Extensible authentication protocol

PB/INFO/065 (11/05)

The Extensible Authentication Protocol (EAP) is best considered as a framework for transporting authentication protocols, rather than as an authentication protocol itself. EAP can be used for authenticating dial-up and VPN connections, and also Local Area Network (LAN) ports in conjunction with IEEE 802.1X.

The Basics

In EAP [1], the party demanding proof of authentication is called the **authenticator** and the party being authenticated is called the **supplicant**. EAP defines four types of packet: request, response, success and failure. Request packets are issued by the authenticator and they solicit a response packet from the supplicant. Any number of request-response exchanges may be used to complete the authentication. If the authentication is successful, a success packet is sent to the supplicant; if not, a failure packet is sent.

The basic EAP packet format is simple. A type field indicates the type of packet, such as a response or a request. An Identifier field is used to match requests and responses. Response and request packets have two further fields. The first, confusingly called ‘type’, indicates the type of data being transported (such as an authentication protocol), and the second, type-data, consists of that data. Note that EAP method is synonymous with type, and both are used frequently.

The EAP specification defines three ‘basic’ authentication EAP types (MD5-Challenge, OTP and GTC) and three non-authentication types (Identity, Nak,* and Notification: * *Nak* is short for 'Negative Acknowledgement Code' but is conventionally spelt and capitalised in this manner). The three ‘basic’ authentication types are not considered secure for typical use, particularly in wireless environments. Consequently other types, which will be discussed later, should be used. The Identity type is used by the authenticator to request the user name claimed by the supplicant, and is typically the first packet transmitted. The Nak type is used by the peer to indicate that a type proposed by the authenticator is unacceptable (for example, the authenticator has proposed an authentication protocol that is unsupported by the peer, or policy forbids its use). If this happens then the authenticator may choose to try another, thereby allowing supplicant and authenticator to negotiate a mutually acceptable authentication protocol. The Notification type, which is rarely used, returns a message that must be displayed to the user.

Finally, EAP permits pass-through authentication. This allows the authenticator to forward all responses, using the RADIUS protocol, to a remote EAP server (in practice, most RADIUS servers also understand EAP). This server assumes the role of the authenticator for the remainder of the EAP session, and attempts to authenticate the supplicant against a user

database server. Pass-through authentication, therefore, permits centralised management of authentication against large numbers of authenticators. Another advantage is that the authenticator does not need to support the type negotiated by the peer and the EAP server. An example EAP exchange is shown in Figure 1 below. The peer refuses OTP authentication, but agrees to MD5-Challenge and is authenticated.

[1]

Figure 1: an example authentication

The following diagram shows an extract of the debugging output from a supplicant (wpa_supplicant [2]) during the above scenario (note the use of ‘method’ which as previously mentioned is a synonym of ‘type’):

```
EAP: Received EAP-Request method=1 id=2                          (step 2)
    EAP: EAP-Request Identity data – hexdump_ascii(len=38):
    00 6e 65 74 77 6f 72 6b 69 64 3d 65 64 75 72 6f     _networkid=eduro
    61 6d 2c 6e 61 73 69 64 3d 43 43 57 33 2c 70 6f     am,nasid=CCW3,po
    72 74 69 64 3d 30                                   rtid=0
EAP: using real identity – hexdump_ascii(len=9):                 (step 4)
    74 65 73 74 2d 75 73 65 72                          test-user
EAP: Received EAP-Request method=21 id=3                         (step 6)
EAP: Building EAP-Nak (requested type 21 not allowed)            (step 7)
EAP: Received EAP-Request method=4 id=4                          (step 8)
EAP-MD5: generating Challenge Response                           (step 9)
EAP: Received EAP-Success
```

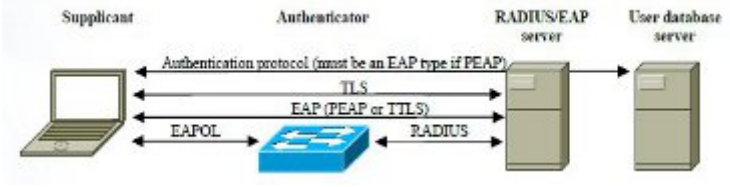
EAP Types TLS, TTLS and PEAP

As previously mentioned, the ‘basic’ authentication types should not be used. They do not provide sufficient protection for use on a shared network and, in particular, do not allow negotiation of the keying material required for IEEE 802.11 wireless LAN encryption. Consequently, a number of more secure types have been developed. Of these, only three have been widely implemented: TLS [3], TTLS [4] and PEAP [5].

The TLS EAP type is based on the Transport Layer Security (TLS) [6] protocol, which uses public key cryptography for authentication and negotiation of keys that can be used to encrypt data. TLS is also the protocol used for securing HTTPS. The main difference is that HTTPS is transported over TCP, whereas EAP TLS is transported over the EAP session between the supplicant and EAP server. As in HTTPS, the supplicant authenticates the server’s identity using a locally stored root certificate. However, unlike most HTTPS transactions, EAP TLS uses a user certificate to authenticate the supplicant to the server.

This means TLS can only be used by organisations with a Certificate Authority (CA) that issues user certificates; as such, although it offers excellent security, it is not widely deployed. Instead, two further EAP types, Protected EAP (PEAP) and Tunneled TLS (TTLS), work around this problem. Both of these types also use TLS for server authentication and encryption, but avoid the need for user certificates by using a second authentication protocol

between the supplicant and the server that is protected by the TLS encryption. This is very similar to conventional HTTPS authentication, where the user's plain-text credentials are protected by TLS. The main difference between the types is that PEAP can only protect other EAP types, whereas TTLS can protect almost any authentication protocol. An overview of the protocol layering is shown in Figure 2 below.



[2]

Figure 2: protocol layering in EAPs PEAP and TTLS.

Considerations When Implementing EAP

Determine how your passwords are stored in your user database(s), and which operating systems you need to support as supplicants and servers. These factors will largely determine which type(s) you deploy. TTLS should be considered as it will support authentication against all user databases. However, Windows' built-in supplicant only supports the MSCHAPv2 EAP type within PEAP, so if this supplicant is used then the EAP server must be able to authenticate MSCHAPv2 credentials. Windows' supplicant also integrates poorly with Windows domain authentication. While this is not typically required for casual network access, it can make it unsuitable for authenticating conventional desktop terminals; 'Longhorn' is expected to rectify these shortcomings. If Windows' supplicant is deemed unsuitable, a third-party supplicant may be used. The Janet IEEE 802.1X factsheet contains a comparison of supplicants. Ensure that your RADIUS server supports EAP, your chosen EAP type(s), and authentication against your user database(s). A variety of products are known to be used by JANET sites, including Microsoft®'s IAS, Open Solution's Radiator, Funk's Steelbelted RADIUS, Cisco®'s ACS and the open-source FreeRADIUS.

Consider how you will acquire a certificate for the RADIUS server. The use of a self-signed certificate is acceptable, although it means distributing the root certificate to every supplicant. If instead you acquire certificates from a vendor, ensure that you configure the supplicants to check the certificate's server name attribute, otherwise a rogue EAP server may be able to masquerade as the authorised EAP server by acquiring a certificate from the same vendor.

Bibliography

1: B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, H. Levkowetz, Ed., RFC3748 - Extensible Authentication Protocol (EAP), 2004.

2: Hostap development team, WPA Supplicant: http://hostap.epitest.fi/wpa_supplicant/ [3].

3: B. Aboba, D. Simon, RFC2716 - PPP EAP TLS Authentication Protocol, 1999.

4: Paul Funk, Simon Blake-Wilson, EAP Tunneled TLS Authentication Protocol, 2002.

5: Ashwin Palekar, Dan Simon, Glen Zorn, Joe Salowey, Hao Zhou, S. Josefsson, Protected EAP Protocol (PEAP)

Version 2, 2003.

6: T. Dierks, C. Allen, The TLS Protocol Version 1.0 , 1999.

Further reading

1: IEEE Computer Society, 802.1X - Port Based Network Access Control, 2001.

2: B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, H. Levkowetz, Ed., RFC3748 - Extensible Authentication Protocol (EAP), 2004.

3: C. Rigney, S. Willens, A. Rubens, W. Simpson, RFC2865 - Remote Authentication Dial In User Service (RADIUS) , 2000.

4: P. Congdon, B. Aboba, A. Smith, G. Zorn, J. Roese, RFC3580 - IEEE 802.1X Remote

Authentication Dial In User Service (RADIUS) Usage Guidelines, 2003.

5: IEEE Computer Society, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1999.

6: IEEE Computer Society, Medium Access Control (MAC) Security Enhancements, 2004.

7: WiFi Alliance, WiFi security at work and on the road

8: Cisco Systems, Network Admission Control: http://www.cisco.com/en/US/netsol/ns466/ [4]

9: Microsoft Corp, Network Access Protection: http://www.microsoft.com/nap [5]

10: Trusted Computing Group, Trusted Network Connect: http://www.trustedcomputinggroup.org/developers/trusted_network_connect/ [6]

Trademarks

Links
[1] http://community.ja.net/system/files/images/wtas-eap-fs-01.jpg
[2] http://community.ja.net/system/files/images/wtas-eap-fs-02.jpg
[3] http://hostap.epitest.fi/wpa_supplicant/
[4] http://www.cisco.com/en/US/netsol/ns466/
[5] http://www.microsoft.com/nap
[6] http://www.trustedcomputinggroup.org/developers/trusted_network_connect/