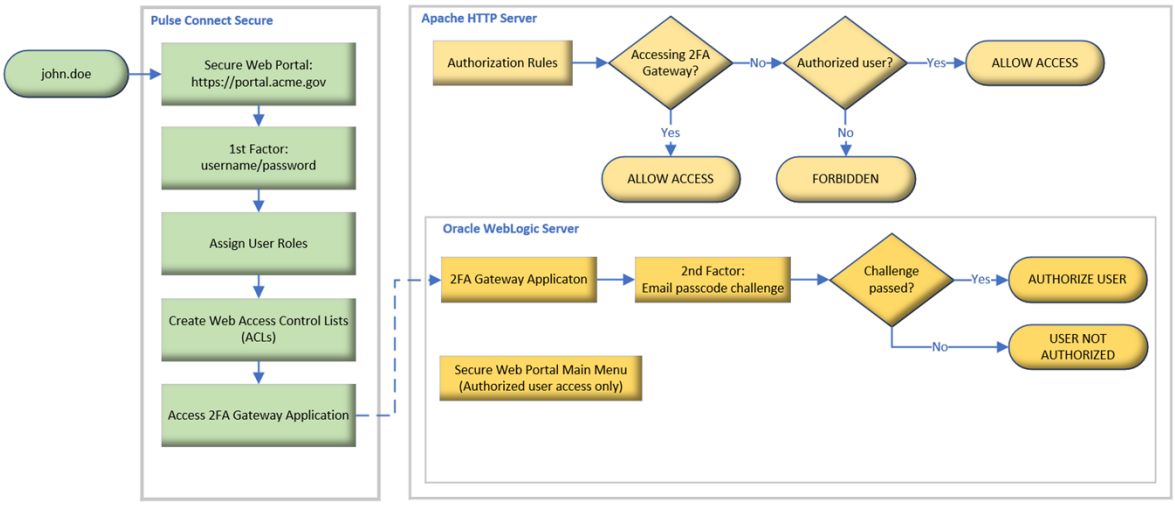

Figure 3. Authentication flow and authorization rules.

The authentication flow and authorization rules shown in Figure 3 include the following actions:

- User john.doe accesses the sign-in page of the secure web portal.
- User john.doe authenticates using the username and password of their account on the LDAP server.
- Upon successful authentication, the PCS configuration assigns user roles including those to access the 2FA Gateway Application and the secure portal main menu.
- Based upon the assigned user roles, the PCS configuration also does the following:
  - Creates ACLs that allow the user to access the 2FA Gateway Application and the secure portal main menu.
  - Sends the custom HTML request headers (listed in Section 3.1) with every request to the backend resources.
  - Redirects the user to the landing page of the 2FA Gateway Application.
- The request to access the 2FA Gateway Application is passed through Apache, which controls all access to the backend applications on the Oracle WebLogic Server through Authorization Rules. These rules are defined in the WebLogic module configuration file mod_wl.conf. The relevant portions of the configuration file are shown in Figure 4.
  - As of Apache Version 2.4, enhancements have been implemented in the configuration file logic for Apache, making it possible to execute IF-THEN-ELSE conditional logic.
  - Apache includes the ap_expr parser that allows HTTP request headers to be evaluated as part of a configuration. For example, the expression "%{HTTP:request_header_name}" would allow Apache to retrieve the value of the "request_header_name" request header.
  - The "IF" condition in the configuration file checks to see if Apache authorization is enabled and if the request is for a backend resource other than the 2FA Gateway Application. If both of these are true, it checks the LDAP to see if the 2fa_service account is a member of the group named john.doe in the 2FAGroups container. This check is made using the "Require ldap-group" directive. If 2fa_service is a member, user john.doe is authorized.
- Access to the 2FA Gateway Application triggers an email to john.doe containing a passcode.

- User john.doe submits the passcode from the email that triggers the 2FA Gateway Application to add the 2fa_service account to the john.doe group in the 2FAGroups container.
- User john.doe is redirected to the secure web portal main menu, which is allowed by the Apache Authorization Rules since user john.doe is authorized.

```
LoadModule weblogic_module /etc/httpd/modules/mod_wl_24.so

LDAPSharedCacheSize 0
LDAPCacheEntries 0
LDAPCacheTTL 0
LDAPOpCacheEntries 0
LDAPOpCacheTTL 0

<IfModule mod_weblogic.c>
  WebLogicHost localhost
  WebLogicPort 7003
</IfModule>

<Location /apex>
  SetHandler weblogic-handler
  Header unset Vary
  RequestHeader unset Authorization
  RequestHeader unset enable_apache_authorization

  <If "%{HTTP:enable_apache_authorization} == 'true' &&
           %{REQUEST_URI} =~ m#/apex/f# &&
           %{QUERY_STRING} =~ m#p=# &&
           %{QUERY_STRING} !~ m#p=(350)(:|$)#">

    AuthType Basic
    AuthName "Secure Portal LDAP"
    AuthBasicProvider ldap
    AuthLDAPURL "ldap://ldap.acme.gov /cn=Admins,dc=acme,dc=gov?cn"
    AuthLDAPBindDN "CN=2fa_service,CN=Admins,DC=acme,DC=gov"
    AuthLDAPBindPassword "<password>"
    AuthzSendForbiddenOnFailure On

    Require ldap-group cn=%{HTTP:application_user},cn=2FAGroups,dc=acme,dc=gov

  </If>

</Location>
```

Figure 4. WebLogic module configuration file (mod_wl.conf).

3.3 URL MANIPULATION PREVENTION IN INTEGRATED SOLUTION

When accessing the secure portal main menu, the logic of the Apache configuration results in an enforcement of LDAP authorization. As previously mentioned, this enforcement is executed with the "Require ldap-group" directive from the Apache configuration file. The user context of the Apache LDAP Authorization Module (mod_authnz_ldap) is determined by the "Authorization" Basic Authentication request header. The value of the header variable takes the "Basic <Base64 encoded username:password of 2fa service account>" form. For this configuration, access is authorized if the 2FA service account (2fa_service) is a member of the group within the "2FAGroups" container with the same name as the authenticating portal user.

If the user attempts to access the secure portal main menu using URL manipulation prior to successfully completing the email passcode challenge, access is forbidden because the required LDAP group membership does not exist. In other words, the authenticated user is not authorized.

4. CONCLUSION

The integrated solution prevents an unauthorized user from accessing a backend application via URL manipulation by integrating the features and capabilities of PCS, Apache, and the custom 2FA Gateway Application. Additional controls could be put in place to make this solution even more secure, including the following examples:

- User session data could be stored using the LDAP or other underlying data storage, such as a database table. This would prevent a user from bypassing the email passcode challenge once a designated amount of time has passed since they last authenticated.
- The 2FA Gateway Application could be enhanced to set an expiration time for a generated passcode or lock the underlying user account in the LDAP when the user submits an invalid passcode *x* number of times.
- HTTP request headers that are used by Apache, but are not needed by Oracle WebLogic, can be deleted as an enhancement to the Authorization Rules. In the mod_wl.conf file, commands can be added to unset headers that fall into this category. These "RequestHeader unset. . ." commands are included in the example configuration file shown in Figure 4.