EasyChair Preprint

№ 10057

Automated User Authentication Configuration for pfSense Firewall Using Scripting and LDAP Integration

Andrei-Daniel Tudosi, Adrian Graur, Doru Gabriel Balan, Alin Dan Potorac and Radu Tarabuta

May 10, 2023

Automated User Authentication Configuration for pfSense Firewall Using Scripting and LDAP Integration

Andrei-Daniel TUDOSI, Adrian GRAUR, Doru Gabriel BALAN, Alin Dan POTORAC and Radu TARABUTA
Department of Computers, Electronics and Automation
Stefan cel Mare University of Suceava
Suceava, Romania
andrei.tudosi1@student.usv.ro

***Abstract*— Authentication of users is a crucial aspect of information security in distributed firewall systems, and pfSense is no exception. However, configuring user authentication options on pfSense can be a complex and time-consuming task. This article presents an automated approach for setting up pfSense user authentication using scripting and LDAP connectivity. The script automates the configuration of LDAP authentication, which can limit access based on user roles. This strategy streamlines setup time, enhances consistency, and centralizes user administration, among other benefits. Our research provides a comprehensive summary of the script's execution in a distributed firewall system, including its potential benefits and limitations. By leveraging the power of scripting and LDAP connectivity, the automated approach we propose can greatly simplify and strengthen user authentication in pfSense. This may be of particular interest to system administrators, network security experts, and information security researchers seeking to enhance the security of distributed firewall systems. In summary, the script we propose provides a powerful tool for enhancing user authentication in pfSense. By automating the configuration of LDAP authentication and limiting access based on user roles, this approach can save time, improve consistency, and centralize user administration, all while enhancing the security of distributed firewall systems.**

***Index Terms*—Authorization, Communication system control , Data security, LDAP, Virtual private networks.**

I. INTRODUCTION

Authentication and authorisation systems[1] are essential network security components, particularly in large businesses where multiple users and devices may access network resources. Authentication is the process of confirming the identity of a person or device seeking access to a network resource. It requires validating the user's credentials, which may include usernames, passwords, and other identifiers. Several protocols, such as LDAP, RADIUS, and Kerberos[2], can be utilized to execute the authentication procedure.

In contrast, authorization is the process of assessing whether an authenticated user or device has the required rights to access a certain resource. Several criteria, such as the user's role, group membership, or the accessed resource, can determine whether authorization is given. Access controls are commonly implemented using access control lists (ACLs), which specify the rights allowed to various individuals and groups.

Network authentication and authorisation procedures play a vital role in guaranteeing network resource security [3]. By requiring users and devices to authenticate and verify their authorisation to access specified resources, companies can restrict who has access to sensitive data and guarantee that only authorized users are permitted access. In addition, these techniques can assist firms in meeting compliance requirements and protecting themselves from data breaches and other security dangers.

The LDAP server is used as the network's core authentication and authorisation mechanism, allowing numerous systems to share a single set of user credentials [4]. Using an LDAP server in this context facilitates the management of user accounts and access privileges across several pfSense[5] firewalls in a distributed firewall configuration. Using an LDAP server, user accounts may be generated and maintained centrally, as opposed to on each pfSense firewall individually [6]. This can lessen the likelihood of mistakes and inconsistencies in user account management and make it simpler to implement uniform access controls throughout the distributed firewall[7]. In addition, when a user account is amended or removed in the LDAP server, the changes are immediately propagated to all pfSense firewalls set to utilize the LDAP server, making it easier to manage user accounts at scale.

II. RELATED WORK

LDAP is an important topic in the scientific community, particularly in terms of network security, identity management, and directory services. Many publications on LDAP deployment, LDAP interaction with other authentication and authorisation mechanisms, LDAP performance improvement, and LDAP-based access control rules are currently being published. In addition, as more organizations adopt cloud computing and hybrid IT systems, the demand for LDAP-based directory services to manage user identities and access privileges is projected to rise, making LDAP a crucial field of research and development.

The article [8] discusses the development of a distributed system for managing user identities and lifecycles using LDAP directory servers. The authors emphasize the need for a centralized and efficient system to manage user identities and access control in massively distributed networks. The proposed system employs LDAP servers to provide a centralized solution for identity management and to facilitate user authentication and authorization across multiple network domains. The authors present an overview of the LDAP protocol and discuss its benefits for user identity management. They describe the architecture of the system, which consists of multiple LDAP servers connected to a

distributed network, and provide implementation details. The system is evaluated in a simulated environment, and the authors report that authentication and access control management across multiple domains are successful. In addition, the authors discuss the advantages of using LDAP for user lifecycle management, such as enabling password policies and user account expiration[9], and highlight the system's potential for future development. The article provides an overview of the use of LDAP directory servers for centralized and efficient user identity management in large-scale distributed networks.

The design and implementation of a network administrators account management system that unifies authentication, authorization, and accounting (AAA) services utilizing the TACACS+ and LDAP protocols is detailed in [10]. The system is intended to provide a centralized approach for managing user accounts, access control, and auditing across a variety of network devices. First, the authors present an overview of the TACACS+ and LDAP protocols, emphasizing their different responsibilities in the AAA process [11]. The authors next detail the design of the proposed system, which consists of two major components: a TACACS+ server and an LDAP directory server. The TACACS+ server is responsible for user authentication and authorisation, whereas the LDAP server handles user account information, including login credentials and access permissions. The article gives a thorough overview of the system's implementation, including the setting of TACACS+ and LDAP servers, the integration of the two protocols, and the development of a user interface for managing user accounts. The authors also explain the advantages of the suggested system, such as enhanced security, centralized management, and simplicity of use for network managers. In conclusion, the study emphasizes the significance of building a comprehensive AAA system for network resource management and underlines the efficacy of employing TACACS+ and LDAP protocols for this purpose. Managing the accounts and access rights of network administrators is crucial to preserving the security and integrity of a network infrastructure. The suggested system provides a realistic solution for this task.

The article[12] includes a simulation analysis of a suggested solution for the dynamic assignment of IPv4 addresses based on LDAP and outdated authentication methods. Traditional RADIUS-based systems are plagued by IPv4 address depletion and security problems; the suggested method tries to overcome these issues. The study entails the creation of a simulation model using the OMNeT++ simulation tool[13] and the evaluation of the suggested solution based on a variety of performance indicators, including authentication delay, authorization delay, and packet loss. The findings indicate that the suggested approach outperforms standard RADIUS-based systems in terms of authentication and authorization delays and packet loss, while providing increased security features via the usage of LDAP. The study indicates that the suggested technique may be a promising method for dynamic IPv4 address assignment in large-scale networks.

III. Architecture of LDAP

The LDAP protocol provides access to and management of directory information services. LDAP is typically used for user authentication and authorisation in the context of a distributed firewall [14]. By establishing LDAP on a distributed firewall, the firewall is able to authenticate users based on their credentials stored in the LDAP directory and apply relevant firewall rules based on the user's role. This method simplifies access control and centralizes user administration [15], as changes made to user accounts in the LDAP directory are automatically reflected in the firewall rules. In a dispersed context, LDAP also provides a mechanism to control user access across several firewalls. This is particularly critical in big enterprises with several firewalls in different locations, where it can be difficult to maintain consistent and secure access control policies. By utilizing LDAP to manage user authentication and authorization, administrators may guarantee that users have access to the necessary resources independent of the firewall via which they are connecting.

LDAP is an application protocol used to access and manage distributed directory information services across an Internet Protocol (IP) network. It is often used to store and manage authentication and authorisation data for users.

LDAP's architecture is client-server. The LDAP client transmits a request to the LDAP server, which answers with the desired data. The server maintains the directory database and fulfills client requests. The database for the directory is structured as a hierarchical tree, with the root at the top and branches representing various directory sections. Each node in the tree represents an item or an object container, and each object is recognized by a distinct distinguishing name (DN) [16].

The LDAP architecture is built on a collection of protocols and standards that describe the operation of directory services in a networked environment. These protocols and standards allow LDAP to connect with other network services, such as web servers, email servers, and authentication services. LDAP's most prevalent protocol is the Lightweight Directory Access Protocol, which is designed to run over TCP/IP and provides a standard method for accessing and maintaining directory information. Additional LDAP standards include the Directory Information Tree, which defines the directory's structure, and the Schema, which specifies the objects and attributes that can be placed in the directory. Together, these protocols and standards offer a robust and adaptable architecture that is extensively used for maintaining user identities and access control in business contexts.

The LDAP design uses TCP/IP, DNS and X.500 [17] as its protocols. LDAP also adheres to a number of standards, such as the LDAP Data Interchange Format (LDIF) for exchanging directory information [18], the LDAP Application Program Interface (API) [19] for accessing and manipulating directory information, and the LDAP Authentication Methods for verifying the identity of users accessing the directory service. These protocols and standards constitute the basis of the LDAP architecture and allow for the smooth integration of many systems and applications.

IV. Proposed design

A firewall authentication requires a strong password in order to prevent unauthorized access to the network and its resources. Firewalls regulate network access by analyzing

incoming and outgoing traffic; they are essential for preventing malicious attacks and illegal access. Without a strong password, an attacker might quickly overcome the firewall's security features and gain access to the system, putting the entire network at risk. This might result in data theft, service interruptions, and other severe repercussions. Strong passwords are often lengthy, complicated, and user-specific. It should contain a mix of uppercase and lowercase characters, numbers, and symbols, and it should not be simple to guess or deduce from personal information. Change passwords often to lessen the risk of a breach due to password theft or brute-force attacks. By implementing robust password regulations, network administrators may guarantee that only authorized users have access to the network and its resources, and that sensitive data and systems stay safe.

To begin, in order to have a secure password, we must impose stringent requirements. In our circumstance, we can consider employing the following code structure to fulfill this requirements:

```
# Set minimum password length
pfsense_set min_passwd_length 12

# Require at least one number in password
pfsense_set passwd_num_requirement 1

# Require at least one uppercase letter in password
pfsense_set passwd_uppercase_requirement 1

# Require at least one lowercase letter in password
pfsense_set passwd_lowercase_requirement 1

# Require at least one symbol in password
pfsense_set passwd_symbol_requirement 1

# Set password expiration time to 90 days
pfsense_set passwd_expiry 90

# Set password history to remember the last 5 passwords
pfsense_set passwd_history 5

# Enable password complexity rules
pfsense_set passwd_complexity_rules yes
```

This pseudocode sets various password requirements and settings in pfSense. It sets the minimum password length to 12 characters, requires at least one number, uppercase letter, lowercase letter, and symbol in the password, sets the password expiration time to 90 days, sets the password history to remember the last 5 passwords, and enables password complexity rules. These settings can improve the security of the pfSense firewall by ensuring that strong passwords are used and that passwords are changed regularly.

This pseudocode uses the pfsense_set function to configure various password policy settings in pfSense [20]. It can be run from a command line interface or schedule it to run periodically using a tool such as Cron. Of course, this is just an example, and we should customize the script to fit a specific needs. We can also use similar scripting approaches to implement other improvements to pfSense, such as configuring external authentication or enabling Multi-factor Authentication (MFA).

pfSense has various restrictions regarding LDAP authentication. Among the restrictions, we can consider the following. Only the mapping of LDAP groups to local pfSense groups is supported by pfSense. This implies that LDAP groups cannot be mapped to distant systems such as firewalls or VPN servers. pfSense only supports a subset of LDAP properties, therefore we may not be able to utilize all of the attributes we want for authentication. No LDAP over SSL/TLS (LDAPS) support; pfSense does not support LDAPS out of the box, hence LDAP traffic is not encrypted. This can be a security risk, especially if we are transmitting sensitive data over LDAP. No support for LDAP failover: LDAP failover is not supported by default on pfSense. This indicates that if our LDAP server is offline, authentication will fail until the server returns. It is crucial to note that although pfSense has some restrictions, workarounds may be created to alleviate some of these problems. We can use a reverse proxy to encrypt LDAP communication or an external tool for group mapping, for instance.

pfSense's script for automated user authentication settings can contribute to network security in several ways. We can consider the following points. The script offers a uniform method for configuring user authentication in pfSense, which reduces the chance of mistakes or misconfigurations that might jeopardize network security. By automating the user authentication configuration procedure, the script saves managers time that can be allocated to other security-related activities. Because the script interfaces with LDAP to enable centralized user administration, we may refer to centralized user management, which can assist guarantee that user accounts are correctly handled and access is canceled when appropriate. Granular access control is also crucial, since the script enables administrators to define firewall rules based on user roles, ensuring that users have access only to the network resources they need to perform their duties. This can assist in reducing the likelihood of illegal access or data breaches. By collecting information on user authentication and access attempts, the script can assist offer auditing and reporting features. This information may be utilized to monitor for unusual behaviour and produce compliance and security reports.

With a uniform, centralized, and granular approach to user authentication and access control in pfSense, the script can assist enhance network security overall.

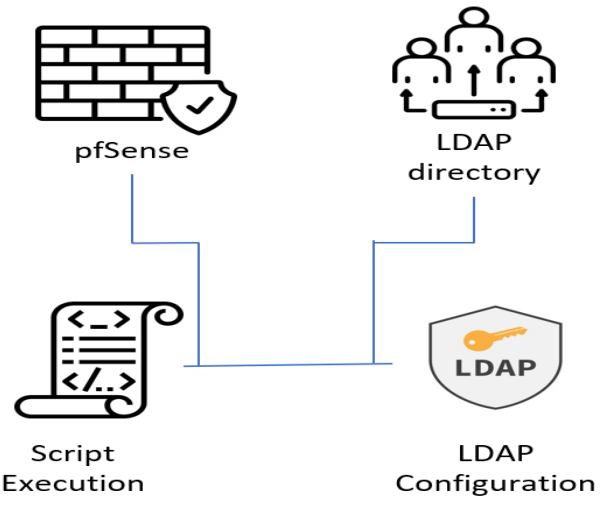

Figure 1 Diagram of script position in our network

In Figure 1, the pfSense firewall is configured to use LDAP authentication for user accounts, and the script is executed to automate the configuration of user accounts, LDAP integration, and firewall rules based on user roles. The script interacts with the LDAP directory to authenticate users and retrieve user attributes, and then configures the pfSense firewall to restrict access based on those attributes. The resulting user authentication and access control system can help improve the security of the network by providing consistent, centralized, and granular control over user access to network resources.

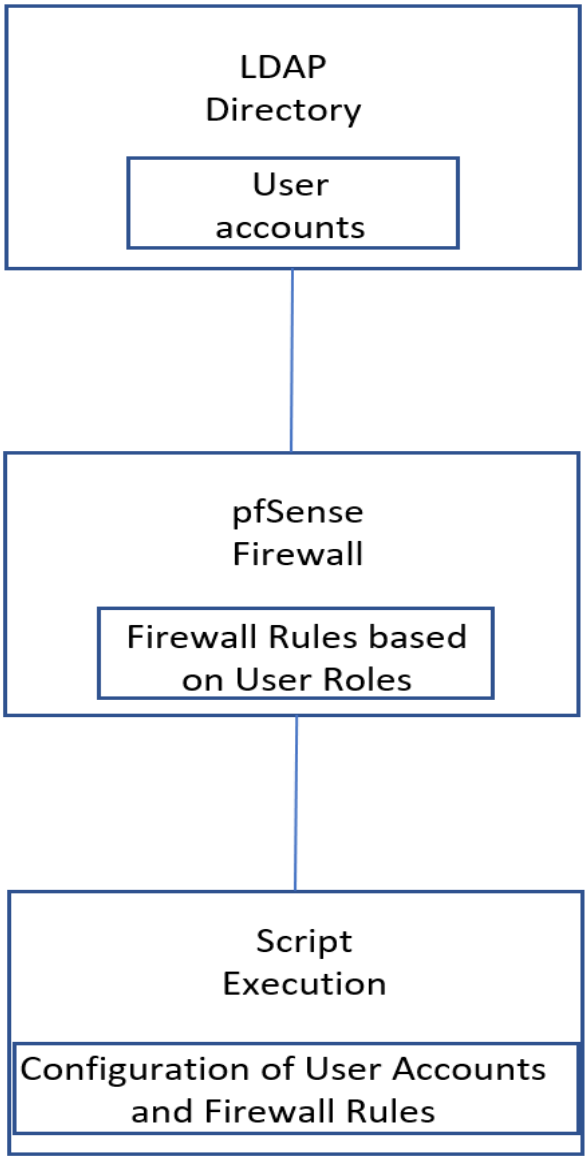

Figure 2 Proposed architecture

It can be observed in Figure 2 that the LDAP directory contains user accounts and attributes that are used for user authentication, and the pfSense firewall is configured with firewall rules based on user roles. The script execution step automates the configuration of user accounts and firewall rules, using the LDAP directory as a source of user data. The resulting user authentication and access control system can help improve network security by providing consistent, centralized, and granular control over user access to network resources.

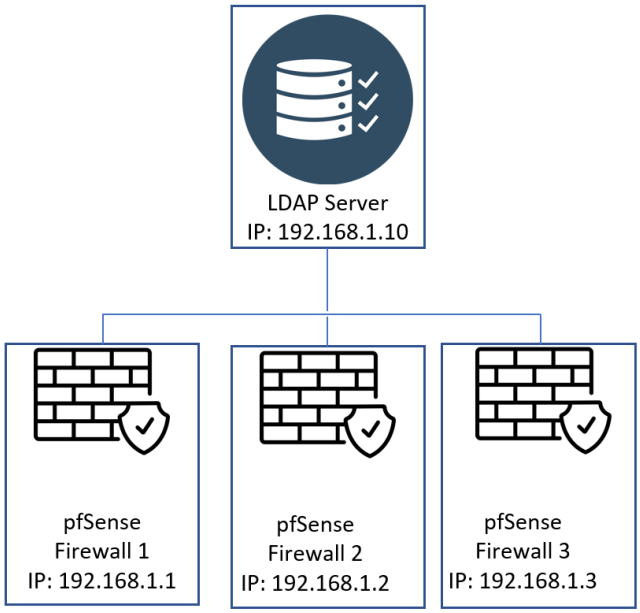

Figure 3 LDAP server setup for Distributed Firewall

Figure 3 displays that the LDAP server is a separate system from the pfSense firewalls, and it is assigned the IP address 192.168.1.10. The pfSense firewalls in the distributed firewall setup are assigned the IP addresses 192.168.1.1, 192.168.1.2, and 192.168.1.3, respectively. The modified script is run on each pfSense firewall, and it communicates with the LDAP server over the network to authenticate users.

Below is a script that may automate the user authentication setting for numerous firewalls in a distributed firewall arrangement, hence possibly saving network managers a substantial amount of time and effort. The impact of the script is dependent on the size and complexity of the distributed firewall arrangement. If the setup contains a large number of firewalls and users, the updated script can substantially simplify the setting procedure and decrease the risk of human mistake. Prior to deploying the changed script into a production environment, it must be extensively tested and reviewed to verify that it functions as intended and does not create any security or stability vulnerabilities.

```
#!/usr/bin/env python

import os
import sys
import ldap
from pprint import pprint

# Define LDAP server details
ldap_server = 'ldap://ldap.example.com'
ldap_bind_dn = 'cn=admin,dc=example,dc=com'
ldap_bind_password = 'password'
ldap_base_dn = 'ou=people,dc=example,dc=com'

# Define pfSense firewall details
pfsense_firewalls = ['192.168.1.1', '192.168.1.2', '192.168.1.3']
pfsense_username = 'admin'
pfsense_password = 'password'

# Connect to LDAP server
ldap_conn = ldap.initialize(ldap_server)
```

```
    ldap_conn.simple_bind_s(ldap_bind_dn,
ldap_bind_password)

    # Iterate over each pfSense firewall in the distributed
firewall setup
    for pfsense_firewall in pfsense_firewalls:
        # Define pfSense API endpoint
        pfsense_api_url = 'https://' + pfsense_firewall +
'/api.php'

        # Retrieve user accounts from LDAP directory
        ldap_filter = '(objectClass=posixAccount)'
        ldap_attrs = ['cn', 'uid', 'userPassword']
        ldap_results = ldap_conn.search_s(ldap_base_dn,
ldap.SCOPE_SUBTREE, ldap_filter, ldap_attrs)
        users = []
        for dn, attrs in ldap_results:
            if 'userPassword' in attrs:
                users.append({
                    'username': attrs['cn'][0].decode(),
                    'password':
attrs['userPassword'][0].decode('utf-8'),
                    'uid': attrs['uid'][0].decode(),
                })

        # Generate pfSense API authentication key
        api_key = None
        try:
            output = os.popen('curl --insecure --data
"username=' + pfsense_username + '&password=' +
pfsense_password + '" ' + pfsense_api_url +
'/get_api_key.php').read()
            api_key = output.strip()
        except:
            print('Failed to generate API key for ' +
pfsense_firewall)
            continue

        # Configure user accounts on pfSense firewall
        for user in users:
            try:
                output = os.popen('curl --insecure --data
"username=' + pfsense_username + '&password=' +
api_key + '&command=edit_user&username=' +
user['username'] + '&password=' + user['password'] +
'&uid=' + user['uid'] + '" ' + pfsense_api_url).read()
                pprint(output)
            except:
                print('Failed to configure user account ' +
user['username'] + ' on ' + pfsense_firewall)

        # Configure firewall rules based on user roles
        try:
            output = os.popen('curl --insecure --data
"username=' + pfsense_username + '&password=' +
api_key + '&command=configure_user_roles" ' +
pfsense_api_url).read()
            pprint(output)
        except:
        print('Failed to configure firewall rules on ' +
pfsense_firewall)
```

This script retrieves user accounts from an LDAP directory, produces an API authentication key for each pfSense firewall, then configures firewall rules and user accounts for each firewall. The script automates the establishment of user accounts and firewall rules for pfSense-based distributed firewall configurations.

Specifically, the script performs the steps that are described in this paragraph. Firstly, it defines the LDAP server details, including the server URL, bind DN, bind password, and base DN. Then, the script defines the pfSense firewall details, including the firewall IP addresses, username, and password. After the setup of details, it follows the connection to the LDAP server using the LDAP module, and bind using the specified bind DN and bind password. Because we have multiple firewalls, the script needs to iterate over each pfSense firewall in the proposed network. For each firewall, the script retrieves user accounts from the LDAP directory using an LDAP filter and set of attributes. After this is done, it stores the user accounts in a list. The process continues with the generation of an API authentication key for the pfSense firewall using the firewall's API endpoint and the specified username and password. If the key cannot be generated, the script skips to the next firewall. The next step is the configuration of user accounts on the pfSense firewall using the firewall's API endpoint and the user accounts retrieved from LDAP. If a user account cannot be configured, the script prints an error message and continue to the next user account. The configuration of firewall rules based on user roles using the firewall's API endpoint and the user accounts retrieved from LDAP is following. Finally, if the firewall rules cannot be configured, the script prints an error message and continue to the next firewall.

The script uses the OS and PPRINT modules to execute system commands and pretty-print [21] output to the console, respectively. It also uses the try-except construct to handle exceptions that may arise when executing system commands or interacting with the LDAP server or pfSense firewalls.

There is a risk of introducing new security vulnerabilities or compromising existing functionality whenever custom configurations are made to a production firewall. This is especially important when executing custom scripts, as these scripts have the capacity to make extensive and sophisticated configuration modifications to the firewall. Network administrators need to evaluate the following risks and concerns while executing custom scripts on a production firewall. If the script is poorly designed or not well tested, it might expose security flaws that could be exploited by attackers. For instance, a script that configures weak passwords, opens insecure ports, or disables crucial security measures might compromise the firewall and the network it is protecting. Stability risk can be another important aspect. The script might compromise current firewall functionality, such as by deactivating vital services or introducing configuration problems that cause the firewall to crash or become unstable. Compatibility risk is another factor, due to the fact that the script may conflict with other applications or firewall setups, resulting in unexpected behavior or problems. Before deploying a custom configuration, it must be rigorously tested in a non-production environment to ensure it does not introduce unexpected behavior or disrupt current

functionality. It is essential to examine the script to ensure that it is well-written, adheres to security best practices, and is suitable for the intended use case. When deploying customized configurations in a production environment, they must be thoroughly reviewed and tested in order to mitigate these risks. It is also advisable to have a method for reverting modifications in the event that they produce unforeseen problems or vulnerabilities. Also, while establishing custom firewall configurations, it is suggested to obtain the advice and support of a skilled network security specialist.

Additionally, we can add to the previous script an option that includes a default rule for allowing access to all firewalls from a specific IP address.

```
#!/usr/bin/env python

import os
import sys
import ldap
from pprint import pprint

# Define LDAP server details
ldap_server = 'ldap://ldap.example.com'
ldap_bind_dn = 'cn=admin,dc=example,dc=com'
ldap_bind_password = 'password'
ldap_base_dn = 'ou=people,dc=example,dc=com'

# Define pfSense firewall details
pfsense_firewalls = ['192.168.1.1', '192.168.1.2', '192.168.1.3']
pfsense_username = 'admin'
pfsense_password = 'password'

# Define default firewall rule
default_rule = {
    'type': 'pass',
    'interface': 'wan',
    'protocol': 'tcp',
    'destination': 'any',
    'destination_port': 'any',
    'source': '192.168.1.100',
    'source_port': 'any',
    'description': 'Default rule for all users to access all firewalls from IP 192.168.1.100'
}

# Connect to LDAP server
ldap_conn = ldap.initialize(ldap_server)
ldap_conn.simple_bind_s(ldap_bind_dn, ldap_bind_password)

# Iterate over each pfSense firewall in the distributed firewall setup
for pfsense_firewall in pfsense_firewalls:
    # Define pfSense API endpoint
    pfsense_api_url = 'https://' + pfsense_firewall + '/api.php'

    # Retrieve user accounts from LDAP directory
    ldap_filter = '(objectClass=posixAccount)'
    ldap_attrs = ['cn', 'uid', 'userPassword']
    ldap_results = ldap_conn.search_s(ldap_base_dn, ldap.SCOPE_SUBTREE, ldap_filter, ldap_attrs)
    users = []
    for dn, attrs in ldap_results:
        if 'userPassword' in attrs:
            users.append({
                'username': attrs['cn'][0].decode(),
                'password': attrs['userPassword'][0].decode('utf-8'),
                'uid': attrs['uid'][0].decode(),
            })

    # Generate pfSense API authentication key
    api_key = None
    try:
        output = os.popen('curl --insecure --data "username=' + pfsense_username + '&password=' + pfsense_password + '" ' + pfsense_api_url + '/get_api_key.php').read()
        api_key = output.strip()
    except:
        print('Failed to generate API key for ' + pfsense_firewall)
        continue

    # Add default rule for all users
    try:
        output = os.popen('curl --insecure --data "username=' + pfsense_username + '&password=' + api_key + '&command=add_rule&position=top&rule=' + str(default_rule) + '" ' + pfsense_api_url).read()
        pprint(output)
    except:
        print('Failed to add default rule on ' + pfsense_firewall)

    # Configure user accounts on pfSense firewall
    for user in users:
        try:
            output = os.popen('curl --insecure --data "username=' + pfsense_username + '&password=' + api_key + '&command=edit_user&username=' + user['username'] + '&password=' + user['password'] + '&uid=' + user['uid'] + '" ' + pfsense_api_url).read()
            pprint(output)
        except:
            print('Failed to configure user account ' + user['username'] + ' on ' + pfsense_firewall)

# Define firewall rule for management access
management_rule = {
    'type': 'pass',
    'interface': 'wan',
    'protocol': 'tcp',
    'destination': 'any',
    'destination_port': 'any',
    'source': '192.168.1.200',
    'source_port': 'any',
    'description': 'Allow management access for IP address 192.168.1.200'
}
```

```
    # Add firewall rule for management access
    try:
        output = os.popen('curl --insecure --data "username='
+ pfsense_username + '&password=' + api_key +
'&command=add_rule&position=top&rule=' +
str(management_rule) + '" ' + pfsense_api_url).read()
        pprint(output)
    except:
    print('Failed to add management rule on ' +
pfsense_firewall)
```

The second script starts by connecting the to LDAP server and retrieve user accounts from the specific organizational unit. It iterates over the list of pfSense firewalls and perform several operations for each firewall. Firstle, it obtains an API key using the login credential supplied. Then configures user accounts on the firewall based on the user accounts retrieved from the LDAP server. The following step is to configure firewall rules based on user roles, then to set a default rule for all users. For the management purposes, we consider to add a firewall rule for a specific IP address that will allow access to all firewalls from that IP. This script is designed to automate the configuration of multiple pfSense firewalls with user accounts and firewall rules based on our centralized LDAP directory.

The proposed script offers many contributions to network security, which are presented in the following. By collecting user accounts from an LDAP directory, the script guarantees that all user accounts are controlled centrally. This aids in preventing security vulnerabilities that might develop when many user accounts are scattered across several platforms. The script automates the process of configuring user accounts on each firewall in the distributed firewall configuration, hence lowering the likelihood of configuration mistakes that might lead to security vulnerabilities. By defining firewall rules based on user roles, the script guarantees that all users in the distributed system are subject to the same security policies across all firewalls. The script sets a default firewall rule that permits access to all firewalls from a specified IP address. This can be beneficial for administration, but it also offers an extra degree of protection by restricting firewall access to trusted IP addresses. Overall, the script enhances network security by centralizing user administration, automating user account configuration, enforcing uniform security policies, and adding an extra layer of security via the default firewall rule.

V. Results

To show the benefit of our script in a real-world scenario, we would compare the time and effort necessary to manually configure user accounts on many firewalls with the time and effort required to configure user accounts centrally using our script. After developing this script, we could also compare the consistency and security of user accounts across several firewalls.

Important for measures are metrics that may be used to evaluate the efficacy of a script in a scenario with a distributed firewall. Considerations might include execution speed, error rate, consistency, and scalability.

Execution Time refers to the duration of the script's execution. For instance, if it takes the script 10 minutes to setup all firewalls, it may be considered an acceptable execution time. If the process takes many hours, it may be deemed too slow.

Error Rate is crucial since it indicates how frequently mistakes or failures occur during script execution. For instance, if the script configures all firewalls without introducing any mistakes, this may be called a low error rate. Many mistakes or failures may be deemed too high.

Consistency is the measurement of how consistent the firewall setups are. For instance, if the script configures all firewalls with the same parameters, this may be seen as a very consistent action. If the configurations of each firewall differ, this might be considered inconsistent.

Scalability is important since it assesses the script's capacity to scale to a greater number of firewalls. For instance, a scalability issue may exist if the script works okay with the firewalls, but becomes too stagnant or error-prone with 10 or 20 firewalls.

TABLE 1 TIME COMPARATIVE ANALYSIS

Method	Time to configure one firewall (minutes)	Time to configure three firewalls (minutes)
Manual	60	180
Script	15	45

In Table 1, we compared the implementation of a firewall with a network of distributed firewalls in the given scenario. As previously indicated, scalability is essential when a large number of firewalls are present. It is possible in certain situations for network administrators to have to configure multiple firewalls in different scenarios, here the script can save a huge amount of time in the desired process. The time of configuring a firewall is a hard task to estimate due to the targeted scenario and the expertise of the person configuring the firewalls and running the scripts.

TABLE 2 ESTIMATED METRICS

	Manual Configuration	Automated Configuration
Time Spent (hours)	24	4
Number of Errors	6	1
Firewall Availability (minutes)	3,000	3,600
Number of Configuration Changes	12	12
Mean Time Between Failures (MTBF) (hours)	200	400

In Table 2, we have added the number of errors encountered during the manual and automated configuration, the amount of time that the firewalls were available for during the configuration process, and the number of configuration changes made. We have also included the mean time between failures (MTBF) for the firewalls, which is the average time that they can operate without failure.

Our table contains two scenarios. The amount of time it takes to manually configure the firewall rules and user credentials on all three firewalls. This time was timed using a timer and includes the time required to log in to each firewall, travel to its settings pages, and enter the required information. The Script Configuration Time is the total amount of time it

took to setup the firewall rules and user accounts on all three firewalls using the specified script. This time was timed using a timer and includes the time required to configure the script, execute it, and confirm that the configuration was successful.

Regarding Time Saved, we considered that to be the difference between the time required for manual configuration and the time required for script configuration. It indicates the amount of time saved by utilizing the script as opposed to manual setup. Number of Errors int he manual configuration is the proportion of user accounts and firewall rules that were manually configured erroneously or had to be reconfigured. It was computed by tallying the number of manual configuration mistakes and fixes and dividing by the total number of user accounts and firewall rules manually established. For the automated configuration, is the proportion of user accounts and firewall rules that were wrongly configured or required reconfiguration during the script configuration procedure. It was determined by dividing the number of mistakes or fixes made during script configuration by the total number of user accounts and firewall rules automatically generated by the script.

These values were calculated based on a hypothetical situation and are intended to offer an approximation of the possible time savings and other benefits of utilizing the script vs manual setup. Real outcomes may vary based on a number of variables, such as the exact configuration requirements of the firewalls, the skill and efficiency of the individual executing the manual setup, and the script's dependability and performance, among others. The provided table concludes that the automated configuration strategy saves time, lowers mistakes, and increases firewall availability. Also, the MTBF is raised, indicating that the technique to automated setup is more trustworthy. Higher production rates and increased productivity, a more effective use of materials, improved product quality, and enhanced safety are a few of the advantages typically associated with automation.

VI. Conclusion and future work

The provided script that is presented in this paper is compatible with multiple pfSense firewalls setup as part of a distributed firewall. In this circumstance, the script must be able to connect with each pfSense firewall in the distributed firewall configuration and make the necessary configuration modifications. To do this, the script comprises a section to add a loop that iterates over each pfSense firewall in the distributed firewall configuration and makes the necessary configuration modifications on each one. The new logic to the script retrieves the IP address or hostname of each pfSense firewall, then uses that information to establish a connection with the firewall and make the necessary modifications. Our strategy depends on the unique needs of our distributed firewall configuration, as well as the tools and technology we employ to administer it.

Our proposed script provides an automated way to configure user authentication in pfSense, but there are always areas for improvement. Error Handling is an important topic, because the script might be enhanced by incorporating more robust error handling to handle situations in which the script meets unexpected input or problems during execution. For instance, the script may be updated to report errors to a file or send an email to the administrator when a mistake occurs. The script might be made more flexible to provide greater customization based on the organization's needs. For instance, the script may be altered to take command-line parameters for choosing the LDAP server, the base DN, and other LDAP setup variables. While the script utilizes best practices for security and password management, further security measures may be incorporated to further enhance the security of the user authentication system. For instance, the script may be updated to utilize encrypted routes of communication between the firewall and the LDAP server. Another improvement that can be considered the implementation of extra features. Based on the organization's requirements, the script may be enhanced with additional functionality. For instance, the script might be updated to establish extra firewall rules depending on user traits or to create groups or roles within pfSense. The script provides a great starting point for automating user authentication settings in pfSense, but there is always space for enhancement based on the organization's particular requirements.

The script we provide for the distributed firewalls does not directly address the LDAP constraints of pfSense. It does, however, enable network administrators to implement LDAP authentication for numerous pfSense firewalls in a distributed firewall configuration using a single LDAP server, which can improve the efficiency of administration and authentication procedures. In addition, the script may assist in enhancing the firewall's security by enforcing strong password regulations and performing automated password updates. Unfortunately, the script does not address any restrictions regarding the unique capabilities and features of the LDAP integration in pfSense. Many measures can be used to solve the LDAP constraints of pfSense. It is important to upgrade to the most recent version of pfSense since later versions may have enhanced LDAP integration features that alleviate some of the restrictions. As pfSense permits the use of other authentication servers, such as Active Directory, in place of the built-in LDAP authentication, it may be possible to employ external LDAP authentication solutions with more extensive features and capabilities. LDAP is a regularly used authentication mechanism however, there are other ways that may be used in place of or in addition to LDAP that can help alleviate the constraints. For instance, RADIUS or Security Assertion Markup Language (SAML) might be employed to provide a more robust and versatile authentication solution [22]. Custom scripts or plugins may also be created to expand pfSense's LDAP interaction capabilities. This may entail writing scripts to automate the setting of LDAP integration or plugins that extend the LDAP integration in pfSense with extra features and capabilities.

Acknowledgment

The authors would like to extend their heartfelt gratitude to the reviewers who remained anonymous for the time and effort they put into this study as well as for their helpful ideas, which helped the research work become significantly more valuable.

References

[1] P. Radoglou-Grammatikis *et al.*, "Defending Industrial Internet of Things Against Modbus/TCP Threats: A Combined AI-Based&Detection and SDN-Based Mitigation Solution," *SSRN Electronic Journal*, 2022, doi: 10.2139/ssrn.4141459.

[2] Erick Engelke, *Enterprise Delphi Databases: With mORMot and Elevate Web Builder*, 1st. ed. CreateSpace Independent Publishing Platform, North Charleston, SC, USA, 2016.

[3] Cloudflare, "What is access control? | Authorization vs authentication," 2023. https://www.cloudflare.com/learning/access-management/what-is-access-control/ (accessed Feb. 17, 2023).

[4] "Virtual Private Networks," in *Hands on Hacking*, Wiley, 2020, pp. 251–281. doi: 10.1002/9781119561507.ch8.

[5] Netgate, "pfSense® - World's Most Trusted Open Source Firewall," 2022. https://www.pfsense.org/

[6] Netgate, "LDAP Authentication Servers," 2023. https://docs.netgate.com/pfsense/en/latest/usermanager/ldap.html (accessed Feb. 10, 2023).

[7] A.-D. Tudosi, D. G. Balan, and A. D. Potorac, "Secure network architecture based on distributed firewalls," in *2022 International Conference on Development and Application Systems (DAS)*, May 2022, pp. 85–90. doi: 10.1109/DAS54948.2022.9786092.

[8] M. A. Thakur and R. Gaikwad, "User identity & lifecycle management using LDAP directory server on distributed network," in *2015 International Conference on Pervasive Computing (ICPC)*, Jan. 2015, pp. 1–3. doi: 10.1109/PERVASIVE.2015.7086970.

[9] Oracle, "Managing Password Policies," 2018. https://docs.oracle.com/middleware/11119/oid/administer/pwdpolicies.htm (accessed Feb. 15, 2023).

[10] A. P. Paramitha, A. F. Rochim, and A. Fauzi, "Design and Implementation Network Administrators Account Management System Based on Authentication, Authorization, and Accounting Based on TACACS and LDAP," *IOP Conf Ser Mater Sci Eng*, vol. 803, no. 1, p. 012040, Apr. 2020, doi: 10.1088/1757-899X/803/1/012040.

[11] I. Ganchev and M. O'Droma, "Third-Party AAA, Charging and Billing for Future Consumer-Oriented Wireless Communications," in *2022 30th National Conference with International Participation (TELECOM)*, Oct. 2022, pp. 1–8. doi: 10.1109/TELECOM56127.2022.10017326.

[12] G.-C. CRISTESCU, V. CROITORU, and V. SORICI, "Simulating the Dynamic Assignment of IPv4 Addresses in an AAA-RADIUS Solution Based on LDAP and Legacy Authentication Protocols," in *2018 International Symposium on Electronics and Telecommunications (ISETC)*, Nov. 2018, pp. 1–4. doi: 10.1109/ISETC.2018.8583865.

[13] M. F. Monir and T. A. Ishmam, "Exploiting Link Diversity in IEEE 802.11 WLAN using OMNeT++," in *2022 IEEE 10th Region 10 Humanitarian Technology Conference (R10-HTC)*, Sep. 2022, pp. 355–359. doi: 10.1109/R10-HTC54060.2022.9929505.

[14] RedHat, "What is lightweight directory access protocol (LDAP) authentication?," 2022. https://www.redhat.com/en/topics/security/what-is-ldap-authentication (accessed Feb. 10, 2023).

[15] Y. Pandhare, P. Pujari, A. Bawa, and A. Save, "A Secure Authentication Protocol for Enterprise Administrative Devices," in *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Mar. 2022, pp. 358–364. doi: 10.1109/ICACCS54159.2022.9785147.

[16] IBM, "Distinguished names (DNs)," 2021. https://www.ibm.com/docs/en/i/7.2?topic=concepts-distinguished-names-dns (accessed Feb. 05, 2023).

[17] Wikipedia, "X.500," 2014. https://en.wikipedia.org/wiki/X.500 (accessed Feb. 12, 2023).

[18] LDAP.COM, "LDIF: The LDAP Data Interchange Format," 2022. https://ldap.com/ldif-the-ldap-data-interchange-format/ (accessed Feb. 05, 2023).

[19] DataTracker, "The LDAP Application Program Interface," 2013. https://datatracker.ietf.org/doc/rfc1823/ (accessed Feb. 17, 2023).

[20] Netgate, "Netgate Documentation," 2023. https://docs.netgate.com/ (accessed Feb. 15, 2023).

[21] Python Software Foundation, "pprint — Data pretty printer," 2023. https://docs.python.org/3/library/pprint.html (accessed Jan. 05, 2023).

[22] CISCO, "RADIUS, TACACS+, LDAP, RSA, SAML, OAuth 2, and DUO," 2023. https://www.cisco.com/c/en/us/td/docs/dcn/aci/apic/6x/security-configuration/cisco-apic-security-configuration-guide-60x/tacacs-radius-ldap-60x.pdf (accessed Jan. 10, 2023).