Design and Implementation of WIRE1x

Yu-Ping Wang[1] **Yi-Wen Liu[2]** **Jyh-Cheng Chen[1,2]**

[1]Institute of Communications Engineering

[2]Department of Computer Science

National Tsing Hua University

Hsinchu, Taiwan

{ichiro, timl, jcchen}@wire.cs.nthu.edu.tw

Abstract

This paper presents the design and implementation of WIRE1x. The WIRE1x is an open source implementation of IEEE 802.1x client (supplicant) developed by the Wireless Internet Research & Engineering (WIRE) Laboratory[1]. The IEEE 802.1x standard defines a port-based network access control to authenticate and authorize devices interconnected by various IEEE 802 LANs. IEEE 802.11i also incorporates 802.1x as its authentication solution for 802.11 wireless LANs. The motivation for the development of WIRE1x is that many users are eager for a free software of 802.1x client to work with various versions of MS Windows. The WIRE1x has been practically used on the wireless LANs deployed at the National Tsing Hua University. This paper illustrates all components of WIRE1x exhaustively to let readers to comprehend the source code easily.

Keywords: 802.1x, 802.11i, Authentication, Wireless LANs, Security

1. INTRODUCTION

Wireless local area network (WLAN) is more and more prevailing in recent years. It however has been widely reported that security has been a weakness of the current 802.11 standards. The Task Group I of IEEE P802.11 Working Group is defining 802.11i [8] to enhance security in current 802.11 standards. 802.11i includes two main developments: Wi-Fi Protected Access (WPA) and Robust Security Network (RSN). It incorporates 802.1x [9] as the authentication solution for 802.11 wireless LANs. The IEEE 802.1x standard is a port-based network access control to authenticate and authorize devices interconnected by various IEEE 802 LANs. 802.11i will be ratified in late 2003. It is expected to play a critical role in improving the overall security of current and future WLANs.

The 802.1x standard has been well-defined. Currently, many manufactures of 802.11 Access Point (AP) also support 802.1x. The 802.1x-capable APs have been deployed in many universities, organizations, and companies. To be authenticated by using 802.1x, end users also need to be 802.1x-capable. Unless 802.1x is embedded in the OS, users generally will need to install a 802.1x client in order to access to the network. Open1x [14], an open-source implementation of 802.1x, supports MacOS X, FreeBSD, OpenBSD, and Linux. Many users however are using MS Windows. They need a 802.1x client software to access to the 802.1x-based LANs. We therefore develop the *WIRE1x* to support various versions of MS Windows. As the name suggested, WIRE1x is an open-source implementation of IEEE 802.1x client (supplicant[2]) developed by the Wireless Internet Research & Engineering (WIRE) Laboratory.

Currently, WIRE1x supports Windows XP (without service pack and with service pack 1), Windows 2000, and Windows 98. Although it only provides EAP MD5-Challenge now, one of the major objectives is to support more advanced authentication methods. WIRE1x works with FreeRADIUS [6]. The implementation is based on Open1x, and is developed by using MS Visual C++. It utilizes libraries of WinPcap [17] and Libnet [12]. Both source code and executable code can be downloaded freely from *http://wire.cs.nthu.edu.tw/wire1x/*. This paper presents all components of WIRE1x exhaustively. The aim of this paper is to share our experience of implementing 802.1x client (supplicant). By reading this paper, one should easily comprehend the source code of WIRE1x.

The rest of this paper is organized as follows.

[1]http://wire.cs.nthu.edu.tw/

[2]Supplicant is a terminology defined in 802.1x which will be described in Section 2.

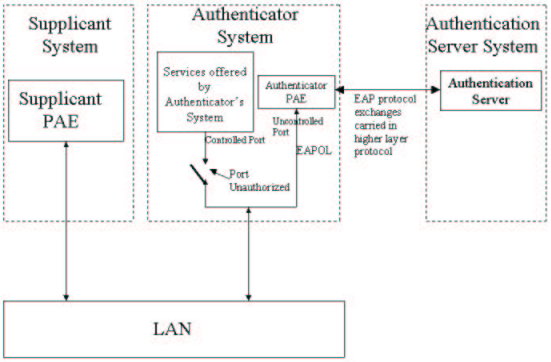

Figure 1 AUTHENTICATOR, SUPPLICANT, AND AUTHENTICATION SERVER

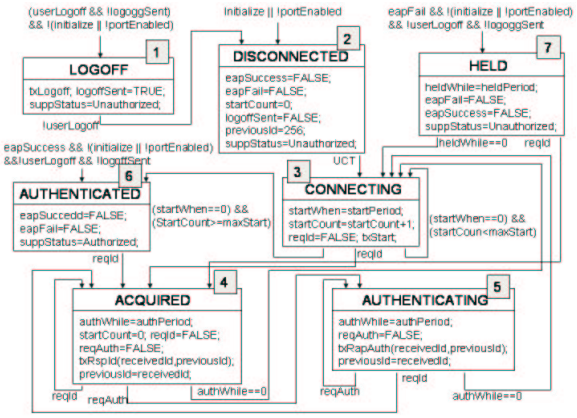

Figure 3 SUPPLICANT PAE STATE MACHINE

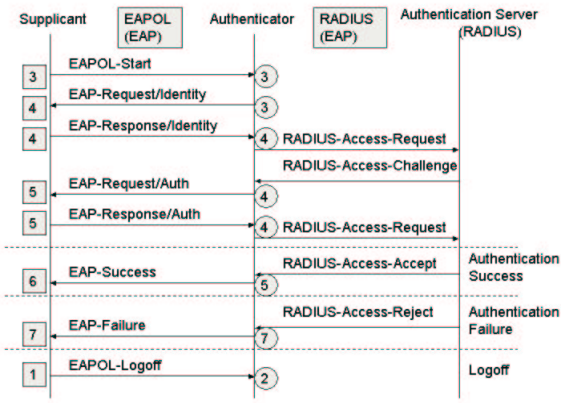

Figure 2 A TYPICAL 802.1X MESSAGE FLOW

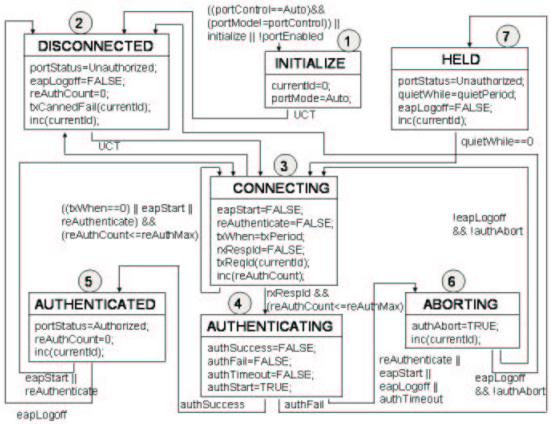

Figure 4 AUTHENTICATOR PAE STATE MACHINE

Section 2 provides a brief overview of IEEE 802.1x. Section 3 describes other available implementations of 802.1x. Section 4 discusses the design and implementation of WIRE1x in details. Section 5 presents the real-world applications of WIRE1x. Section 6 summarizes the paper and points out future work.

2. OVERVIEW OF IEEE 802.1X

The IEEE 802.1x defines a mechanism for port-based network access control. It is based upon the EAP (Extensible Authentication Protocol) [3] to provide compatible authentication and authorization mechanisms for devices interconnected by IEEE 802 LANs. As depicted in Fig. 1, there are three main components in the 802.1x authentication system. In a WLAN, supplicant is a MN (mobile node). AP usually represents an authenticator. AAA (Authentication, Authorization, and Accounting) server such as RADIUS server [15] is an authentication server. The port in 802.11 WLANs represents the association between a MN and an AP. Both the supplicant and the authenticator have a PAE (Port Access Entity) that operates the algorithms and protocols associated with the authentication mechanisms. In Fig. 1, the authenticator's controlled port is in the unauthorized state, which will block all traffic except 802.1x messages. The authenticator PAE will switch to the authorized state after the MN is authenticated successfully.

Based upon EAP, the 802.1x standard provides a number of authentication mechanisms including MD5 (Message Digest 5) [16], TLS (Transport Layer Security) [1], TTLS (Tunneled TLS) [7], PEAP (Protected Extensible Authentication Protocol) [2], LEAP (Lightweight Extensible Authentication Protocol) [13], and others. It also defines EAPOL (EAP over LANs)

Table 1 802.1X SUPPLICANTS

Vendor	OS Supported	EAP Supported
Meetinghouse	MS Windows 98/ME/NT/2000/XP	MD5,TLS,TTLS
Funk	MS Windows 98/ME/2000/XP	MD5,LEAP
Free1x.ORG	Linux/BSD	TLS
Microsoft	MS Windows 2000/XP	MD5,TLS
Cisco	MS Windows/MacOS/Linux	MD5,LEAP

to encapsulate EAP messages between the supplicant and the authenticator. The authenticator PAE relays all EAP messages between the supplicant and the authentication server.

Fig. 2 shows a typical 802.1x message exchange. The associated PAE state transitions in supplicant and authenticator are specified as well. The supplicant PAE state machine is shown in Fig. 3. The authenticator PAE state machine is shown in Fig. 4. In Fig. 2, the digits in rectangles refer to the supplicant PAE state in Fig. 3, and digits in circles refer to the authenticator PAE state in Fig. 4. In Fig. 2, RADIUS is served as the authentication server. This does not limit the use of other AAA server such as Diameter [4] as the authentication server. Detailed discussion of Fig. 2 can be found in [5], [11].

3. RELATED WORK

This section discusses related implementations of 802.1x in the marketplace.

Table 1 [10] lists most of available 802.1x supplicants. The supported operating systems and EAP authentications are also specified. Excepted Free1x, most of them are commercial products and not free to use. The source code are not open to everyone either. In addition, none of them support most of the authentication mechanisms including MD5, TLS, TTLS, PEAP, and LEAP. Table 2 [10] shows the supported EAP authentications in various authentication servers. As indicated, not all authentication mechanisms are supported in each authentication server. Thus, an end user would not be able to use a single 802.1x client to access to all WLANs if there is no common authentication mechanism supported in the supplicant and authentication servers. This will limit the roaming capability significantly.

The supplicant software is indispensable to IEEE 802.1x standard. We observe that the success of 802.1x will greatly depend on end users. We believe a free 802.1x supplicant software which works with various versions of MS Windows and supports most of EAP authentication methods will boost the deployment of 802.1x, and thus 802.11i. Therefore, we are developing WIRE1x and hope that most users could access to WLANs with a more secure way.

Table 2 AUTHENTICATION SERVERS AND SUPPORTED EAP

Vendor	EAP Supported
Funk Software	MD5,LEAP,TLS,TTLS
Interlink Networks	MD5,LEAP
Hewlett-Packard	MD5,LEAP
Microsoft	MD5,TLS
FreeRadius.ORG	MD5,TLS
Cisco Systems	MD5,LEAP,TLS

4. WIRE1X

The software architecture of WIRE1x can roughly be divided into three components as illustrated in Fig. 5. The *supplicant PAE state machine* follows the specification defined in the IEEE 802.1x for a supplicant. The state machine is also depicted in Fig. 3. As the name suggested, the *EAP authentication methods* supports various authentication mechanisms in EAP. The *WinPcap and Libnet* are two free source libraries which are responsible for capturing and writing packets from and to the data link layer. Next sections first discuss the three components in details. A typical operation of WIRE1x is then discussed.

4.1 Supplicant PAE State Machine

The supplicant PAE state machine is the core of any implementation of 802.1x supplicant. It specifies the behavior of the supplicant and interacts with the authenticator. Again, the supplicant PAE state machine is specified in Fig. 3 and the source code can be

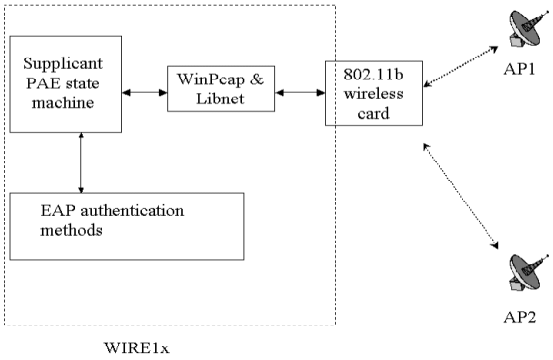

Figure 5 SOFTWARE ARCHITECTURE OF WIRE1X

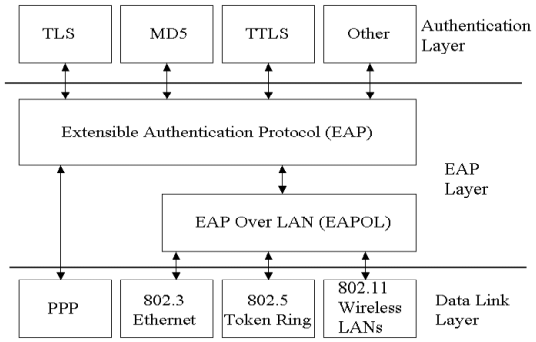

Figure 6 BLOCK DIAGRAM OF EAP AND ASSOCIATED LAYERS

downloaded from *http://wire.cs.nthu.edu.tw/wire1x/*. In WIRE1x, roughly speaking, it is implemented in four files: dot1x_globals.cpp, eap.cpp, eapol.cpp, and os_generic.cpp. All variables of state machine is defined in dot1x_globals.h. Additionally, EAP code field and type field specified in RFC2284 is defined in eap.h. The eap.cpp is responsible for building the response frame and decoding the EAP packet. Moreover, EAPOL header and Ethernet header are defined in eapol.h. The eapol.cpp is responsible for starting EAPOL process, performing necessary PAE state action, transiting to proper state, decoding the EAPOL packet, and transmitting EAPOL frame. Furthermore, os_frame_funcs.h is comprised of get_frame(), send_frame(), more_frames() and so on. In os_generic.cpp, get_frame() employs pcap_dispatch() to capture EAP frames. The send_frame() employs libnet_write_link() to send EAP frames.

4.2 EAP Authentication Methods

WIRE1x is expected to be versatile in authentication mechanisms. It will be implemented to support renown authentication methods such as LEAP and PEAP in addition to MD5, TLS, and TTLS. Fig. 6 shows that all of these authentication methods are based on EAP. Any new authentication mechanism can be added to WIRE1x easily.

Based on Table 1 and Table 2, one can see that MD5 is most popular in the marketplace because it is ease to use. Users simply type in their usernames and passwords to be authenticated. Comparatively, MD5 is more vulnerable than others. TLS has been proposed for mutual authentication in which users can also authenticate the AP to prevent forged APs. The supplicant and authentication server need to have a valid certificate when using EAP TLS. Additionally, TTLS uses TLS to authenticate the server with a certificate and establish an encryption tunnel. The client does not necessarily know the certificate. LEAP is developed by Cisco Systems, Inc. and is implemented in most Cisco products. PEAP is proposed by Microsoft, Cisco, RSA Security to securely transport authentication data including passwords over 802.11 WLANs. The purpose of PEAP is to protect the EAP negotiation within a TLS channel. Because the initial EAP identity Request/Response exchange is sent in clear, attackers snooping on the conversation can collect user identities for subsequent attacks. By initially negotiation a TLS channel, PEAP provides identity protection. PEAP is considered more advanced than LEAP because it supports secure mutual authentication. Generally, PEAP and TTLS are identical. Both of them use TLS to set up an end-to-end tunnel to transfer the user's credentials without having to use a certificate on the client. Both of them are recognized as the main streams in the future.

Currently WIRE1x supports MD5 only. TLS is implemented and is under testing. All of the authentication methods mentioned above will be implemented in WIRE1x subsequently.

4.3 WinPcap and Libnet

As described earlier, WinPcap and Libnet are used to capture/write packets from/to data link layer to associate with an AP.

WinPcap is used for packet capture and network

analysis in Win32 platform. It includes a kernel-level packet filter, a low-level dynamic link library (packet.dll), and a high-level and system-independent library (wpcap.dll which is based on libpcap version 0.6.2). It is in charge of the following tasks:

1) The pcap_findalldevs() in wire1xDlg.cpp prints the lists of interfaces. Therefore an user can select a proper interface to be authenticated.
2) The pcap_dispatch() in os_generic.cpp captures packets from AP.
3) The setup_pcap() in os_generic.cpp adjusts parameters such as filter. Filter can make the supplicant receive EAP frames only.
4) The pcap_close() in os_generic.cpp shuts down WinPcap.

Libnet is a generic networking API that provides access to several protocols. It is used only for libnet_write_link() in os_generic.cpp to write packets to AP.

4.4 Operation

This section uses the message exchange depicted in Fig. 2 and the associated state machine in Fig. 3 to demonstrate a typical authentication procedure of of WIRE1x. The example uses MD5-Challenge as the authentication method.

1) User opens and selects the device wishing to be authenticated by pcap_findalldevs(). MN starts to associate with AP. Both of MN and AP then will transition to the CONNECTING state.
2) MN sends an EAPOL-start frame by libnet_write_link() to the AP to initialize the authentication process.
3) When the AP receives EAPOL-Start, it will reply with EAP-Request/Identity to obtain the MN's identity. When the MN captures the EAP frame by pcap_dispatch(), the EAP frame is parsed by eap_decode_packet() and eapol_decode_packet() located in eap.cpp and eapol.cpp, respectively. Moreover, according to the result that determined by eap_decode_packet() and eapol_decode_packet(), the supplicant PAE state machine transits to ACQUIRED state if the request is received successfully.

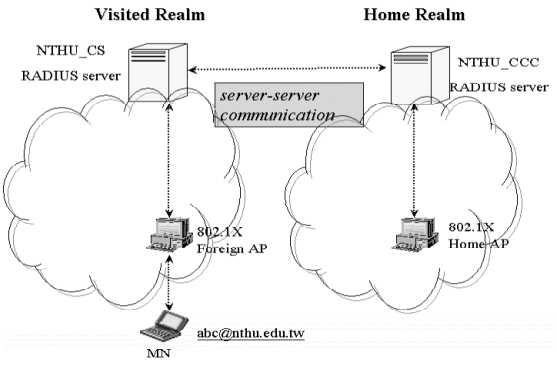

Figure 7 AUTHENTICATION AT VISITED NETWORK

4) The MN sends back EAP-Response/Identity containing MN's identity to the authenticator. Subsequently, the authenticator and authentication server will perform necessary message exchanges.
5) When the MN receives EAP-Request/Auth which contains RADIUS-Access-Challenge, the supplicant PAE state transits to the AUTHENTICATING state. The MN replies an EAP-Response/Auth to the authenticator in which RADIUS-Access-Request is encapsulated.
6) Based on the result of EAP MD5-Challenge authentication method which is coded in eapmd5.cpp and md5.cpp, the RADIUS server decides whether to authorize the user or not. If authorized, the supplicant captures the EAP-Success and the supplicant PAE state machine transits to the AUTHENTICATED state. Otherwise, the supplicant captures the EAP-Failure and transits to the HELD state.

5. REAL-WORLD APPLICATIONS

The WIRE1x has been practically used at the National Tsing Hua University (NTHU). At NTHU, each department/institute is responsible for the deployment of networking facilities in its own building(s). The university has no authority over the areas owned by the department/institute. The Computer & Communication Center (CCC) operated by the university is responsible for the networking facilities in public areas on campus. Thus, following standards will be essential for roaming and integration even inside campus.

Currently both CCC and the Computer Science (CS) department at NTHU have deployed WLANs by using 802.1x and RADIUS to authenticate users. To achieve roaming, both of their RADIUS servers could connected together as shown in Fig 7. Assuming a user *abc* has an account at CCC. Once the user roams to the WLANs covered by the CS department, the CS RADIUS server can authenticate the user by relaying the authentication messages back to the CCC RADIUS. The CS RADIUS server acts as a proxy client to the CCC RADIUS server. With only one account at CCC, the user still could roam to other WLANs.

At NTHU, most users are using MS Windows. Since WIRE1x was released on June 18, 2003, the website of WIRE1x has been visited around 3000 times. There have been more than 320 downloads of source code and 350 downloads of executable code of WIRE1x.

6. Summary and Future Work

This paper presents the motivation of WIRE1x. The implementation are discussed in details. By reading this paper, one should be able to examine the source code of WIRE1x in addition to use it. Currently, WIRE1x supports only some specific wireless cards and provides only EAP MD5-Challenge. However, we will continue to improve it. We hope WIRE1x will not only be used at NTHU, but will also play a crucial role for the integration of WLANs among different universities, companies, and organizations.

Acknowledgments

We thank other members of the WIRE Lab, especially Chin-Hsing Lin, Wen-Ting Wu, and Jui-Hung Yeh, for helping the development of WIRE1x. We also thank the support of the Computer and Communication Center, National Tsing Hua University, and the Computer Center, Department of Computer Science, National Tsing Hua University.

This work was sponsored in part by MOE Program for Promoting Academic Excellent of Universities under the grant number 89-E-FA04-1-4, National Science Council under the grant numbers 91-2219-E-007-023 and 92-2213-E-007-019, and Industrial Technology Research Institute under the contracts of T1-92019-3 and 2F-92050-4.

References

[1] Aboba, B., and Simon, D. PPP EAP TLS authentication protocol. IETF RFC 2716, Oct. 1999.

[2] Andersson, H., Josefsson, S., Zorn, G., Simon, D., and Palekar, A. Protected EAP protocol (PEAP). draft-josefsson-pppext-eap-tls-eap-02.txt, Feb. 2002.

[3] Blunk, L., and Vollbrecht, J. PPP extensible authentication protocol (EAP). IETF RFC 2284, Mar. 1998.

[4] Calhoun, P. R., Arkko, J., Guttman, E., Zorn, G., and Loughney, J. Diameter base protocol. draft-ietf-aaa-diameter-12.txt, Aug. 2002.

[5] Chen, J.-C., Jiang, M.-C., and Liu, Y.-W. Wireless LAN security and IEEE 802.11i. *IEEE Wireless Communications* (2004). To appear.

[6] freeRADIUS. http://www.freeradius.org/.

[7] Funk, P., and Blake-Wilson, S. EAP tunneled TLS authentication protocol (EAP-TTLS). draft-ietf-pppext-eap-tls-02.txt, Feb. 2002.

[8] IEEE Std 802.11i/D2.0. Draft supplement to standard for telecommunications and information exchange between systems - part 11: wireless MAC and PHY specifications: specification for enhanced security, Mar. 2002.

[9] IEEE Std 802.1X-2001. IEEE standard for local and metropolitan area networks, port-based network access control, Oct. 2001.

[10] The InteropNet Labs (iLabs). http://www.ilabs.interop.net/WLAN_Sec_2002_Spring/ni_2002_las_about_us.pdf.

[11] Jiang, M.-C., Liu, Y.-W., and Chen, J.-C. Integration of Mobile IP with WLAN Security. In *Proc. of 2002 Taiwan Area Network Conference* (Hsinchu, Taiwan, Oct. 2002), pp. 950–955.

[12] Libnet. http://libnet.sourceforge.net/.

[13] Lightweight extensible authentication protocol - LEAP. http://www.cisco.com/.

[14] Open1x. http://www.open1x.org/.

[15] Rigney, C., Willens, S., Rubens, A., and Simpson, W. Remote authentication dial in user service (RADIUS). IETF RFC 2865, June 2000.

[16] Rivest, R. The MD5 message-digest algorithm. IETF RFC 1321, Apr. 1992.

[17] WinPcap. http://winpcap.polito.it/.