Establishing a Session Database for SDN Using 802.1X and Multiple Authentication Resources

Frederik Hauser, Mark Schmidt, Michael Menth
Chair of Communication Networks, University of Tuebingen, Tuebingen, Germany

***Abstract*—Network control systems based on identities allow fine-grained access control for users. They require a network-wide session database containing information about active authenticated and authorized users. We propose an authentication and authorization (AA) module (AAM) as a controller application for software-defined networking to establish a network-wide session database and provide a prototypical implementation with OpenFlow. End systems issue authentication requests and the switch redirects them to the AAM. The AAM either relays them to a RADIUS server as in legacy 802.1X (pass-through mode) or processes them based on directly attached AA resources (authentication server mode). After successful authentication, the AAM authorizes the requesting user and maintains a network-wide session database of authenticated and authorized identities. As the AAM interfaces to end systems and AA resources through existing protocols, i.e., EAP and RADIUS, its use is compatible with current infrastructures. Through implementation as distributed network functions, the AAM can be scaled so that high rates of authentication requests can be supported.**

I. Introduction

Securing networks by introducing authentication and authorization is a major goal in network security. Especially large-scale networks with thousands of users require sophisticated network access control. Infrastructures for deploying authentication, authorization, and accounting (AAA) provide the technical foundations that have been used for many years. So far, authorization mostly supports coarse-granular access permissions or identity-based VLAN tagging. However, today's demands for secure network admission control go beyond that.

The rise of software-defined networking (SDN) led to extensive work on fine-granular network control systems that are based on user identities. Approaches like Ethane [1], Resonance [2] or Kinetics [3] allow network control with abstract identity-centric rules and stateful network control actions. This concept is called identity-based security [4] and has the potential to improve and simplify security solutions for large-scale networks with different security levels and user-specific access rights.

Network control systems based on user identities require network-wide session databases and reliable authentication mechanisms. The latter should be compatible to existing AAA infrastructures and end systems. Today, the majority of approaches for network access control applications in OpenFlow-based SDN suggest authentication using web frontends or MAC address mappings. Both approaches reveal shortcomings regarding compatibility, usability, and security that are pointed out in Section III. 802.1X is the most widely applied method for authentication and authorization (AA) in networks today. Some research works have adopted 802.1X for SDN but without leveraging the increased flexibility offered by SDN. In particular, there is no common best practice to perform AA using 802.1X for SDN.

We suggest an authentication and authorization module (AAM) as controller application to maintain a network-wide session database which interfaces with 802.1X to end systems and with other protocols, e.g., RADIUS, to authentication resources. The AAM can be easily used with existing end systems implementing 802.1X but does not suffer from the limitations of current 802.1X deployments. In particular, the AAM maintains a session database as needed for fine-granular network access control and can also leverage other authentication resources than RADIUS, e.g., user information in local databases.

The remainder of this paper is structured as follows. Section II shortly reviews key aspects of 802.1X. Section III discusses related work. In Section IV, we describe the AAM in detail. Section V describes its prototypical implementation and experimental evaluation giving a proof-of-concept. Section VI discusses how network function virtualization may be used to scale the AAM to large request rates. Finally, Section VII summarizes this work and draws conclusions.

II. Authentication and Authorization using 802.1X

In the following, we give an overview of the 802.1X architecture, describe the supporting protocols EAP and RADIUS, illustrate the operation of 802.1X, and discuss limitations of current deployments.

A. 802.1X Architecture

IEEE 802.1X [5]–[7] describes port-based network admission control in Ethernet networks. Although originally introduced for wireline networks, 802.1X is mainly known from wireless 802.11 networks today. A prominent application of 802.1X is eduroam [8], a federation of wireless university campus networks worldwide which allows participants to connect to the Internet in foreign institutions.

Figure 1 shows the architecture of 802.1X which adopts the components of the abstract AAA architecture in [9] with a different nomenclature. A network host is called supplicant system and contains a supplicant module. A LAN edge switch controlling the access of network hosts to the network is called

This work was supported by the bwNET100G+ project which is funded by the Ministry of Science, Research and the Arts Baden-Wuerttemberg (MWK). The authors alone are responsible for the content of this paper.

authenticator system and contains an authenticator module. The AAA server is called authentication server system and contains the authentication server. It is responsible for executing the actual authentication and provision of authorization information, and is triggered by the authenticator.

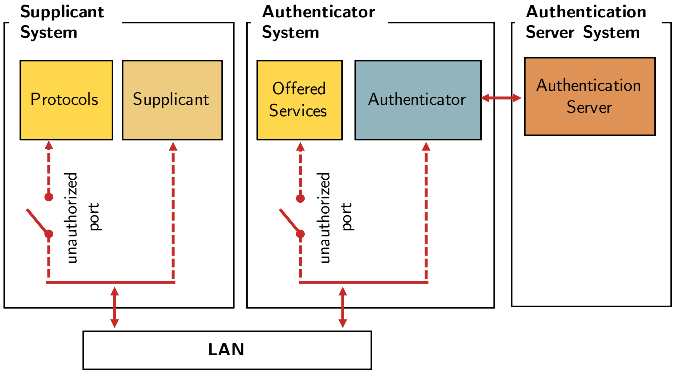


Fig. 1: Port-based authorization model of 802.1X according to [6].

The ports within the supplicant and authenticator system can be considered as abstract port entities. Without successful authentication and authorization, the supplicant system can reach only the authenticator module on an authenticator system. A 802.1X AA procedure is always initiated by the supplicant sending a start message to the authenticator. After successful authentication and authorization the unauthorized ports become authorized. Authorization in this context can be coarse- or fine-granular. The authentication server may inform the authenticator with a binary information whether access should be granted, or it may also provide a specific VLAN tag [10] for prospective user traffic.

Figure 2 shows that 802.1X encompasses both frontend and backend AA. Frontend authentication between the supplicant and authenticator modules is defined by the Extensible Authentication Protocol (EAP) [11]. Backend AA between the authenticator module and the authentication server can be performed by the Remote Authentication Dial In User Service (RADIUS) [12] or the Diameter protocol [13].

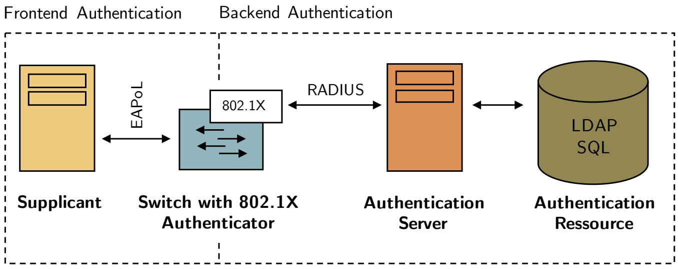


Fig. 2: Interaction of components in the classic 802.1X architecture.

B. EAP and RADIUS in 802.1X

In the following, we introduce EAP and RADIUS in the context of 802.1X by looking at the examplary AA process depicted in Figure 3. We solely focus on RADIUS because it is the most-widely used protocol for backend AA.

1) Initialization of AA in 802.1X: The supplicant module on the network client initiates AA by sending an EAPOL-Start message. EAPoL is an EAP-over-LAN encapsulation to transport EAP messages within Ethernet frames that was introduced with 802.1X. EAP facilitates communication between supplicant and authenticator, and it provides a fixed request and response scheme to exchange authentication data between supplicant and authentication server.

2) Identity-based AA in 802.1X: The authenticator requests the client's identity and forwards it to the RADIUS authentication server. RADIUS supports large domains that consist of a large number of hierarchically organized RADIUS servers. Each identity (e.g. an user) is associated with a domain and known by the RADIUS server of that domain. Therefore, the identity is the most relevant information for routing AA attempts within RADIUS infrastructures. This principle is used, e.g., in eduroam which allows users to leverage the wireless university campus network infrastructure on foreign campuses or special venues like IETF meetings and conferences.

3) Authentication in 802.1X: Authentication is performed between supplicant and authentication server. The authenticator decapsulates EAP packets from EAPoL frames and reencapsulates them in RADIUS frames and vice versa. The flexible message structure of EAP allows the use of different authentication procedures. Simple approaches carry plain-text identity information or simple MD5-hashed passwords, but more secure authentication procedures like IKEv2 for EAP [14], EAP Tunneled TLS [15], and EAP-TLS [16] are also supported. As the authenticator only relays EAP messages in pass-through manner, it does not need to implement any EAP type specifics.

4) Authorization in 802.1X: After successful authentication, the RADIUS server may return authorization data, e.g., a VLAN tag. The authenticator applies the authorization data on the particular physical port on the switch, e.g., it sets a VLAN tag. Afterwards, the authenticator confirms successful AA to the supplicant with an EAP-Success message.

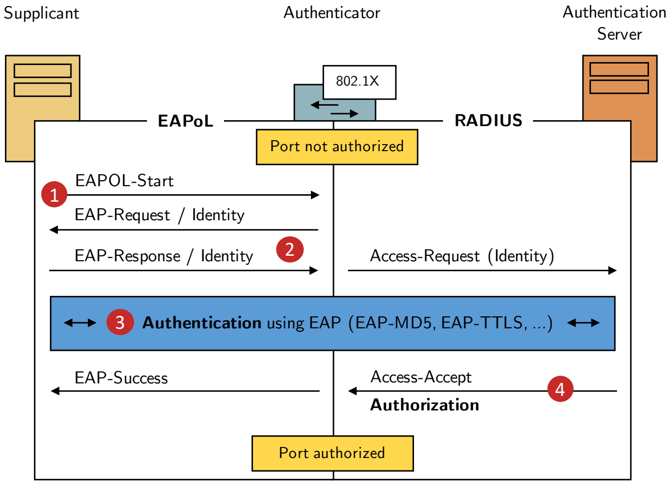


Fig. 3: Communication example of 802.1X based authentication and authorization (AA).

C. Limitations of 802.1X

In [17], client-side security concerns of 802.1X are pointed out. Here, we discuss limitations regarding flexibility with respect to the infrastructure side.

1) Dependence on RADIUS or Diameter: 802.1X requires a RADIUS or Diameter server for backend AA. In most cases, RADIUS servers use AA data stored on external resources like an Lightweight Directory Access Protocol (LDAP) server or an SQL database. The use of RADIUS is an advantage if user data needs to be accessed from a foreign domain, but for exclusively local applications this is unnecessary overhead because RADIUS is an additional service which requires configuration and administration effort. Direct interaction between an authenticator and an AA resource is more lightweight and may be used in parallel to other RADIUS resources. Another aspect is formal administration overhead. Official RADIUS infrastructures are managed by central computation centers and adding users is a major administration process which may be desirable to avoid for separate experimental infrastructure or student labs.

2) Change of Authorization: Backend authentication in 802.1X does not support unsolicited messages from an authentication server to an authenticator. As a consequence, changes in AA data, e.g., revocation of a user's permission to access the network, cannot be applied to existing sessions. Due to this limitation, dynamic authorization extensions [18] to RADIUS have been defined but they are are supported by only a few authenticator implementations.

3) Stateless Property of RADIUS: Today, an unlimited number of concurrent authorized network accesses may be initiated with 802.1X using the credentials of the same identity. This is due to the stateless property of the system, i.e., authenticator and authentication servers do not keep records of currently authorized user sessions. Due to this shortcoming, the *Simultaneous-Use* extension [19] was defined to limit the number of concurrent sessions and to introduce some session context on the RADIUS server. The *radutmp* module [20] for FreeRADIUS implements this extension but is hardly deployed. Active sessions are tracked by little standardized RADIUS accounting messages or through the use of SNMP, Finger, and telnet which is not a reliable solution to the problem.

III. Related Work

We give an overview of network control systems based on user identities and report state of the art for authentication in SDN.

A. Network Control Systems Based on User Identities

Ethane [1] introduced abstract rules based on user identities, host classes, and protocols for the definition of abstract rules for network access. Resonance [2] extends this rather stateless view by introducing state graphs for network hosts. Their states change in the network control system, e.g., when a network security scanner detects a malware infection on an authorized host. Kinetics [3] introduces a domain-specific language for the definition of fine-granular network control rules. In addition, it allows formal verification of these rules. Ravel [21] focuses on plain network rule representation in an SQL database and heavily uses view abstractions. All these approaches require a session database, but none of them addresses how to build and maintain it using common AA mechanisms.

B. Authentication and Authorization (AA) in SDN

MAC address mapping and web frontends are most widely used for AA in SDN. Moreover, some research prototypes adopt 802.1X in various forms to provide the same service as today, but they do not maintain a session database.

1) Identification and Authorization Using MAC Addresses: In this approach, MAC addresses are either used as identities or they are mapped to identities. The identities are used to fetch authorization data from an AA resource, e.g., a RADIUS or an LDAP server. In contrast to authentication, this process of identification does not verify the identity claimed by the network client. MAC addresses are unique but not confidential, especially network devices like printers often have a printed label revealing their MAC addresses. Moreover, MAC addresses are easy to eavesdrop. Even a novice attacker can spoof MAC addresses and, therefore, easily impersonate network hosts to obtain access to the network.

2) AA Using Web Frontends: When web frontends are used for AA and an unauthorized user sends traffic, the user is redirected to a web frontend to provide credentials, e.g., a user name and password, or a client certificate. This is problematic, because it requires a valid IP configuration prior to authorization. Moreover, an HTTPS-capable web browser is not available on all platforms like printers, document scanners, phones, or surveillance cameras. Besides, re-directions require web browser usage. If browsers are not used, like on pure administration workplaces, users have to recognize that network connectivity is currently limited due to missing authorization and manually visit the web frontend. Providing credentials in a web frontend is cumbersome, especially if users are required to complete frequent re-authentications.

The web frontend directly interfaces an AA resource, e.g., an LDAP server or an SQL database. Therefore, the web application needs to implement the actual AA procedure. First, it may compare the input from a user with a hashed password from the AA resource. Then, it may report the authorization data, possibly including a VLAN tag, to the SDN controller. The authors in [22] and [23] focus on compatibility with existing backend authentication infrastructures that are based on RADIUS. The web frontend acts as a RADIUS client performing the AA procedure with the help of a RADIUS server.

3) AA in OpenFlow-Based SDN Using 802.1X: Some research prototypes for OpenFlow-based SDN adopt 802.1X for AA. Most of them make use of hostapd [24], an open-source user-space implementation for an 802.1X authenticator.

The SDN controller *FAUCET* [25] forwards EAPOL frames to a user-space instance of hostapd [24] that authenticates and authorizes the network client. The frontend hostapd_cli interfaces hostapd and outputs information about all AA attempts in a log file. A script monitors the log file and reports successful authentication attempts to the SDN controller. As an alternative approach, *AuthFlow* [26] extends hostapd by introducing an SSL-based communication channel to directly signal successful AA attempts to the SDN controller. Finally, *FlowIdentity* [27] builds a wrapper for running an instance of hostapd within the SDN controller Trema.

Instead of authenticating and authorizing users or end systems, *FlowNAC* [28] uses 802.1X to build a fine-granular network access control system to authenticate different applications on a network host. To enable multiple authentication and authorization processes per host, FlowNAC introduces EAPOL-in-EAPOL encapsulation as an extension to legacy 802.1X. This deviation from the original standard requires changes on all 802.1X components (supplicant modules, authenticator modules, and authentication servers) to support the additions. In particular, the implementations of the 802.1X supplicant (wpa_supplicant) and authenticator (hostapd) were extended to support EAPoL-in-EAPoL encapsulations.

IV. The AAM Architecture

The AAM is an AA module which serves as application for an SDN controller. We describe how the AAM implements the 802.1X concept, how it leverages multiple authentication resources, and how it maintains a session database.

A. *Implementation of 802.1X in an SDN Context*

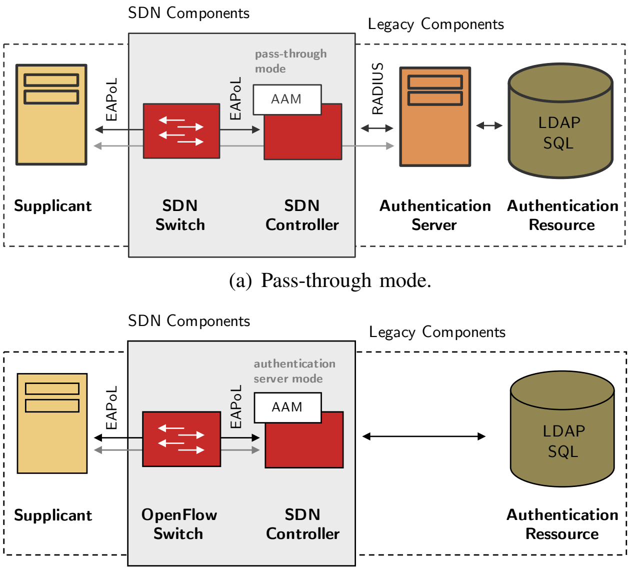
Figure 4(a) illustrates how 802.1X can be adopted for SDN. In legacy 802.1X infrastructures, the authenticator resides on the edge switch as shown in Figure 2. In SDN, we propose to implement the authenticator as a module, the AAM, on the SDN controller.

A network host initiates AA by sending an EAPOL-Start message as depicted in Figure 3. The SDN edge switch is instructed to forward that message to the controller which redirects it to the AAM. The AAM adopts the functionality of a legacy 802.1X authenticator, i.e., it relays communication between the supplicant and the authentication server as depicted in Figure 3. Therefore, it does not need to implement specific EAP types, e.g., EAP-TLS, EAP-TTLS, or EAP-PEAP. We designate this first mode of the AAM as *pass-through mode*.

After successful authentication, the RADIUS server may return authorization data, e.g., a binary permission to access a particular network or a VLAN tag. This authorization data is transmitted in the RADIUS Access-Accept message as shown in Figure 3. The AAM implements mechanisms to translate authorization data from RADIUS Access-Accept messages into corresponding SDN rules to be applied on the SDN edge switch.

B. *Integration of Alternative AA Resources*

As mentioned before, legacy 802.1X authenticators interoperate only with RADIUS or Diameter and so does the above described concept for SDN. To allow for more flexibility, alternative AA resources should be supported. Figure 4(b) shows our proposal for that integration. The AAM essentially acts both as authenticator and authentication server. That means, the AAM must implement all EAP type specifics and perform the desired authentication procedure using authentication data from the alternative resources. We designate this second mode of the AAM as *authentication server mode*.



(a) Pass-through mode.

(b) Authentication server mode.

Fig. 4: Interaction of components in the SDN-enabled 802.1X architecture

A simple example for an alternative AA is a CSV file containing user names, hashed passwords, and authorization data stored on the SDN controller accessible by the AAM. A more complex example is an LDAP or an SQL database with appropriate information that is remotely accessibly by the AAM but without a RADIUS server in between. The AA resources also provide authorization data that the AAM should apply after successful authentication in the form of appropriate flow rules on the edge switch.

If multiple AA resources exist, the AAM must choose the appropriate one. We propose two selection options for that problem: port-based resource selection and identity-based resource selection.

We first explain port-based resource selection. When an edge switch redirects the initial EAP request of the supplicant to the AAM, it includes context information, in particular the physical port and identifier of the SDN edge switch over which the packet was received. The AAM may use this port to determine the authentication resource to be used. A potential use case is a student lab where particular Ethernet ports may be authenticated and authorized using a simple AA resource instead of the RADIUS architecture for the overall campus network.

The alternative identity-based AA resource selection is limited to specific EAP methods. All EAP methods have an identical initialization routine where the authenticator requests the supplicant's identity to be transmitted in plain text. As an improvement for providing confidentiality about the identity on intermediate nodes, most EAP types support the concept of outer and inner identities. The outer identity, e.g., anonymous@foo.bar, is transmitted in the initialization in plain text and only serves as forwarding hint within a distributed

RADIUS infrastructure to find the RADIUS server in the home organization of the user. The actual identity of the user, e.g., john.smith@foo.bar is transmitted within an encrypted tunnel between supplicant and authentication server. The authors in [29] leverage multiple RADIUS infrastructures and use the outer identity to select the one to be used. We follow a similar approach and use the outer identity to determine the appropriate AA resource.

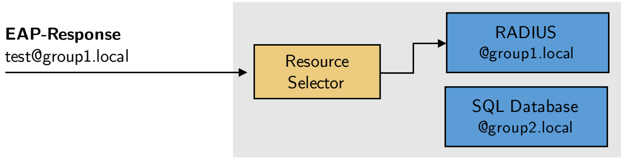


Fig. 5: The AAM selects the appropriate AA resource using the outer identity in the EAP response from the supplicant.

C. Network-Wide Session Database

We propose that the AAM maintains a network-wide session database which may be used by an identity-based network control system. This approach allows maintenance of user state while leaving AA servers and resources simple and stateless as originally desired.

The session database contains information about all authenticated and authorized identities, in particular all of their (simultaneously) active sessions. After a network host has successfully passed AA, the AAM adds a corresponding session entry to the session database. Conversely, the AAM uses port-down events from the SDN edge switch to remove sessions. As this feature is not available on all switches, sessions may be closed without notification. Therefore, we propose that the AAM may periodically issue EAP re-authentication requests to keep sessions and authorization alive.

Figure 6 illustrates exemplary contents of a session database for two identities. Each of their session entries contains information regarding the time of the last successful AA, the AA method used, the physical port of the network host and authorization information received by the AA resource.

```
{test@group1.local : { max_sessions : 2,
    sessions : (
        { aaa_time : Mo 13 Jun 2016 14:16:26 CEST,
          aaa_method : Radius(ip=10.0.20.100, meth=EAP-MD5),
          phys_port : OF-Switch(ip=10.0.20.222, port=1),
          assigned_vlans : (10)},
        { aaa_time : Mo 13 Jun 2016 14:18:31 CEST,
          aaa_method : Radius(ip=10.0.20.100, meth=EAP-MD5),
          phys_port : OF-Switch(ip=10.0.20.222, port=2),
          assigned_vlans : (10)},
         )},
 test@group2.local : { max_sessions : 1,
    sessions : (
        { aaa_time : Mo 13 Jun 2016 12:18:31 CEST,
          aaa_method : SqlDb(ip=10.0.20.101, meth=EAP-MD5),
          phys_port : OF-Switch(ip=10.0.20.222, port=3),
          assigned_vlans : (20)},
         )}
}
```

Fig. 6: Exemplary contents of a session database.

External applications can interact with the AAM by using communication techniques like REST interfaces. For example, after detection of uncommon behavior, a network security scanner may issue a de-authorization of a network host. The AAM triggers the necessary actions on the session database and SDN edge switches. Vice versa, the AAM could request the network security scanner to perform a scan on a network host that passed AA in a network for the first time.

V. Prototypical Implementation & Functional Validation

We explain the implementation of the AAM, describe our semi-virtualized testbed, and validate the AAM approach through experimentation.

A. Prototypical Implementation of the AAM

We implemented the AAM as a proof-of-concept for the Ryu SDN controller framework [30]. Its source code package contains several bootstrap applications that we used as starting point. We chose the *SimpleSwitch* application which controls an OpenFlow-based SDN switch in such a way that it acts like a simple Ethernet switch.

In contrast to most approaches presented in section III, we did not reuse the open-source 802.1X authenticator hostapd but provided a native Python-based implementation. We chose *dpkt* for network packet generation and parsing. We provided own implementations for EAP and RADIUS as they are not contained in *dpkt*.

For AAM's authentication server mode, the AAM requires implementations for EAP types (e.g., EAP-TTLS or EAP-PEAP) and interfaces for AA resources (e.g. LDAP database or SQL server). We designed an object-oriented class hierarchy and heavily used the concept of mixins to minimize intersections. For simplicity reasons, we implemented only EAP-MD5 as a single EAP type.

The initialization routine of the SDN controller installs on all connected SDN switches two proactive rules for each physical port that is marked as controlled by the AAM. The first rule forwards all EAPoL frames from network clients to the controller which relays them to the AAM. EAPoL frames are identified by the Ethernet frame's EtherType value 0x888E. The second rule drops all other packets at the switch port. This implements the behavior of the unauthorized state prior to successful AA.

The AAM also translates authorization data into flow rules and applies them to the corresponding port on the SDN edge switch. For example, simple permission of an identity just removes the flow rule for packet drop. More complex authorization data may result in flow rules setting a desired VLAN tag or allowing only a set of predefined destinations.

B. Testbed

We used the logical experimentation setup depicted in Figure 7 for the validation of the AAM. We experimented with a hardware- and a software-based OpenFlow-capable switch. The testbed is entirely virtualized except for the hardware switch. In the following, we describe the virtualization methodology and the OpenFlow switches in more detail.

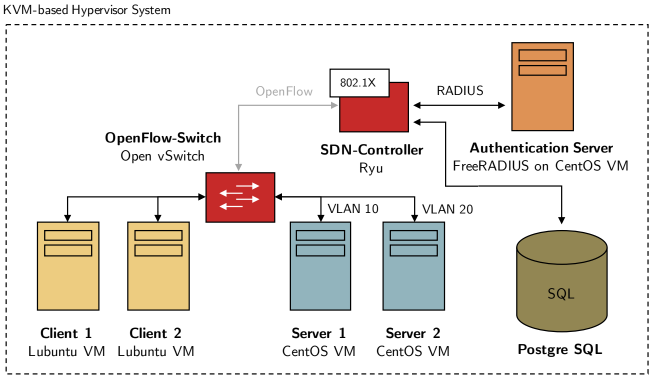


Fig. 7: Logical experimentation setup.

We used QEMU-based virtual machines with KVM acceleration managed by libvirt. As shown in Figure 7, we used Lubuntu 14.04 for user network hosts with a wpa_supplicant. As server network host, we used CentOS 7, and FreeRADIUS as backend authentication server.

We first experimented with HP Enterprise 2920 OpenFlow hardware switches. They failed handling EAPoL packets with firmware version 16.01.0006, i.e., flow rules installed to forward EAPoL frames to the controller were not effective, the frames were simply dropped. We consider this an incompatibility issue caused by the hybrid mode of the switch. Besides pure OpenFlow operation, the switch can be used as legacy L3/L4 switch with CLI and web frontend configuration. In legacy mode, the switch includes an 802.1X authenticator. As a hardware-based alternative, we used the SDN prototyping platform Zodiac FX with firmware version 0.66 that showed the expected behavior. As a software-based replacement, we used Open vSwitch [31] in version 2.4.0.

C. Validation Scenario

We used FreeRADIUS server and Postgre SQL as AA resources. The RADIUS server contained authentication data for user test@group1.local along with the VLAN tag 10 as authorization data. Authentication data for the user test@group2.local along with the VLAN tag 20 were stored in the SQL database. The AA resource was chosen on the outer identity provided in the EAP-Response message as depicted in Figure 3.

We performed various tests to validate the AAM's behavior. We tested the ability to use multiple AA resources. The 802.1X supplicant of Client 1 was alternately configured with the identity managed by the RADIUS server and the identity managed by the SQL database as AA resource. Depending on the identity, Client 1 got access to different VLANs, i.e., only Server 1 or Server 2 was reachable by an ICMP ping. We tested the session database, in particular its ability to limit the number of concurrent sessions. To that end, Client 1 and Client 2 were both authenticated with the same identity. We first set the number of concurrent sessions to be unlimited – both clients could be authorized. We then limited the number of concurrent sessions to 1 and verified that only a single session could be established.

VI. Scalability of the AAM

The AAM handles authentication requests, possibly initiates re-authentication requests, and performs authentication processes and cryptographic operations in authentication server mode. This may lead to high load in large networks and overload a central SDN controller.

We suggest to deploy the AAM as network function. It can be easily scaled to many instances because AA processes between multiple network hosts and authenticators are independent of each other and can be parallelized. Multiple AAM instances operate on the same session database that may also be distributed. The SDN controller installs flow rules for EAPoL frames on the edge switches to directly forward such frames to a particular AAM instance. This offloads the SDN controller and protects it from unauthorized network traffic.

VII. Conclusion

In this work we proposed to adopt 802.1X for authentication and authorization (AA) in SDN in such a way that multiple AA resources can be used and that a session database can be maintained. The former is useful for the integration of temporary and experimental local accounts, the latter for fine-grained network control systems which were a driver for early SDN research.

We implemented a controller application for OpenFlow-based SDN, the AA module (AAM). It uses EAP for frontend authentication so that it is compatible with existing end systems. In pass-through mode, the AAM leverages RADIUS or Diameter for backend authentication, which is easy to implement. In authentication server mode, the AAM acts as authentication server, i.e., it implements complex EAP types and checks AA data in LDAP servers, SQL databases, or local files. Independently of the specific mode, the AAM updates a session database with admitted sessions whenever a new session is authorized or whenever the controller recognizes a session teardown.

We implemented the AAM for the Ryu controller and conducted experiments in a semi-virtualized OpenFlow-controlled network including hardware and software switches. We validated the functionality of the proposed concept and observed that some switches refuse to forward EAPoL frames to their controller which may be problematic for the deployment of the AAM without appropriate patches. As the AAM performs time-consuming operations in authentication server mode, we suggested an organization of the AAM as a distributed network function to relieve the controller from potentially heavy load and to scale the AAM and the session database to very large networks.

VIII. Acknoledgements

The authors would like to thank Stefan Winter for fruitful discussions.

References

[1] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking Control of the Enterprise," in *ACM SIGCOMM*, Kyoto, Japan, Aug. 2007.

[2] A. Nayak, A. Reimers, N. Feamster, and R. Clark, "Resonance: Dynamic Access Control for Enterprise Networks," in *Workshop: Research on Enterprise Networking (WREN)*, Barcelona, Spain, Aug. 2009.

[3] H. Kim, J. Reich, A. Gupta, M. Shahbaz, N. Feamster, and R. Clark, “Kinetic: Verifiable Dynamic Network Control,” in *USENIX Syposium on Networked Systems Design & Implementation (NSDI)*, 2015.

[4] Cyberoam. Cyberoam’s Layer 8 Technology. [Online]. Available: https://www.cyberoam.com/layer8.html

[5] IEEE, “802.1X-2001 IEEE Standard for Local and Metropolitan Area Networks - Port-Based Network Access Control,” 2001.

[6] ——, “802.1X-2004 IEEE Standard for Local and Metropolitan Area Networks - Port-Based Network Access Control,” 2004.

[7] ——, “802.1X-2010 IEEE Standard for Local and Metropolitan Area Networks - Port-Based Network Access Control,” 2010.

[8] Eduroam, “Eduroam - About.” [Online]. Available: https://www.eduroam.org/index.php?p=about

[9] C. de Laat, G. Gross, L. Gommans, J. Vollbrecht, and D. Spence, “Generic AAA Architecture,” RFC 2903 (Experimental), aug 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2903.txt

[10] P. Congdon, M. Sanchez, and B. Aboba, “RADIUS Attributes for Virtual LAN and Priority Support,” RFC 4675 (Proposed Standard), sep 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4675.txt

[11] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz, “Extensible Authentication Protocol (EAP),” RFC 3748 (Proposed Standard), 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3748.txt

[12] C. Rigney, S. Willens, A. Rubens, and W. Simpson, “Remote Authentication Dial In User Service (RADIUS),” RFC 2865 (Draft Standard), jun 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2865.txt

[13] V. Fajardo, J. Arkko, J. Loughney, and G. Zorn, “Diameter Base Protocol,” RFC 6733 (Proposed Standard), 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6733.txt

[14] H. Tschofenig, D. Kroeselberg, A. Pashalidis, Y. Ohba, and F. Bersani, “The Extensible Authentication Protocol-Internet Key Exchange Protocol version 2 (EAP-IKEv2) Method,” RFC 5106 (Experimental), feb 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5106.txt

[15] P. Funk and S. Blake-Wilson, “Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0),” RFC 5281 (Informational), aug 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5281.txt

[16] D. Simon, B. Aboba, and R. Hurst, “The EAP-TLS Authentication Protocol,” RFC 5216 (Proposed Standard), 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5216.txt

[17] S. Brenza, A. Pawlowski, and C. Pöpper, “A practical investigation of identity theft vulnerabilities in eduroam,” in *ACM Conference on Security and Privacy in Wireless and Mobile Networks (ACM WiSec)*, New York, NY, USA, Jun. 2015.

[18] M. Chiba, G. Dommety, M. Eklund, D. Mitton, and B. Aboba, “Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS),” RFC 5176 (Informational), jan 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5176.txt

[19] FreeRADIUS, “FreeRADIUS Manual - Simultaneous-Use Attribute.” [Online]. Available: ftp://ftp.gnu.org/old-gnu/Manuals/radius/html_node/radius_177.html#SEC180

[20] Raddb, “raddb 3.0.10 Documentation - rlm_radutmp.” [Online]. Available: http://networkradius.com/doc/3.0.10/raddb/mods-available/radutmp.html

[21] A. Wang, X. Mei, J. Croft, M. Caesar, and B. Godfrey, “Ravel: A database-defined network,” in *ACM Symposium on SDN Research (SOSR)*, Santa Clara, CA, USA, Mar. 2016.

[22] V. Dangovas and F. Kuliesius, “Sdn-driven authentication and access control system,” in *The International Conference on Digital Information, Networking, and Wireless Communications (DINWC)*. Society of Digital Information and Wireless Communication, 2014, p. 20.

[23] F. Kuliesius and V. Dangovas, “Sdn enhanced campus network authentication and access control system,” in *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, July 2016, pp. 894–899.

[24] J. Malinen, “hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator.” [Online]. Available: https://w1.fi/hostapd/

[25] “FAUCET SDN: 802.1x authentication on FAUCET (NFV offload of authentication).” [Online]. Available: https://faucet-sdn.blogspot.de/2016/07/8021x-authentication-on-faucet-nfv.html

[26] D. M. Ferrazani Mattos and O. C. M. B. Duarte, “Authflow: authentication and access control mechanism for software defined networking,” *Annals of Telecommunications*, pp. 1–9, 2016. [Online]. Available: http://dx.doi.org/10.1007/s12243-016-0505-z

[27] S. T. Yakasai and C. G. Guy, “Flowidentity: Software-defined network access control,” in *Network Function Virtualization and Software Defined Network (NFV-SDN), 2015 IEEE Conference on*, Nov 2015, pp. 115–120.

[28] J. Matias, J. Garay, A. Mendiola, N. Toledo, and E. Jacob, “Flownac: Flow-based network access control,” in *European Workshop on Software Defined Networks (EWSDN)*, Budapest, Hungary, Sep. 2014.

[29] Z. Cao, J. Fitschen, and P. Papadimitriou, “Freesurf: Application-centric wireless access with sdn,” in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM ’15. New York, NY, USA: ACM, 2015, pp. 357–358. [Online]. Available: http://doi.acm.org/10.1145/2785956.2790000

[30] Ryu, “Ryu SDN Framework.” [Online]. Available: https://osrg.github.io/ryu/

[31] O. vSwitch, “Open vSwitch - Production Quality, Multilayer Open Virtual Switch.” [Online]. Available: http://openvswitch.org/