RadWi: Experimenting with WPA-2 enterprise and RADIUS

***Abstract*—In this document we describe how to build a simple billing platform for WiFi using open-source tools. In our architecture we rely heavily on WPA-2 Enterprise stack of protocols and use our custom python-based implementation of RADIUS for authentication, authorization, and accounting purposes. To implement the billing part, we created custom software. The billing works as follows: (i) users receive or purchase vouchers which specify the amount of traffic included, and, of course, username and password; (ii) prior to distribution of vouchers the operator generates previously mentioned information (using custom tool) and stores it in the database; (iii) the users can connect to any Radwi WiFi network at any time and can freely use the Internet as long as balance (amount of traffic unused) is positive; and finally, (iv) at certain time quanta the Radius server updates the database and reduces the balance of the client by amount of traffic which was used. We foresee that this business model is viable and can be used in many public places. Although the system can be monetized, our primary goal is experimentation and hands on experience in building secure Enterprise WiFi networks.**

I. Introduction

Today WPA-2 Enterpise stack of protocols is defacto standard for building secure WiFi networks. For that matter in this document we describe how to build commercial product using these standards. Our primary goal, however, is hands on experience.

In this pace, this document has the following structure. First, we give definitions to identification, authentication, authorization, and accounting. Next, we discuss the extensible authentication framework (EAP). We then turn our attention to RADIUS protocol and discuss interaction of EAP-TTLS protocol. Next, we switch to discussion of wireless encryption protocols used in modern WiFi networks. We then move on to discussion of overall architecture of the system. For that matter we discuss the interactions with RADIUS server, how billing is organized and finally scratch the surface regarding monetization issues. We then describe what kind of hardware and software we have used to build the prototype solution. Next, to get an idea of how the system will perform in real-life settings we sat down and measured the performance of basic operations such as encryption, decryption and key derivation. Here we will report such metrics as duration of protocol execution and energy consumed during the protocol run. We conclude the document with the discussion of how the system can be deployed in public places. For example, here we touch such aspects as voucher production and distribution.

II. Background

Modern wireless enterprise grade networks rely heavily on many standards and protocols. And in this chapter we will review the the building blocks of such networks. We start the discussion with description of RADIUS protocol, and then move on to the discussion of EAP framework. We conclude the section with the discussion of wireless encryption and key management standards.

A. Triple A: Authentication, Authorization, Accounting

While authentication and authorization are important for securing the networks, accounting is important for such purposes as billing. Every secure network should employ triple A solutions. In this pace, to get a clear picture, in the next few paragraphs we give definitions for these terms.

We start with the identification process because it will be later on used in the definition of authentication. In this manner, identification is a process of verifying a unique identifier of a device or a person. In other words, identification is the process of checking who the person is or what kind of device is connecting to a network. Authentication, on the other hand, is verification of whether the person or device is indeed the true holder of the identifier. In network security protocols, this process typically involves verification of whether the person or device possesses some kind of secret information. This is typically done with so-called challenge-response protocols. For example, the server (which also knows the secret) can ask the device to encrypt or hash a random value and send it back to the server. Once the response is received, the server computes similar digest and verifies the results. If the two digests match, then the authentication succeeds, meaning that the device proved that it is the genuine holder of the previously presented identity.

Authorization is a process of checking access rights to a resource, such as document, network or computing asset. In computer networks, for example, the access server can check whether an authenticated user has rights to access, or should we say utilize, network bandwidth.

And finally, accounting relates to the process of bookkeeping of what kind of resources the user was accessing and for how long the user was utilizing these resources.

B. EAP Authentication framework

Extensible Authentication Protocol (EAP) is an authentication framework mostly used in wireless networks and point-to-point connections. Basically, EAP is a framework for providing the transport and usage of material and parameters generated by EAP methods [1]. There are over 40 methods currently defined. We will describe some of these methods in the proceeding paragraphs.

Nimble out-of-band authentication for EAP (EAP-NOOB) is a generic bootstrapping solution for the devices

which do not have preconfigured authentication credentials and which are not yet registered on any server. It is typically used for Internet of Things (IoT) which do not have information about network, owner and server [**?**]. Authentication of this method relies on the existence of out-of-band (OOB) channels. Examples of OOB channels are QR codes, NFC tags, audio, light, etc. In this manner, the device usually performs Elliptic Curve Diffie-Hellman (ECDH) key exchange with the server using in-band EAP channel. The user than confirms the exchange by transmitting OOB message to server or end-device. For example, the user can transmit the confirmation message (which can be for example fingerprint of derived key) over infrared channel after which the device will complete the pairing.

Lightweight Extensible Authentication Protocol (LEAP). Is a proprietary EAP method developed by Cisco. LEAP uses Microsoft Challenge-handshake authentication protocol (MS-CHAP) for authentication. LEAP allows to derive dynamic Wireless Equivalent Privacy (WEP) keys upon successful authentication. The drawback of this protocol is that credentials are not strongly protected and can be easily compromised []. For that reason protected EAP (EAP-PEAP) was developed as a secure alternative to LEAP.

EAP-MD5. EAP-MD5 is specified in RFC 2284. EAP-MD5 operates much like CHAP protocol. In this authentication method, after link establishment, RADIUS server sends a challenge (in a form of a nonce) to the end-user and expects response to the challenge in a form of keyed MD5 fingerprint for previously sent challenge. Since EAP-MD5 is vulnerable to dictionary attack, and there is no additional protection (such as IPSec or TLS tunnel), this method is considered insecure. EAP-MD5 also does not provide methods for authenticating the server, so man-in-the-middle attacks are also possible.

EAP Transport Layer Security (EAP-TLS). EAP-TLS is specified in RFC 5216. It was the original standard for wireless networks and relies heavily on Transport Layer Security (TLS) protocol [**?**]. The authentication method requires that both the client and the server present signed certificates during TLS base exchange for mutual authentication. After the successful authentication the Access Server (RADIUS) generates pairwise master key to be used between wireless access point and end-user (wireless station).

EAP Tunneled Transport Layer Security (EAP-TTLS). The authentication method is defined in RFC 5281. EAP-TTLS is an extension to TLS protocol. EAP-TTLS provides mutual authentication. The difference between EAP-TTLS and EAP-TLS is that in EAP-TTLS the client does not need to present the certificate to server. Instead, after establishing a secure channel by means of TLS (during which the RADIUS server authenticates itself using TTP signed certificate), the client authenticates itself to the server by means of tunneled EAP, CHAP (challenge-handshake protocol), MS-CHAP (Microsoft challenge-handshake protocol), or PAP (password authentication protocol).

Protected EAP (PEAP). This method is similar to EAP-TTLS in that both protocols prior to client authentication establish TLS tunnel. During tunnel establishment the client authenticates the server using server side certificate signed by trusted third party (TTP). After the tunnel establishment, the client is authenticated using existing EAP methods (such as LEAP). Upon successful authentication of the client, RADIUS issues pairwise master key which will be shared by access station and access point.

EAP Internet Key Exchange Version 2 (EAP-IKEv2). EAP-IKEv2 is described in RFC 5106. EAP-IKEv2 is the EAP method based on Internet Key Exchange protocol []. Much like EAP-TLS, EAP-IKEv2 provide mutual authentication between peer (end-user) and RADIUS server. However, EAP-IKEv2 can use different methods for authentication. First, authentication can be done with asymmetric keys signed by trusted third party (TTP). Second, authentication can be carried out with low-entropy shared secrets, such as passwords. And finally, authentication can be achieved with high-entropy symmetric keys.

To give the feeling of how EAP packet is organized in Figure 3 we show its structure.

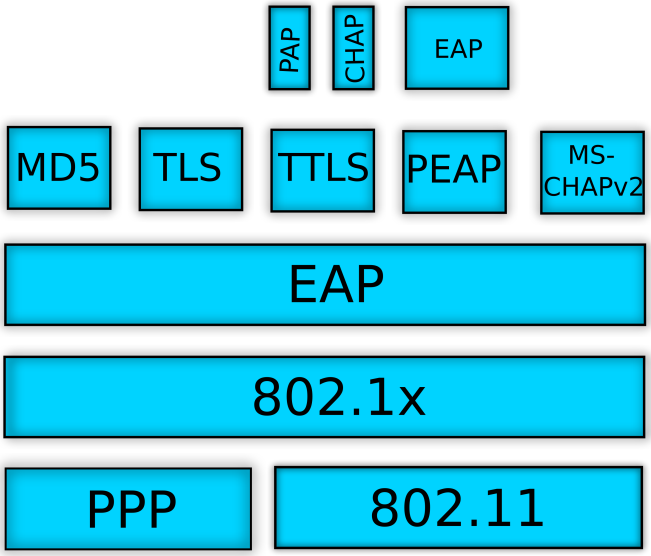

Fig. 1: EAP packet structure[1]

C. IEEE 802.1x

IEEE 802.1x is a port-based network access control and provides authentication mechanism to devices wishing to attach to LAN or WLAN []. Overall, IEEE 802.1x defines encapsulation of the EAP over IEEE 802.3 and 802.11 networks, which is known as EAP over LAN or simple EAPOL. According to this standard whenever IEEE 802.3 Ethernet, or in case of wireless network IEEE 802.11, user attaches to network, the port remains closed and the network switch or wireless access point will forward only packets between the end-user and AAA server such as RADIUS. When wired switch or wireless access point receives EAPOL frame it will wrap those inside RADIUS packets and send to the RADIUS server. On the other hand, when RADIUS responds, the EAP packets will be stripped from RADIUS packet and forwarded to an end-user as EAPOL frame. The IEEE 802.1x port will be trusted, or authorized and, thus, all traffic will be allowed only,

[1] Adapted from https://www.ixbt.com/comm/prac-wpa-eap.shtml

after reception of Access Accept message from the RADIUS server.

D. RADIUS protocol

Remote Authentication Dial-In User Service (RADIUS) is a networking protocol, operating over User Datagram Protocol (UDP) on port 1812 (accounting is performed on 1813 UDP port). The protocol provides centralized authentication, authorization, and accounting (AAA or Triple A) management for users who connect to and use network services.

RADIUS was designed to serve the purpose of allowing a NAS to forward a dial-up users request and its credentials to a backend server. RADIUS was originally designed to accommodate PAP (password authentication protocol) and CHAP (challenge handshake authentication protocol). The specification of RADIUS protocol consists of several RFCs: RFC 2058, RFC 2138, RFC 2865 and RFC 2866.

In RADIUS typically exist three parties who participate in authentication, authorization and accounting: an end user, wishing to gain access to a network, Network Access Server (NAS), which usually acts as RADIUS client and Access Server (AS), which checks credentials, makes admission decisions and can also optionally return configuration information to a client.

RADIUS message set is rather simple and consist only of eight messages, of which only first four are specified in the base specification [4]. The six most commonly used messages are:

- **Access request**. This message is generated by the NAS towards the RADIUS server to forward the request from or on behalf of a user.
- **Access challenge**. This message is sent by RADIUS server towards the client to question the user about the knowledge of a secret.
- **Access accept**. This message is sent by the RADIUS server to NAS to indicate the successful completion of request.
- **Access reject**. This message is sent by the RADIUS server to NAS to indicate the rejection of a request.
- **Accounting request**. This message is sent from NAS to RADIUS server to convey the accounting information.
- **Accounting response**. This message is sent by the RADIUS server to the NAS to acknowledge that the accounting information sent previously by the client has been received. The message also indicates the result of the performed accounting function by the server.

In Figure 2 we show the basic RADIUS packet structure. The first byte is the message type code. Next byte is identifier, which is used for matching requests with responses. Next two bytes used to store the length of the RADIUS packet including the code, identifier, length, authenticator and optional attributes. Authenticator field is 16 bytes long and is used for security purposes. The authenticator field (in request packets) is merely a random string of 16 bytes and is used to authenticate replies from the RADIUS server. In that manner, in response packets authenticator field contains MD5 hash over code field, identifier, length, request authenticator field from the access-request packet, and response attributes, followed by the shared secret. We should note that passwords transmitted in RADIUS packets are also secured with so-called password hiding mechanism, which is a keyed version of MD5 hash. Shared secret followed by the request authenticator field is put through a one-way MD5 hash to create a 16 bytes digest value which is XORed with the password entered by the user, and the XORed result is placed in the user-password attribute in the access-request packet (the reverse operation is trivial if a party known the shared secret). More detailed definition of RADIUS packet formats can be found in RFC 2865.

The attribute-value pairs (AVP) carry data in both requests and responses. The AVP consists of the attribute type, attribute length and attribute value. Almost every hardware manufacturer creates its own list of attributes. In our Enterprise WLAN setup, among many other attributes, we will use Mikrotik-Total-Limit and Mikrotik-Rate-Limit attributes, so that the NAS can rate-limit the connections as well as limit the amount of traffic received and transmitted.

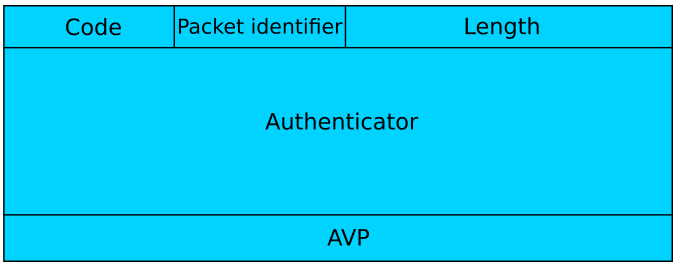

Fig. 2: Radius packet structure

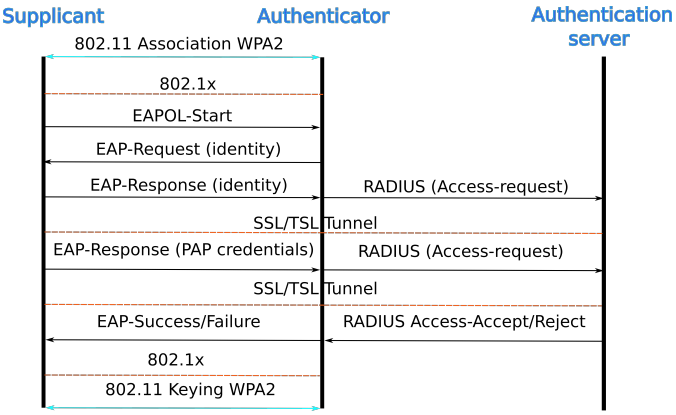

Fig. 3: EAP usage in 802.11 WPA2 Enterprise setting[2]

In addition to PAP and CHAP, RADIUS can also be used to carry EAP messages. In Figure [**?**] we demonstrate the interaction of end-user with the NAS and AS over EAP-TTLS.

[2]Adapted from https://www.eduroam.us/docs/tech_overview

E. Wireless encryption algorithms

In this section we will discuss the solutions which allow users to send messages in an encrypted format over wireless links. More specifically, the focus will be on wireless IEEE 802.11 networks.

WEP, or Wired Equivalent Privacy, is an old standard which defines how the keying material is derived, what encryption and message authentication algorithms to be used. For encryption, RC4 stream cipher is being used, which is known to be insecure []. For integrity checks this standard employs CRC-32 algorithm.

WPA

WPA-2

WPA-2 Enterprise uses EAP framework for authentication and key management. In essence, WPA-2 Enterprise uses the same algorithms for encryption, authentication and key management as WPA-2. The major difference is how the master secrets are derived: In WPA-2 Enterprise wireless station and authentication server use TLS protocol to negotiate master secret, later on the authentication server delivers the keying material to access point using RADIUS protocol.

III. Architecture

In Figure 4 we present the high-level architecture of the demo network and some details about interaction of the billing system with the Radius server and frontend application. The presented architecture does not however describe the interaction with the banks and other payment systems.

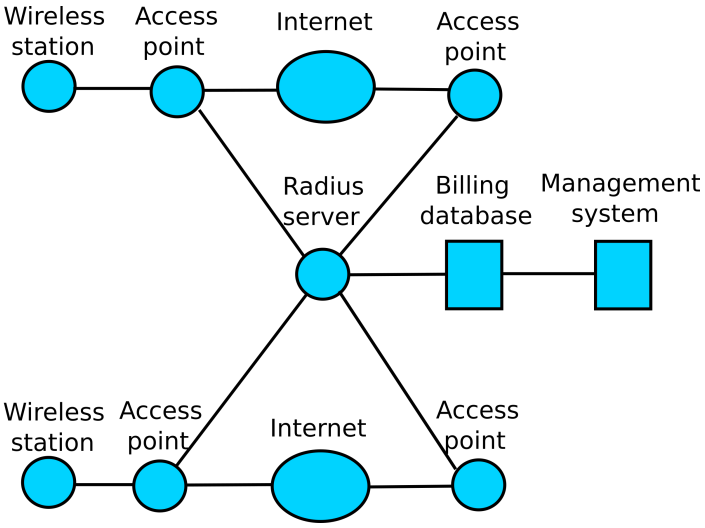

Fig. 4: High-level architecture of the Intranet and billing system

IV. Implementation

We have opted to use our own custom implementation of a RADIUS server. Thus, using the following RFCs - RFC 2865 (RADIUS), RFC 5281 (EAP TTLS Authentication Protocol), RFC 2866 (RADIUS accounting), RFC 2869 (RADIUS Extensions), RFC 5246 (The Transport Layer Security (TLS) Protocol Version 1.2) - we have implemented basic RADIUS server [3]. Our implementation comprises roughly $6K$ LOC and has minimal support for TLS server version 1.2 (we implemented support for only two cipher suits (i) TLS RSA with AES 256 in CBC mode for encryption and SHA 256 for hashing; (ii) TLS RSA with AES 128 in CBC mode for encryption and SHA 256 for hashing).

V. Configuration

Our first step was related to configuration of MikroTik router. First, we have created bridge interface and included WLAN1 and Ethernet ports in it. Then, we have configured wireless adapter: (i) we have configured security profile (WPA-2 Enterprise) and (ii) we have configured WLAN1 interface to operate in wireless access point mode. Finally, we have configured the RADIUS client on MikroTik with the following command.

```
radius add
protocol=udp
secret="highentropysecret"
address=192.168.0.105
service=wireless
accounting-port=1813
disabled=no
```

Beacause decryption of pre-master secret takes more than one second (on the Raspberry PI), we had to increase the RADIUS timeout to two seconds, otherwise the RADIUS client was closing socket too early and kernel was sending `ICMP destination port unreachable` during the handshake.

We then configured our custom implementation of RADIUS server. Basically the only configuration that was required was related to generation of x509 certificate. We have accomplished this using `openssl` tool:

```
openssl req -newkey rsa:8192
-nodes -keyout key.pem
-x509 -days 365
-out certificate.pem
```

We have noticed that if the user does not specify the Certificate Authority's (CA) certificate, no alert is generated on Android phone signalizing that the certificate is self-signed.

VI. Hardware and software

For hardware we have used the following. For Wireless Access Point we have decided to use relatively cheap Mikrotik RB951Ui-2HnD [2]. We have made such choice primarily because of the Mikrotik RouterOS [3], which among many other features supports RADIUS authentication and accounting. Moreover, this Linux-based operating system has flexible settings for WPA-2 Enterprise, routing, traffic shaping and many more. For the server side hardware we have used cheap Raspberry PI microcomputer which is powerful enough to support RADIUS, MySQL and web-based management tools. Of course, this is only good enough for prototyping and demos.

[3]The source code is available online at https://github.com/dmitriykuptsov/radwi/tree/master/radius

Operation ↓ Metric →	Duration (s)	Current (A)	Energy (J)
AES encryption (256 bits key)	$8.6 \cdot 10^{-5}$	0.16	$7.0 \cdot 10^{-5}$
AES decryption (256 bits key)	$7.1 \cdot 10^{-5}$	0.16	$5.5 \cdot 10^{-5}$
SHA 256 hash computation	$1.3 \cdot 10^{-4}$	0.16	$1.1 \cdot 10^{-4}$
RSA encryption (8192 bits modulus)	$4.2 \cdot 10^{-2}$	0.16	$3.4 \cdot 10^{-2}$
RSA decryption (8192 bits modulus)	1.1	0.16	$9.0 \cdot 10^{-1}$
RSA signature computation (8192 bits modulus)	1.1	0.16	$9.0 \cdot 10^{-1}$
RSA signature verification (8192 bits modulus)	$4.3 \cdot 10^{-2}$	0.16	$3.5 \cdot 10^{-2}$

VII. Experimental evaluation

We divided all experiments into two parts. In the first phase, we have performed series of microbanchmarkings (AES encryption, AES decryption, RSA signature generation, RSA signature verification, packet processing, and database requests). Here we have measured such characteristics as duration and energy consumption. In the second phase, we have tested the overall execution of the system.

Fig. 5: Demo setup

For all primitive operations, such as encryption, decryption, hash calculation, etc., we have measured two things: (i) duration of the operation (in seconds); (ii) electrical current draw (in Ampers). Using duration and current measurements we have calculated average energy consumption as $E = UIt$. We present the results in Table [**?**]. For each experiment we have perfomed 100 iterations and present the mean values as results. For all operations we have used messages of size 1000 bytes (for AES cipher the size of the message was 1024 bytes). The operating voltage of the microcomputer was 5V.

To measure the current we have used the setup shown in Figure [**?**]

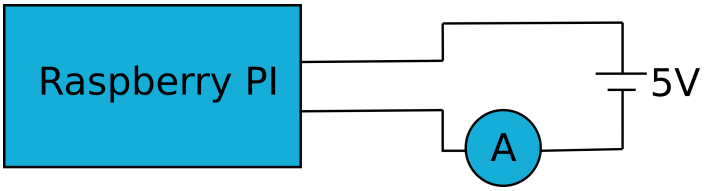

Fig. 6: Current measurement apparatus

VIII. Discussions

IX. Conclusions

References

[1] Extensible Authentication Protocol. https://en.wikipedia.org/wiki/Extensible_Authentication_Protocol.

[2] Mikrotik RB951Ui-2HnD. https://routerboard.uz/products/wifirouters/RB951Ui-2nD.html.

[3] Mikrotik RouterOS. https://i.mt.lv/pdf/ros/ros_3_0_reference_manual.pdf.

[4] M. Nakhjiri and M. Nakhjiri. *AAA and Network Security for Mobile Access: Radius, Diameter, EAP, PKI and IP Mobility*. John Wiley and Sons, 2005.