The Universal Fog Proxy: A Third-party Authentication Solution for Federated Fog Systems with Multiple Protocols

Asad Ali, Ali Utkan Şahin, Öznur Özkasap, and Ying-Dar Lin

Abstract

Fog computing is suitable for latency constrained applications useful to end users and IoT devices in smart cities, factories, and homes. A federation among fogs is beneficial for subscribers and providers in terms of enhanced capability, capacity, coverage, and services. To realize such a federation, a third-party authentication mechanism among fog providers is required, so that a subscriber of a fog can access the services provided by the other fogs without having to create new accounts. In this article, we propose a transparent and standard-compliant universal fog proxy that provides third-party authentication among OpenID Connect (OIDC), 802.1x, and Protocol for Carrying Authentication for Network Access (PANA) without requiring a new protocol. The proxy consists of virtual counterparts of the entities involved in these protocols so that it provides transparency. For example, when a fog using OIDC receives an authentication request, the proxy relays and behaves as a virtual Identity Provider (vIdP) for the fog using OIDC and a virtual supplicant for the fog using 802.1x. We applied our solution to nine scenarios across OIDC, 802.1x, and PANA. Experimental results show that the proxy takes 4–52 percent of the total authentication time of 0.128–3.504s for nine scenarios, with a larger percentage in scenarios involving OIDC due to multiple re-directions among virtual components. The scenarios involving 802.1x take a considerably longer time, though a low percentage (4–12 percent) by the proxy, as the spanning tree protocol in an 802.1x switch takes about one second to converge when adding a new device to the network.

Introduction

Fog is essentially a small cloud that provides control, computing, networking, and storage functions closer to a user [1]. Although various terms (e.g., edge computing, mobile computing, and mobile cloud computing) are used for similar computing paradigms, we use the fog computing definition provided by the Industrial Internet Consortium [1]. Fog computing has the advantage of having lower latency over cloud computing, as it brings the services provided by cloud computing closer to users through processing units located at the network's edge [2]. This makes fog computing suitable for applications such as augmented reality, virtual reality, and self-driving vehicles. Fog computing is useful for both traditional devices and Internet of Things (IoT) devices for providing different types of low-latency services, storage, and computation.

Fog Federation

A fog service provider provides data, computing, storage, and application services to end-users which are similar to the cloud. However, the difference between a cloud and a fog is that a cloud is a centralized system, while a fog is a distributed decentralized infrastructure. The major characteristics of a fog that distinguish it from a cloud are dense geographical distribution, mobility support, and closer proximity. With multiple fog service providers, each providing different types of services with various capacities and capabilities, two main problems arise: end users have to subscribe to different fog service providers separately and maintain multiple accounts; and providers face the risk of losing subscribers if they are not able to provide a service required by their users. A solution to these problems already exists for cloud service providers where they share their resources with each other via a federation.

However, a cloud federation cannot be directly used in a fog environment because a cloud can be accessed via the Internet from anywhere, which is not the case with a fog as it is widely distributed. Some fog service providers that provide computational services via a LAN require a user to be in range of the access point (for example 802.1x or PANA) to access resources. Therefore, the authentication protocols of clouds cannot always be used in a fog environment and establishment of a trusted federation among fogs needs translation among multiple protocols. Such a federation among fogs is needed to enhance their capacity, capability, and coverage, and it will allow end-users to maintain a single account while providing lower-latency as compared to federated clouds. As a result, low-latency applications will be able to reap the fruits of this federation. Fog computing has application in smart cities, smart factories, smart hospitals, and smart homes [3] and federation among fogs will be useful in such settings.

Authentication and Latency Issues

In order to access fog services, users need to authenticate themselves first. Although federation among fogs is beneficial for both service providers and end users, an authentication issue arises: When a subscriber of one fog (source fog, where the user has an account) wants to access the services provided by another fog (target fog, where the user does not have an account), the subscriber needs to

Asad Ali and Ying-Dar Lin are with the National Yang Ming Chiao Tung University (NYCU); Ali Utkan Şahin and Öznur Özkasap are with Koç University.

Digital Object Identifier:
10.1109/MNET.111.2100168

0890-8044/21/$25.00 © 2021 IEEE

authenticate itself with the target fog while migrating from source to target fog. The authentication process can be different for different fogs as they may use different authentication protocols. Some fogs may support OpenID Connect (OIDC) for third-party authentication, while some may not support OIDC and only support protocols such as 802.1x and Protocol for Carrying Authentication for Network Access (PANA). Such differences among fog service providers create an issue in case of third-party authentication where the user accesses a target fog via its source fog. If a user opts to create a new account at a target fog, subscription to every fog introduces a one-time latency.

Research Questions

In the light of the issues identified, the questions that we aim to answer with this research are: Is it achievable to authenticate a user with a target service provider via third-party authentication? If yes, how would one coordinate multiple authentication protocols while keeping the process transparent and secure? How much latency does our proposed solution result in? There are a few studies in the literature that propose fog federation or fog authentication for security, but none of them solves the problem of third-party authentication in fog.

The Universal Fog Proxy

In order to solve the authentication and latency issue, we propose a transparent and standard-compliant universal fog proxy that provides third-party authentication among fogs using different authentication protocols, such as OIDC, 802.1x, and PANA protocols. PANA and 802.1x are used for both user and device authentication while OIDC is only for users. We consider the user's authentication in this work, and hence, we propose solutions for PANA, 802.1x, and OIDC. The proxy consists of a proxy controller along with virtual counterparts of entities in OIDC, 802.1x, and PANA. The reason for using virtual counterparts is to have multiple roles for different fogs as per their authentication protocols, so that our proposed universal fog proxy would appear transparent to them without proposing any modifications. Proxy also retains pairwise state for multiple authentication pairs as differentiation among them is required. The main contributions of this work are summarized as follows:

- We propose a transparent universal fog proxy that allows a user to access multiple fogs with a single account and enables federation among multiple fog service providers that have different authentication protocols such as 802.1x, PANA, and OIDC.
- The proposed proxy provides translation among multiple authentication protocols and provides multiple login options for a user to select accordingly.
- The universal fog proxy is standard-compliant and secure.

The remainder of this article is organized as follows. The next section provides the background to existing authentication protocols in a fog computing paradigm, along with related work. Following that, we state the problem with an example. We then document the proposed design and architecture for the proposed universal fog proxy. The implementation of the prototype, modules, and testbed is presented following that, and then we give the results and evaluation along with a security analysis. The final section concludes the article and provides some insight for future work.

Related Work

There exist multiple fog service providers that can use different protocols for authentication, such as OIDC, 802.1x, and PANA. OIDC is a widely-used protocol that provides third-party authentication solutions on top of the OAuth 2.0 protocol by allowing users to access an application by authenticating themselves with another application. This is currently not very widely spread across the fog computing paradigm, but in time, OIDC will not only be adopted in clouds or in mobile environments but will also be widely adopted by fog computing [4]. The entities involved in OIDC are a user, a relying party (RP), and an OpenID provider (IdP or OP). The 802.1x protocol comes into play when a user needs to connect to another user over a local area network (LAN) or a wireless local area network (WLAN).

The entities involved in 802.1x are a supplicant (i.e., a client end user), an authenticator (i.e., usually a switch), and a RADIUS authentication server (AAA), which authenticates the supplicant. PANA is another protocol, similar to 802.1x, which is widely used in resource-constrained Zigbee devices as a user-authentication solution, and provides authentication to a user that wants to connect to a network, and uses EAP for key distribution, authentication, and key agreement. The entities involved in PANA are a PANA Client (PaC), a PANA Authentication Agent (PAA), and a RADIUS authentication server (AAA Server). We consider these three protocols in our implementation and experiments.

A third-party federated authentication solution needs to combine existing authentication protocols and glue them together, so that the fog service providers do not have to change their existing infrastructure and protocols. A comparison of the protocols used for federation and authentication in fog environments is provided in Table 1. These studies are compared on the basis of their objectives, the protocols involved, and whether they provide authentication and federation.

There are some studies [5–7] in the literature that propose a fog collaboration or federation with the objective of load distribution, revenue maximization, and efficiency. However, these studies do not consider a case where a user needs third-party authentication. Some studies [8-10] provide authentication with the objectives of security, privacy, and efficiency, but none of them provides federated authentication, and they propose proprietary protocols for authentication in a fog environment. Other studies [11–14] propose authentication in a fog environment by using existing schemes such as challenge-response authentication, and existing protocols such as 802.1x, PANA, and OIDC, but they do not propose a federation among multiple fog service providers and do not provide translation among multiple protocols for creating an authentication federation among these providers. The solution we propose is a transparent universal fog proxy that provides federation and third-party authentication for fogs using multiple existing authentication protocols without proposing any changes to these protocols. To the best of our knowledge, no work has proposed a transparent proxy that provides federation and third-party authentication for fogs using multiple authentication protocols.

Problem Statement

In this section, we state the problem and describe the authentication model. We assume that there exist multiple fog service providers, each using different authentication protocols, including OIDC, 802.1x, and PANA. A user needs an account to access the services provided by different providers, as authentication is indispensable for the security of fog computing. So to resist network attacks and improve the security of communication, authentication needs some sort of account and credentials. The fog nodes deployed by the same vendor do not require re-authentication: we consider cases where a user with an account with a service provider needs to access resources provided by another service provider that also needs an account for authentication.

We assume that there is a user (first party) that has an account on one of the fog service providers, called the source fog (second party), who wants to access the services provided by another fog service provider, called the target fog (third party). The user should not need to create a new account as this introduces latency and is inconvenient for the user. Therefore, there is a need for some mechanism whereby the user can access the target fog services through source fog credentials. The user account can reside in any fog service provider and use any of the above mentioned three protocols for authentication. Thus, the problem is as follows:

- *Given:* We have multiple fog service providers using multiple authentication protocols including OIDC, 802.1x, and PANA.
- *Objective:* To achieve federation and third-party authentication with low latency.
- *Constraints:* The solution must provide transparency, protocol translation, and security without the need to create new accounts.

Assumptions: We assume that there are multiple fog service providers A, B, C, and D using different authentication protocols. In our model, providers A, B, and C support OIDC for third-party authentication while provider D does not support OIDC and rather uses 802.1x and PANA for authentication. We also assume that providers A and B are in an authentication federation, but C and D lie outside this authentication federation. It must be noted that although provider C supports OIDC, we assume that provider C is not registered as the relying party (RP) with provider A (i.e., IdP). This assumption is valid because a user can log in to Spotify using its Apple, Facebook, or Google account, but it cannot log into the Google account using its Facebook or Apple account. Hence, it is entirely possible that there are multiple fog service providers that individually support OIDC by acting as IdPs, but it is not always the case that all fog service providers will allow login via all other fog service providers that support OIDC, that is, act as relying parties.

We also assume that a user is originally the subscriber of service provider A (source fog provider), and when the user migrates to B, C, or D, these will be the target service providers. As A and B are in an authentication federation, the user can easily access the services provided by provider B, but when the user migrates to service provider C or D, the user would need to register itself with them, which would require the user to buy multiple subscriptions. In order to avoid this, a third-party authentication mechanism is needed, so that the user can access the services of all providers without having to obtain new subscriptions. This third-party authentication is possible if there is a federation among these service providers. It should be noted that access control is not in the scope of this article, as it is handled by the protocols themselves. For example, 802.1x allows access control lists to be defined and configured for the authenticated users.

Scheme	Objective	Authentication	Federation	Protocols
Federated fog [5]	Load distribution	×	✓	N/A
Fog federation [6]	Revenue maximization	×	✓	N/A
CFC-IoV [7]	Energy efficiency	×	✓	N/A
CLASC [8]	Security, Efficiency	✓	×	N/A
LAAP [9]	Security, Privacy	✓	×	Proprietary
EADA [10]	Security, Efficiency	✓	×	Proprietary
FOCUS [11]	Security	✓	×	Challenge-Response Authentication
IoT security [12]	End-to-end security	✓	×	802.1x
ANASTACIA [13]	Security orchestration in SDN/NFV-enabled Fog	✓	×	PANA
Self-sovereign OIDC [14]	Decentralized authentication	✓	×	OIDC
Ours Universal Fog Proxy	Third-party Authentication, Federation	✓	✓	OIDC, 802.1x, PANA

TABLE 1. Methods for federation and authentication in fog systems.

This gives us nine scenarios for authentication as we consider three protocols which can be either on the source side or the target side. There are several issues that arise when we consider these nine scenarios:

- Third-party authentication — the authentication of a user between multiple fog service providers using its source credentials.
- Intermediate communication — design of an entity that acts as a communication intermediary between multiple fog service providers.
- Protocol gluing — a method to glue different authentication protocols like OIDC, 802.1x, and PANA together. The components involved in these authentication protocols also have to be taken into account. The message flows of these authentication protocols are different from each other, which causes a message mismatch. We need to solve this mismatch in order to achieve third-party authentication by gluing them together.

Solution: The Universal Fog Proxy

To solve the issues discussed in the previous section, we propose a universal fog proxy that resides between multiple fog service providers and supports multiple protocols. The main design idea of the proxy is transparency, so that the service providers do not have to change their existing protocols or modify their infrastructure. In order to keep the proxy design transparent, we propose virtual counterparts inside the proxy that play dif-

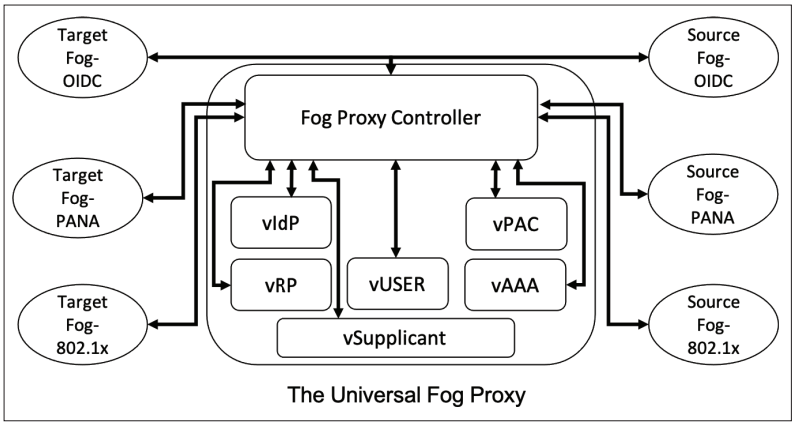

FIGURE 1. Universal fog proxy architecture.

ferent roles while connecting with the fog service providers. The proxy plays these roles according to the authentication components of the involved protocols such as user, RP, IdP, supplicant, AAA, PaC, and PAA. The proxy also keeps the soft state for a user according to the pair of protocols that are on the source side and the target side.

Architecture

The architecture of the universal fog proxy is shown in Fig. 1. The design of the proxy components is in accordance with the authentication components of OIDC, 802.1x, and PANA. The reason for selecting these components is to make the proxy appear transparent to both source and target service providers. Hence, the components inside the proxy are virtual user (vUser), virtual relying party (vRP), virtual OpenID provider (vIdP), virtual supplicant (vSupplicant), virtual authentication server (vAS or vAAA), and virtual PANA Client (vPaC). A proxy controller is also designed to control these components and act as a communication intermediary among them and the source and target service providers. The third-party authentication solution is hosted in this proposed universal fog proxy which can be deployed by a third-party or a fog service provider itself.

The components inside the proxy are the virtual counterparts of the authentication entities in the OIDC, 802.1x, and PANA protocols. Depending on the deployed authentication protocol at the target fog, the proxy will activate either the vIdP (for OIDC), or the vAAA (for 801.x and PANA) at the target side. Similarly, depending on the protocol deployed at the source fog, either the vSupplicant (for 802.1x), the vUser (for OIDC), or the vPaC (for PANA) will be activated at the source. The proxy controller is responsible for the coordination of these virtual components.

Third-Party Authentication Message Flow

We describe one of the nine possible scenarios, as shown in Fig. 2, by dividing it into *request* and *response* stages. We explain the message flow of one scenario and the rest of the scenarios will follow the same approach, but with different messages. While naming a scenario (e.g., 802.1x-to-OIDC), the source-side protocol appears first (i.e., 802.1x) and the target-side protocol appears later (i.e., OIDC).

802.1x-to-OIDC: This scenario is shown in Fig. 2. The request stage begins when the user contacts the OIDC relying party (RP) at the target fog. Once the user chooses to authenticate itself through vIdP, the target RP redirects the user to the vIdP log in endpoint where the vIdP asks for the user's credentials and lets the user choose the source fog that it is subscribed to. If the source fog uses 802.1x as its authentication protocol, the credentials are transferred to the vSupplicant by the proxy controller. The vSupplicant then performs an EAP authentication with the source switch on the user's behalf. For optimization purposes, as soon as the proxy controller is initialized, the vSupplicant initiates the 802.1x authentication and lets the source-side switch wait for a response. The vSupplicant is then able to directly supply the credentials to the source-side switch without any initialization overheads.

During the response stage, the source AAA replies with a RADIUS Accept message after authenticating the vSupplicant. The proxy controller instructs the vIdP to send a vIdP consent page to

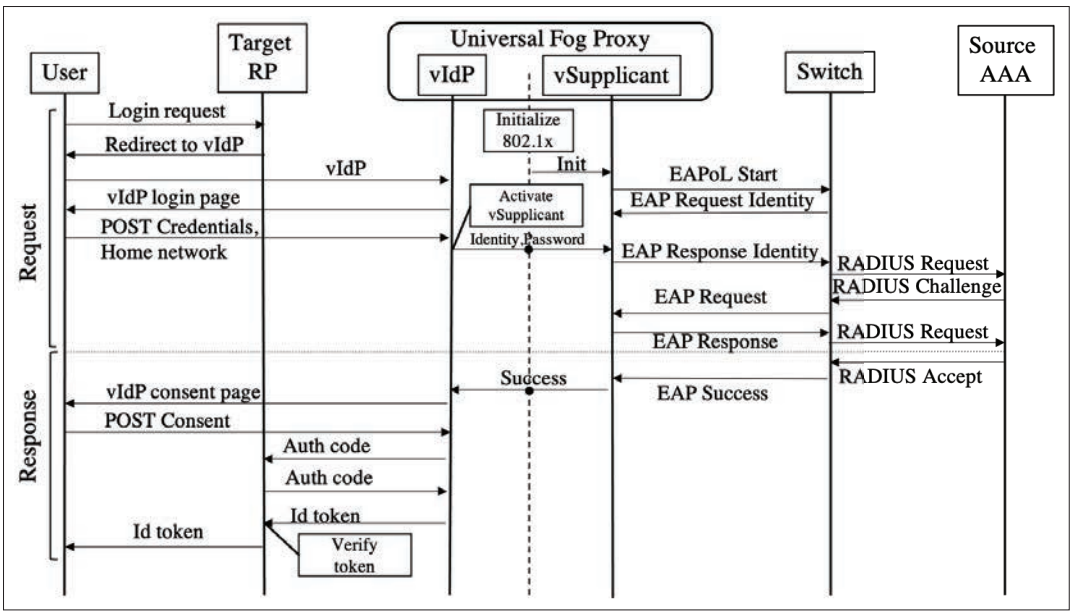

FIGURE 2. Third-party Authentication message flow for 802.1x-to-OIDC.

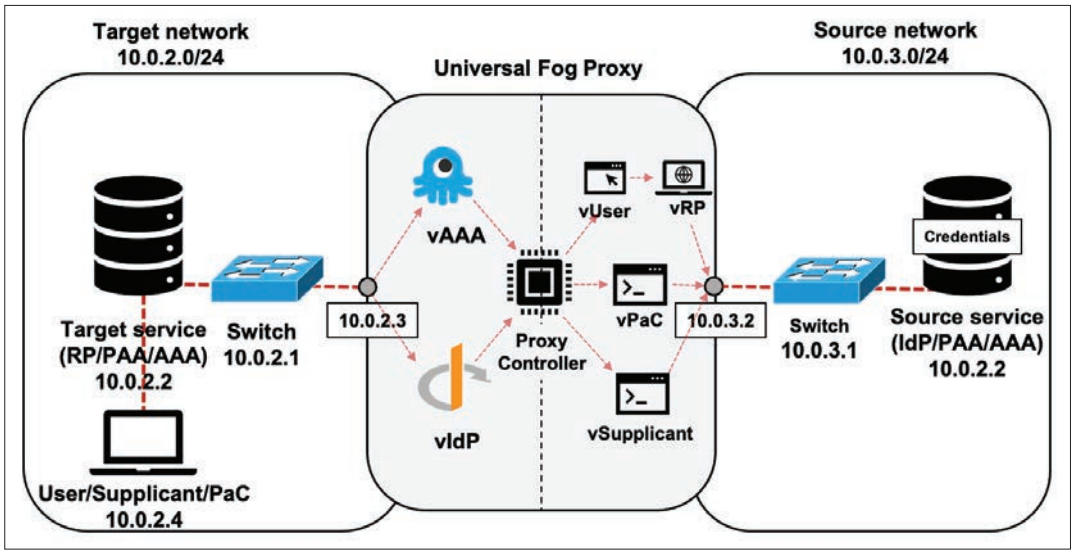

FIGURE 3. Experimental testbed: Universal Fog Proxy.

the user. Once the user gives consent, the target RP acquires the authorization code and the identity token from the vIdP. Finally, the user is able to access the services provided by the target fog without creating a new account. There are eight more scenarios for which we have designed solutions, namely 802.1x-to-802.1x, 802.1x-to-PANA, OIDC-to-802.1x, OIDC-to-OIDC, OIDC-to-PANA, PANA-to-802.1x, PANA-to-OIDC, and PANA-to-PANA. These scenarios are similar to the flow shown in Fig. 2, but with different messages depending upon the involved protocols on the source side and target side.

Security Analysis

Rios *et al.* [15] identified the main threats in fog systems as data manipulation and replay attacks, data leakage, impersonation, and denial-of-service attacks. In our security analysis, we primarily consider a threat model where the user is potentially malicious. We assume that an unauthorized third party may attack the network both actively and passively. All the other entities in the system are trusted. The protocols considered in this work support certificate-based authentication and digital signatures (through TLS), thus man-in-the-middle and data manipulation attacks are mitigated both at the source side and the target side. For the same reason, it is infeasible for an entity to imitate the proxy. Replay attacks are also mitigated due to the existence of nonce utilized by TLS.

In a typical TLS execution, the certificate is provided to the client (e.g., the user in Fig. 2) by the server (e.g., vIdP in Fig. 2) during the handshake phase. Thus, the client does not need to have access to the certificate beforehand. The client verifies the certificate by using the public key of the certificate authority that has issued it. The client terminates the communication if it cannot verify the certificate. We assume that the user already has access to these public keys since most browsers by default come with pre-installed public keys of common certificate authorities (e.g., VeriSign, COMODO). This security measure does not affect our proposed transparent solution as the purpose of transparency is to adhere to the existing message flows of OIDC, 802.1x, and PANA in the fog service providers so that they do not have to modify the existing protocols. The usage of certificates does not modify these message flows and ensures that our proposed solution is transparent.

It is possible for an eavesdropper to perform packet-sniffing, but the confidentiality of the user's credentials is secured through encryption provided by the TLS protocol. It must be noted that although the user's credentials are always transferred in an encrypted fashion, at the proxy, the credentials must be decrypted and encrypted again with the public key of the opposite side (i.e., source or target side). We consider the proxy as an honest and trusted entity to realize the end-to-end encryption between the user and the source fog service provider. A malicious user may trivially flood a proxy with authentication initiation messages. Each new incoming request spawns a new thread at the proxy, which makes the solution weak against a trivial denial-of-service attacks.

Implementation

We implemented the universal fog proxy for OIDC, 802.1x, and PANA, and executed nine different scenarios on a testbed, as shown in Fig. 3. The testbed consisted of six different devices of which two were Cisco 2960-X switches and the remaining four were computers. One switch was used at the source network and the other at the target network. The first computer was used as the target fog service provider and included a target 802.1x authentication server (AAA), a target OIDC RP, and a target PAA to accommodate all three protocols at the target side. The second computer was used as a source fog service provider and included the source 802.1x AAA, the source PAA, and the source OIDC IdP. The third computer was used as the 802.1x supplicant, OIDC user, and PaC. Finally, the fourth computer was used as the universal proxy on all of the federated scenarios. We have published our implementation on GitHub (https://github.com/utkn/Fog-Fog-Federation).

We utilized the open-source IETF compliant OpenPANA (https://github.com/OpenPANA/openpana) project in our PANA components (i.e., PaC, PAA, and vPaC). Virtual supplicant (vSupplicant) and 802.1x supplicant were implemented using wpa supplicant (https://github.com/digsrc/wpa supplicant) daemon, which is an IETF-compliant standard UNIX software. OIDC components

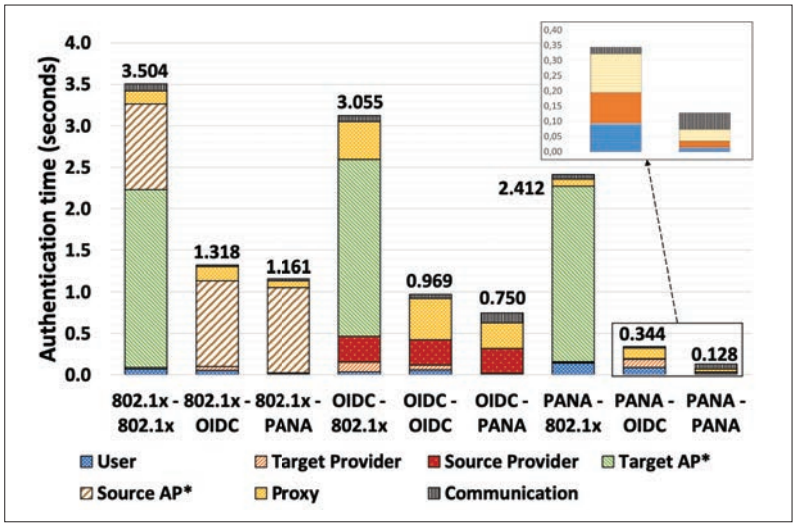

FIGURE 4. Latency for all scenarios and latency breakdown.

were deployed with two different standard open-source OIDC implementations: the source IdP was implemented with Django OIDC Provider (https://github.com/juanifioren/django-oidc-provider), and the remaining OIDC components (i.e., vRP, vIdP, and target RP) were implemented using AuthLib (https://github.com/lepture/authlib). Authentication servers (i.e., AAA and vAAA) were implemented with FreeRADIUS (https://github.com/FreeRADIUS/freeradius-server) which is a widely-used open-source RADIUS server. In our experiments with PANA at either the source network or the target network, we integrated the authentication servers into PAAs. These authentication servers were simple FreeRADIUS servers running locally on the PAA device. In the implementation, the credentials of the users were not stored at the target fog, and each login was identical. For the scenarios where we had the 802.1x at source, we deployed a two-login process to provide captive portal functionality. In this case, the first authentication did not grant any access to the user except at the captive portal.

Results and Evaluation

After implementation, we measured the third-party authentication time per scenario to determine the time taken by each of them. We then divided the time measurements into sections to see how much time was taken by the user, the target service provider, the source service provider, and the proxy. We then carried out a detailed bottleneck analysis for all the scenarios. Finally, we investigated the difference between the time taken with our proxy-based authentication and the time taken if a user had to authenticate separately with both parties.

Latency for All Scenarios

For the first set of results, we compared the third-party authentication times for all nine scenarios, as can be seen in Fig. 4. It was evident that 802.1x-to-802.1x took the most time compared to the other scenarios. Most of the time spent in 802.1x authentication was during the EAP authentication with the Cisco switch. The switch had to enable the port, perform the necessary configurations associated with the device, and apply the ACLs. When we had 802.1x at the source side, we performed two 802.1x authentications on the target side. The first was the initial authentication where the supplicant was given restricted access to the network (i.e., all the connections are redirected to the captive portal), and the second occurred when the supplicant was given full access to the network. Therefore, in 802.1x-to-802.1x, there were three 802.1x authentications in total: two at the target network and one at the source network. PANA-to-PANA took the least time of all. We suspect that this was because PANA was implemented at Layer 3 and utilized IP fragmentation. On the other hand, 802.1x was implemented at Layer 2 and the certificates needed to be transferred through discrete EAP payloads which incurred a higher overhead compared to PANA (http://www.panasec.org/).

Latency Breakdown

For the second set of results, we divided the authentication time into multiple sections to see how much time was taken by the user, the target service provider, the source service provider, and the proxy. The main purpose of this was to find out if the major portion of authentication time was taken by the proxy or not. It can be seen from Fig. 4 that the proxy took 3.61–51.73 percent of total authentication time that ranged between 0.128–3.504s for the nine scenarios. The proxy took more time (12.39–51.73 percent) in scenarios involving OIDC, as most of the time was taken by the vRP and the vUser because of multiple re-directions. In scenarios involving the 802.1x protocol, the proxy took less time (3.61–12.39 percent) and the bottleneck was introduced by the source or target access point (802.1x switch).

Bottleneck Analysis

In order to further investigate the bottleneck, we broke all the scenarios down to the smallest unit to observe which particular message took the largest percentage of total authentication time. We identified the message in each scenario that took the longest time and consequently formed a bottleneck. We also identified the entity which was responsible for this message. The analysis showed that the bottleneck occurred at the access point where 802.1x was involved, as the Cisco switch created a single active path with the new device via the spanning tree protocol before sending the final EAP_success message. We measured that this process takes approximately 1 second. In conclusion, our proposed proxy did not result in a bottleneck in any of the scenarios.

Authentication Delay with vs. without Universal Fog Proxy

Figure 5 compares the authentication time between the proposed proxy and normal authentications of the three protocols without the proxy. In the absence of the proxy, the user would need to have two accounts, one with the source service provider and another with the target service provider, and the user would need to perform two authentications. The universal fog proxy-based authentication was compared with the concatenation of the other two involved protocols in all nine scenarios. It can be seen that the proxy took more time for authentication compared to the concatenation of the two protocols because the controller inside the proxy took time to initialize the modules as per require-

ment and to provide the necessary communication among the modules. The advantage of the proxy was that it eliminated the requirement of registering a new account with the target service provider.

Conclusions and Future Work

Fog federation will not only be useful for users but it will also be useful for service providers. In order to achieve this, third-party authentication issues need to be resolved in such a way that the different protocols of fog service providers can work together. To address this, we propose a standard compliant universal fog proxy which transparently coordinates these existing authentication protocols and provides third party authentication among them. We deployed a testbed and ran experiments for OIDC, 802.1x, and PANA which showed that the proxy successfully provides third-party authentication, taking 3.61–51.73 percent of total authentication time that ranged between 0.128–3.504s for nine scenarios. The scenarios involving 802.1x took considerably longer time, and the proxy's contributions to these scenarios were low (3.61–12.39 percent). In these cases, the bottleneck was introduced by a 802.1x switch due to the spanning tree protocol.

There are different possible directions in which to extend this work in the future:

- The proxy can be extended in terms of protocols by including other authentication protocols in a fog computing paradigm such as CoAP and MQTT for device authentication. This is really important as inclusion of more protocols will increase the deployment feasibility of the proxy.
- The proxy can be extended in terms of computing paradigms by including cloud computing and 3GPP edge computing. This is quite important for the cloud and 3GPP edge service providers as they will be able to form a vertical federation and support latency-constrained applications such as virtual reality, augmented reality, and self-driving vehicles.
- The capability of the proxy can also be extended by introducing an application state transfer. This will allow the proxy to support mobile users who move from place to place while accessing the same application.
- Finally, this work provides little insight into the security threats posed to the proxy; a complete security threat analysis of the proxy will be beneficial to identify further security risks that require mitigation.

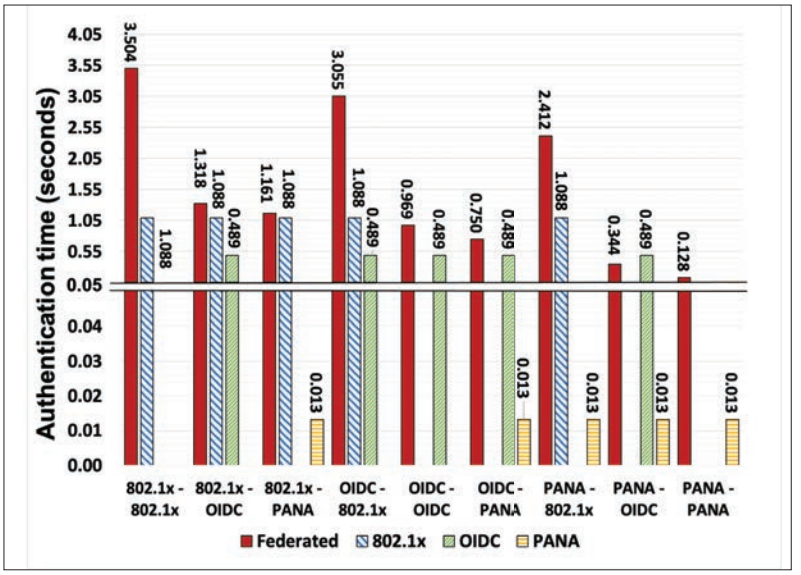

FIGURE 5. Authentication delay with vs. without universal fog proxy.

References

[1] OpenFog Consortium *et al.*, "Openfog Reference Architecture for Fog Computing," Architecture Working Group, 2017, pp. 1–162.
[2] Y. Li *et al.*, "Resource Allocation for Optimizing Energy Efficiency in NOMA-based Fog UAV Wireless Networks," *IEEE Network*, vol. 34, no. 2, 2020, pp. 158–63.
[3] Luiz Bittencourt *et al.*, "The Internet of Things, Fog and Cloud Continuum: Integration and Challenges," *Internet of Things*, vol. 3, 2018, pp. 134–55.
[4] J. Navas and M. Beltrán, "Understanding and Mitigating OpenID Connect Threats," *Computers & Security*, vol. 84, 2019, pp. 1–16.
[5] H. Shamseddine *et al.*, "A Novel Federated Fog Architecture Embedding Intelligent Formation," *IEEE Network*, vo. 35, no. 3, May/June 2021, pp. 198–204.
[6] M. Mukherjee *et al.*, "Revenue Maximization in Delay-Aware Computation Offloading among Service Providers with Fog Federation," *IEEE Commun. Letters*, vol. 24, no. 8, Aug. 2020, pp. 1799–1803.
[7] W. Zhang, Z. Zhang, and H.-C. Chao, "Cooperative Fog Computing for Dealing with Big Data in the Internet of Vehicles: Architecture and Hierarchical Resource Management," *IEEE Commun. Mag.*, vol. 55, no. 12, 2017, pp. 60–67.
[8] M. Cui, D. Han, and J. Wang, "An Efficient and Safe Road Condition Monitoring Authentication Scheme Based on Fog Computing," *IEEE Internet of Things J.*, vol. 6, no. 5, 2019, pp. 9076–84.
[9] P. Gope, "LAAP: Lightweight Anonymous Authentication Protocol for D2D-Aided Fog Computing Paradigm," *Computers & Security*, vol. 86, 2019, pp. 223–37.
[10] A. Yang *et al.*, "Delegating Authentication to Edge: A Decentralized Authentication Architecture for Vehicular Networks," *IEEE Trans. Intelligent Transportation Systems*, 2020.
[11] S. Alharbi *et al.*, "Secure the Internet of Things with Challenge Response Authentication in Fog Computing," *Proc. IEEE 36th Int'l. Performance Computing and Commun. Conf. (IPCCC)*, IEEE, 2017, pap. 1–2.
[12] I. F. Kilincer *et al.*, "An Effective Security Method Based on Combining 802.1x, DMZ and SSL-VPN for IoT Network Security," *Acta Infologica*, vol. 4, no. 2, 2020, pp. 65–76.
[13] A. M. Zarca *et al.*, "Managing AAA in NFV/SDN-Enabled IoT Scenarios," *Proc. Global Internet of Things Summit (GIoTS)*, IEEE, 2018, pp. 1–7.
[14] Z. A. Lux *et al.*, "Distributed-Ledger-based Authentication with Decentralized Identifiers and Verifiable Credentials," *Proc. 2nd Conf. Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, IEEE, 2020, pp. 71–78.
[15] R. Rios *et al.*, "From SMOG to Fog: A Security Perspective," *Proc. Second Int'l. Conf. Fog and Mobile Edge Computing (FMEC)*, IEEE, 2017, pp. 56–61.

Biographies

Asad Ali received his Master's degree in electrical engineering from the National University of Science & Technology, Pakistan. Currently, he is pursuing his Ph.D. degree from National Yang Ming Chiao Tung University, Taiwan. His research interests are network security, wireless communications, network design, and optimization.

Ali Utkan Şahin is currently pursuing his M.Sc. at EPFL, Switzerland. He received his B.Sc. from Koç University, Istanbul, Turkey, where he worked at the Distributed Systems and Reliable Networks (DISNET) Research Laboratory as an undergraduate research assistant. His research interests include network security, distributed systems, and cryptography.

Öznur Özkasap is a professor with the Department of Computer Engineering, Koç University. She received her Ph.D. in computer engineering from Ege University in 2000 and was a graduate research assistant with the Department of Computer Science, Cornell University, where she completed her Ph.D. dissertation. She is leading the DISNET Research Laboratory. She serves as an area editor for *IEEE Transactions on Parallel and Distributed Systems*; *Future Generation Computer Systems*, Elsevier; and *Cluster Computing*, Springer.She is a recipient of the Career Award of the Scientific and Technological Research Council of Turkey and the Informatics Association of Turkey 2019 Prof. Aydın Köksal Computer Engineering Science Award.

Ying-Dar Lin [F] is a Chair Professor of computer science at National Yang Ming Chiao Tung University, Taiwan. He received his Ph.D. in computer science from the University of California at Los Angeles in 1993. His research interests include network softwarization, cybersecurity, and wireless communications. His work on multihop cellular was the first along this line, and has been cited over 1000 times. He is an IEEE Distinguished Lecturer. He has served or is serving on the Editorial Boards of several IEEE journals and magazines, and was the Editor-in-Chief of *IEEE Communications Surveys & Tutorials* during 2016–2020.