802.1X Port-Based Authentication HOWTO

Lars Strand

2004-08-18

Revision History

Revision 1.0 2004-10-18 Revised by: LKS

Initial Release, reviewed by TLDP.

Revision 0.2b 2004-10-13 Revised by: LKS

Various updates. Thanks to Rick Moen for language
This document describes the software and procedures to set up and use IEEE 802.1X Port-Based Network Access Control using Xsupplicant as Supplicant with FreeRADIUS as a back-end Authentication Server.

1. Introduction

1.1. What is 802.1X?

1.2. What is 802.11i?

1.3. What is EAP?

1.4. EAP authentication methods

1.5. What is RADIUS?

2. Obtaining Certificates

3. Authentication Server: Setting up FreeRADIUS

3.1. Installing FreeRADIUS

3.2. Configuring FreeRADIUS

4. Supplicant: Setting up Xsupplicant

4.1. Installing Xsupplicant
4.2. Configuring Xsupplicant

5. Authenticator: Setting up the Authenticator (Access Point)

5.1. Access Point

5.2. Linux Authenticator

6. Testbed

6.1. Testcase

6.2. Running some tests

7. Note about driver support and Xsupplicant

8. FAQ

9. Useful Resources

10. Copyright, acknowledgments and miscellaneous

10.1. Copyright and License

10.2. How this document was produced

10.3. Feedback

10.4. Acknowledgments

A. GNU Free Documentation License

A.1. PREAMBLE

A.2. APPLICABILITY AND DEFINITIONS
A.3. VERBATIM COPYING

A.4. COPYING IN QUANTITY

A.5. MODIFICATIONS

A.6. COMBINING DOCUMENTS

A.7. COLLECTIONS OF DOCUMENTS

A.8. AGGREGATION WITH INDEPENDENT WORKS

A.9. TRANSLATION

A.10. TERMINATION

A.11. FUTURE REVISIONS OF THIS LICENSE

A.12. ADDENDUM: How to use this License for your documents

1. Introduction

This document describes the software and procedures to set up and use 802.1X: Port-Based Network Access Control using Xsupplicant with PEAP (PEAP/ MS-CHAPv2) as authentication method and FreeRADIUS as back-end authentication server.

If another authentication mechanism than PEAP is preferred, e.g., EAP-TLS or EAP-TTLS, only a small number of configuration options needs to be changed. PEAP/MS-CHAPv2 are also supported by Windows XP SP1/Windows 2000 SP3.

-----

1.1. What is 802.1X?
The 802.1X-2001 standard states:

"Port-based network access control makes use of the physical access characteristics of IEEE 802 LAN infrastructures in order to provide a means of authenticating and authorizing devices attached to a LAN port that has point-to-point connection characteristics, and of preventing access to that port in cases which the authentication and authorization fails. A port in this context is a single point of attachment to the LAN infrastructure." --- 802.1X-2001, page 1.

Figure 802.1X: A wireless node must be authenticated before it can gain access to other LAN resources.

1. When a new wireless node (WN) requests access to a LAN resource, the access point (AP) asks for the WN's identity. No other traffic than EAP is allowed before the WN is authenticated (the "port" is closed). The wireless node that requests authentication is often called Supplicant, although it is more correct to say that the wireless node contains a Supplicant. The Supplicant is responsible for responding to Authenticator data that will establish its credentials. The same goes for the access point; the Authenticator is not the access point. Rather, the access point contains an Authenticator. The Authenticator does not even need to be in the access point; it can be an external component. EAP, which is the protocol used for authentication, was originally used for dial-up PPP. The identity was the username, and either PAP or CHAP authentication was used to check the user's password. Since the identity is sent in clear (not encrypted), a malicious sniffer may learn the user's identity. "Identity hiding" is therefore used; the real identity is not sent before the encrypted TLS tunnel is up. 2. After the identity has been sent, the authentication process begins. The protocol used between the Supplicant and the Authenticator is EAP, or, more correctly, EAP encapsulation over LAN (EAPOL). The Authenticator re-encapsulates the EAP messages to RADIUS format, and passes them to the Authentication Server. During authentication, the Authenticator just
relays packets between the Supplicant and the Authentication Server. When the authentication process finishes, the Authentication Server sends a success message (or failure, if the authentication failed). The Authenticator then opens the "port" for the Supplicant. 3. After a successful authentication, the Supplicant is granted access to other LAN resources/Internet.

See figure 802.1X for explanation.

Why is it called "port"-based authentication? The Authenticator deals with controlled and uncontrolled ports. Both the controlled and the uncontrolled port are logical entities (virtual ports), but use the same physical connection to the LAN (same point of attachment).

[8021X-Ports]

Figure port: The authorization state of the controlled port.

Before authentication, only the uncontrolled port is "open". The only traffic allowed is EAPOL; see Authenticator System 1 on figure port. After the Supplicant has been authenticated, the controlled port is opened, and access to other LAN resources are granted; see Authenticator System 2 on figure port.

802.1X plays a major role in the new IEEE wireless standard 802.11i.

-----

1.2. What is 802.11i?

1.2.1. WEP

Wired Equivalent Privacy (WEP), which is part of the original 802.11 standard, should provide confidentiality. Unfortunately WEP is poorly designed and easily cracked. There is no authentication mechanism, only a weak form of access control (must have the shared key to communicate).

As a response to WEP broken security, IEEE has come up with a new wireless security standard named 802.11i. 802.1X plays a major role in this new standard.

1.2.2. 802.11i

The new security standard, 802.11i, which was ratified in June 2004, fixes all WEP weaknesses. It is divided into three main categories:

1. Temporary Key Integrity Protocol (TKIP) is a short-term solution that fixes all WEP weaknesses. TKIP can be used with old 802.11 equipment (after a driver/firmware upgrade) and provides integrity and confidentiality.
2. Counter Mode with CBC-MAC Protocol (CCMP) [http://www.ietf.org/rfc/rfc3610.txt] RFC2610 is a new protocol, designed from ground up. It uses AES [FIPS 197] as its cryptographic algorithm, and, since this is more CPU intensive than RC4 (used in WEP and TKIP), new 802.11 hardware may be required. Some drivers can implement CCMP in software. CCMP provides integrity and confidentiality.
3. 802.1X Port-Based Network Access Control: Either when using TKIP or CCMP, 802.1X is used for authentication.

In addition, an optional encryption method called "Wireless Robust Authentication Protocol" (WRAP) may be used instead of CCMP. WRAP was the original AES-based proposal for 802.11i, but was replaced by CCMP since it became plagued by property encumbrances. Support for WRAP is optional, but CCMP support is mandatory in 802.11i.

802.11i also has an extended key derivation/management, described next.

1.2.3. Key Management
1.2.3.1. Dynamic key exchange and management

To enforce a security policy using encryption and integrity algorithms, keys must be obtained. Fortunately, 802.11i implements a key derivation/management regime. See figure KM.

[8021X-KeyManagement]

Figure KM: Key management and distribution in 802.11i.

1. When the Supplicant (WN) and Authentication Server (AS) authenticate, one of the last messages sent from AS, given that authentication was successful, is a Master Key (MK). After it has been sent, the MK is known only to the WN and the AS. The MK is bound to this session between the WN and the AS. 2. Both the WN and the AS derive a new key, called the Pairwise Master Key (PMK), from the Master Key. 3. The PMK is then moved from the AS to the Authenticator (AP). Only the WN and the AS can derive the PMK, else the AP could make access-control decisions instead of the AS. The PMK is a fresh symmetric key bound to this session between the WN and the AP. 4. PMK and a 4-way handshake are used between the WN and the AP to derive, bind, and verify a Pairwise Transient Key (PTK). The PTK is a collection of operational keys:  + Key Confirmation Key (KCK), as the name implies, is used to prove the posession of the PMK and to bind the PMK to the AP.  + Key Encryption Key (KEK) is used to distributed the Group Transient Key (GTK). Described below.  + Temporal Key 1 & 2 (TK1/TK2) are used for encryption. Usage of TK1 and TK2 is ciphersuite-specific. See figure PKH for a overview of the Pairwise Key Hierarchy. 5. The KEK and a 4-way group handshake are then used to send the Group Transient Key (GTK) from the AP to the WN. The GTK is a shared key among all Supplicants connected to the same Authenticator, and is used to secure multicast/broadcast traffic.

[8021X-KeyHierarchy]

Figure PKH: Pairwise Key Hierarchy
1.2.3.2. Pre-shared Key

For small office / home office (SOHO), ad-hoc networks or home usage, a pre-shared key (PSK) may be used. When using PSK, the whole 802.1X authentication process is elided. This has also been called "WPA Personal" (WPA-PSK), whereas WPA using EAP (and RADIUS) is "WPA Enterprise" or just "WPA".

The 256-bit PSK is generated from a given password using PBKDFv2 from [[http://www.ietf.org/rfc/rfc2898.txt] RFC2898], and is used as the Master Key (MK) described in the key management regime above. It can be one single PSK for the whole network (insecure), or one PSK per Supplicant (more secure).

1.2.4. TSN (WPA) / RSN (WPA2)

The industry didn't have time to wait until the 802.11i standard was completed. They wanted the WEP issues fixed now! [http://www.wi-fi.org/] Wi-Fi Alliance felt the pressure, took a "snapshot" of the standard (based on draft 3), and called it Wi-Fi Protected Access (WPA). One requirement was that existing 802.11 equipment could be used with WPA, so WPA is basically TKIP + 802.1X.

WPA is not the long term solution. To get a Robust Secure Network (RSN), the hardware must support and use CCMP. RSN is basically CCMP + 802.1X.

RSN, which uses TKIP instead of CCMP, is also called Transition Security Network (TSN). RSN may also be called WPA2, so that the market don't get confused.

Confused?
Basically:

* TSN = TKIP + 802.1X = WPA(1)   * RSN = CCMP + 802.1X = WPA2

In addition comes key management, as described in the previous section.

-----

1.3. What is EAP?

Extensible Authentication Protocol (EAP) [[http://www.ietf.org/rfc/rfc3748.txt] RFC 3748] is just the transport protocol optimized for authentication, not the authentication method itself:

" [EAP is] an authentication framework which supports multiple authentication methods. EAP typically runs directly over data link layers such as Point-to-Point Protocol (PPP) or IEEE 802, without requiring IP. EAP provides its own support for duplicate elimination and retransmission, but is reliant on lower layer ordering guarantees. Fragmentation is not supported within EAP itself; however, individual EAP methods may support this." --- RFC 3748, page 3

-----

1.4. EAP authentication methods

Since 802.1X is using EAP, multiple different authentication schemes may be added, including smart cards, Kerberos, public key, one time passwords, and others.

Some of the most-used EAP authentication mechanism are listed below. A full list of registered EAP authentication types is available at IANA: [http://www.iana.org/assignments/eap-numbers] http://www.iana.org/assignments/eap-numbers.

Warning Not all authentication mechanisms are considered secure!
* EAP-MD5: MD5-Challenge requires username/password, and is equivalent to the PPP CHAP protocol [[http://www.ietf.org/rfc/rfc1994.txt] RFC1994]. This method does not provide dictionary attack resistance, mutual authentication, or key derivation, and has therefore little use in a wireless authentication environment. * Lightweight EAP (LEAP): A username/password combination is sent to a Authentication Server (RADIUS) for authentication. Leap is a proprietary protocol developed by Cisco, and is not considered secure. Cisco is phasing out LEAP in favor of PEAP. The closest thing to a published standard can be found [http://lists.cistron.nl/pipermail/cistron-radius/2001-September/002042.html] here. * EAP-TLS: Creates a TLS session within EAP, between the Supplicant and the Authentication Server. Both the server and the client(s) need a valid (x509) certificate, and therefore a PKI. This method provides authentication both ways. EAP-TLS is described in [[http://www.ietf.org/rfc/rfc2716.txt] RFC2716]. * EAP-TTLS: Sets up a encrypted TLS-tunnel for safe transport of authentication data. Within the TLS tunnel, (any) other authentication methods may be used. Developed by Funk Software and Meetinghouse, and is currently an IETF draft. * Protected EAP (PEAP): Uses, as EAP-TTLS, an encrypted TLS-tunnel. Supplicant certificates for both EAP-TTLS and EAP-PEAP are optional, but server (AS) certificates are required. Developed by Microsoft, Cisco, and RSA Security, and is currently an IETF draft. * EAP-MSCHAPv2: Requires username/password, and is basically an EAP encapsulation of MS-CHAP-v2 [[http://www.ietf.org/rfc/rfc2759.txt] RFC2759]. Usually used inside of a PEAP-encrypted tunnel. Developed by Microsoft, and is currently an IETF draft.

----- 1.5. What is RADIUS?

Remote Authentication Dial-In User Service (RADIUS) is defined in [[http://www.ietf.org/rfc/rfc2865.txt] RFC2865] (with friends), and was primarily used by ISPs who authenticated username and password before the user got authorized to use the ISP's network.
802.1X does not specify what kind of back-end authentication server must be present, but RADIUS is the "de-facto" back-end authentication server used in 802.1X.

There are not many AAA protocols available, but both RADIUS and DIAMETER [[http://www.ietf.org/rfc/rfc3588.txt] RFC3588] (including their extensions) conform to full AAA support. AAA stands for Authentication, Authorization, and Accounting (IETF's AAA Working Group).

-----

2. Obtaining Certificates

Note OpenSSL must be installed to use either EAP-TLS, EAP-TTLS, or PEAP!

When using EAP-TLS, both the Authentication Server and all the Supplicants (clients) need certificates [[http://www.ietf.org/rfc/rfc2459.txt] RFC2459] . Using EAP-TTLS or PEAP, only the Authentication Server requires certificates; Supplicant certificates are optional.

You get certificates from the local certificate authority (CA). If there is no local CA available, OpenSSL may be used to generate self-signed certificates.

Included with the FreeRADIUS source are some helper scripts to generate self-signed certificates. The scripts are located under the scripts/ folder included with the FreeRADIUS source:

CA.all is a shell script that generates certificates based on some questions it ask. CA.certs generates certificates non-interactively based on pre-defined information at the start of the script.

Note The scripts uses a Perl script called CA.pl, included with OpenSSL. The path to this Perl script in CA.all and CA.certs may need to be changed.
to make it work.

Tip More information on how to generate your own certificates can be found in the SSL certificates HOWTO.

-----

3. Authentication Server: Setting up FreeRADIUS

FreeRADIUS is a fully GPLed RADIUS server implementation. It supports a wide range of authentication mechanisms, but PEAP is used for the example in this document.

-----

3.1. Installing FreeRADIUS

Installing FreeRADIUS

1. Head over to the FreeRADIUS site, [http://www.freeradius.org/](http://www.freeradius.org/), and download the latest release. # cd /usr/local/src # wget ftp://ftp.freeradius.org/pub/radius/freeradius-1.0.0.tar.gz # tar zxfv freeradius-1.0.0.tar.gz # cd freeradius-1.0.0 2. Configure, make and install: # ./configure # make # make install You can pass options to configure. Use ./configure --help or read the README file, for more information.

The binaries are installed in /usr/local/bin and /usr/local/sbin. The configuration files are found under /usr/local/etc/raddb.

If something went wrong, check the INSTALL and README included with the source. The [http://www.freeradius.org/faq/](http://www.freeradius.org/faq/) RADIUS FAQ also contains valuable information.

-----

3.2. Configuring FreeRADIUS
FreeRADIUS has a big and mighty configuration file. It's so big, it has been split into several smaller files that are just "included" into the main radius.conf file.

There is numerous ways of using and setting up FreeRADIUS to do what you want: i.e., fetch user information from LDAP, SQL, PDC, Kerberos, etc. In this document, user information from a plain text file, users, is used.

Tip The configuration files are thoroughly commented, and, if that is not enough, the doc/ folder that comes with the source contains additional information.

Configuring FreeRADIUS

1. The configuration files can be found under /usr/local/etc/raddb/ # cd /usr/local/etc/raddb/ 2. Open the main configuration file radiusd.conf, and read the comments! Inside the encrypted PEAP tunnel, an MS-CHAPv2 authentication mechanism is used. a. MPPE

[[http://www.ietf.org/rfc/rfc3078.txt] RFC3078] is responsible for sending the PMK to the AP. Make sure the following settings are set: # under MODULES, make sure mschap is uncommented! mschap { # authtype value, if present, will be used # to overwrite (or add) Auth-Type during # authorization. Normally, should be MS-CHAP authtype = MS-CHAP # if usemppe is not set to no, mschap will # add MS-CHAP-MPPE-Keys for MS-CHAPv1 and # MS-MPPE-Recv-Key/MS-MPPE-Send-Key for MS-CHAPv2 # usemppe = yes # if mppe is enabled, requireencryption makes # encryption moderate # requireencryption = yes # requirestrong always requires 128 bit key # encryption # requirestrong = yes authtype = MS-CHAP # The module can perform authentication itself, OR # use a Windows Domain Controller. See the radius.conf file # for how to do this. } b. Also make sure the "authorize" and "authenticate" contains: authorize { preprocess mschap suffix eap files } authenticate { # # MSCHAP authentication. Auth-Type MS-CHAP { mschap } # # Allow EAP authentication. eap } 3. Then, change the clients.conf file to specify what network it's serving: # Here, we specify which network we're serving client 192.168.0.0/16 { # This is the shared secret between the Authenticator (the
# access point) and the Authentication Server (RADIUS). secret = SharedSecret99 shortname = testnet } 4. The eap.conf should also be pretty straightforward. a. Set "defaulteaptype" to "peap": defaulteaptype = peap b. Since PEAP is using TLS, the TLS section must contain: tls { # The private key password privatekeypassword = SecretKeyPass77 # The private key privatekeyfile = ${raddbdir}/certs/cert-srv.pem # Trusted Root CA list CAfile = ${raddbdir}/certs/demoCA/cacert.pem dhfile = ${raddbdir}/certs/dh randomfile = /dev/urandom } c. Find the "peap" section, and make sure it contain the following: peap { # The tunneled EAP session needs a default # EAP type, which is separate from the one for # the non-tunneled EAP module. Inside of the # PEAP tunnel, we recommend using MS-CHAPv2, # as that is the default type supported by # Windows clients. defaulteaptype = mschapv2 } 5. The user information is stored in a plain text file users. A more sophisticated solution to store user information may be preferred (SQL, LDAP, PDC, etc.). Make sure the users file contains the following entry: "testuser" User-Password == "Secret149"

----- 4. Supplicant: Setting up Xsupplicant

The Supplicant is usually a laptop or other (wireless) device that requires authentication. Xsupplicant does the bidding of being the "Supplicant" part of the IEEE 802.1X-2001 standard.

-----

4.1. Installing Xsupplicant

Installing Xsupplicant

1. Download the latest source from from [http://www.open1x.org/](http://www.open1x.org/) http://www.open1x.org/ # cd /usr/local/src # wget http://belnet.dl.sourceforge.net/sourceforge/open1x/xsupplicant-1.0.tar.gz # tar zxfv xsupplicant-1.0.tar.gz # cd xsupplicant 2. Configure, make, and install: # ./configure # make # make install 3. If the configuration file wasn't installed (copied) into the "etc" folder, do it manually: # mkdir -p /usr/local/etc/1x # cp etc/tls-example.conf /usr/local/etc/1x
If installation fails, check the README and INSTALL files included with the source. You may also check out the official documentation.

4.2. Configuring Xsupplicant

Configuring Xsupplicant

1. The Supplicant must have access to the root certificate. If the Supplicant needs to authenticate against the Authentication Server (authentication both ways), the Supplicant must have certificates as well. Create a certificate folder, and move the certificates into it:

   # mkdir -p /usr/local/etc/1x/certs
   # cp root.pem /usr/local/etc/1x/certs/
   # (copy optional client certificate(s) into the same folder)

2. Open and edit the configuration file:

   # startup command: the command to run when Xsupplicant is first started.
   # This command can do things such as configure the card to associate with the network properly.

   startupcommand =