

Securing Network Access with 802.1X, Radius, and LDAP

SHUTTING OUT STRANGERS

The Radius protocol is typically used to authenticate users in dial-up scenarios. But Radius is also useful in LAN environments: in combination with 802.1X, Radius forces users to authenticate at a low level before the switch opens up a port.

BY MICHAEL SCHWARTZKOPFF

original photo: www.sxc.hu

Attacks from internal networks are dangerous and more difficult to prevent than external attacks. An attacker who plugs in to an internal network with a laptop gains wide-ranging access to network data. One way of preventing an attack is to implement an authentication function in OSI Layer 2 using the 802.1X [1] protocol. An 802.1X-capable switch and a Freeradius server are all you need to implement Layer 2 authentication. Because Layer 2 authentication operates at the level of the local, physical network, it prevents an intruder from even using the physical network without authentication.

Radius (Remote Authentication Dial-in User Service Protocol) responses from a Linux server typically include the IP address and standard gateway for the user, but the protocol has more potential. You can use a Radius server to assign a VLAN to the user's switch port. This technique avoids the need for a complex router infrastructure, but still restricts the size of the broadcast domain. Additionally, VLANs can be

used to separate company departments logically, improving security at the same time. Although users can log in wherever they are (from the canteen, for example), they will always see their own network environment.

Plain vanilla 802.1X handles authentication, with the Freeradius server providing the AAA services (Authentication, Authorization, and Accounting.) The Freeradius server accesses an OpenLDAP directory server for account information. The whole system is suitable for both Linux and Windows clients. It supports redundant layouts for high-availability using proxies for Radius and directory replication for the LDAP database.

The whole solution can also be applied to WLAN security (see the box "802.1X and Wireless LAN"), and administrators can benefit from the Radius server's accounting options.

802.1X and EAP

The IEEE 802.1X protocol provides access control in OSI Layer 2 (the MAC Layer). IEEE 802.1X supports the

authentication of clients while establishing a connection to the network, before the client is assigned an IP address via DHCP (Dynamic Host Configuration Protocol). Among other things, the standard specifies how the authentication protocol (EAP, Extensible Authentication Protocol) is encapsulated in Ethernet frames.

EAP provides a framework for various authentication methods that support more than the usual combination of user name and password. EAP uses the Network Access Server (Authenticator) to open a tunnel to the authentication server across the network, and it allows other protocols to use the tunnel. 802.1X defines a number of special terms:

- A client requesting authentication is known as a Supplicant.
- The server that authenticates the client is known as the Authentication Server.
- The intermediate device between these two entities is the Network Authentication Server (NAS) or Authenticator.

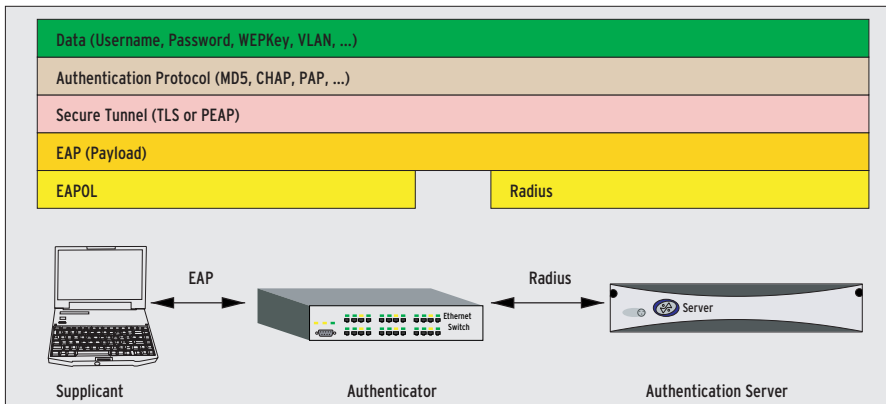


Figure 1: The 802.1X protocol covers multiple layers. EAP handles exchanges between the supplicant and the authenticator; Radius is responsible for the path between the authenticator and the authentication server.

This setup will work on any network that delivers Ethernet packets. The Interopnet Labs white paper provides a useful overview [1].

EAP Variants

EAP defines a variety of authentication methods. EAP/MD5 transfers a hash with the user name, password, and an arbitrary seed. The server uses the clear-text password and the arbitrary seed to generate its own hash, which it compares with the incoming hash. This method is simple, but it is insecure against dictionary attacks. Also, in a wireless LAN, it is impossible to create dynamic WEP keys using EAP/MD5. Thus, this approach is only suitable for smaller, wired networks.

With the second variant, EAP/TLS, the server and client both need X.509 certificates. This method is a lot safer, but it assumes a working PKI (Public Key Infrastructure). A third method is PEAP, Protected Extensible Authentication Protocol. With PEAP, only the server needs a certificate; the protocol uses the certificate to establish a TLS connection and send the encrypted user name and password (MSCHAPv2, Microsoft Challenge Handshake Authentication Protocol). Administrators simply need to install the server certificate on every client.

When the client logs off or closes the connection, PEAP detects the change and terminates the authorization, closing the connection on both sides.

Spreading the Load

In wired-only networks, EAP/MD5 is often the best option. It is all you need to

dynamically assign VLANs, and – in contrast to PEAP – it is supported by a wide variety of switches. Additionally, the administrative effort involved is far less than with PEAP or EAP/TLS.

A switch normally provides NAS functionality, translating the EAPOL protocol (EAP over LAN) from the supplicant to Radius, which is what the access server expects. Most devices give you this option when configuring 802.1X. You need to enter the address and password of the Radius server. In many cases, administrators can configure multiple servers to provide higher levels of avail-

ability and to offer a backout solution in case the main server goes down.

Freeradius

Freeradius [4] is a good choice of Radius server. The recent version 1.0 adds support for a large number of EAPs, specifically, for PEAP. The developers have introduced an option for authenticating against a Windows domain. And Freeradius has had the ability to retrieve account data from typical sources */etc/passwd*, LDAP, MySQL, PostgreSQL, or Oracle databases, for quite a while now.

The install is quite simple and follows the three step approach of *configure && make && make install*. If all goes well, this should put the server configuration files in */usr/local/etc/raddb*. The first thing Freeradius needs to do is to grant access to the Radius clients (the NAS in our case). The *clients.conf* file handles this. The configuration for a switch with the static IP address *192.168.200.20* would look like this, for example:

```
client 192.168.200.20 {
    secret = testing123
    shortname = switch
}
```

802.1X and Wireless LAN

Freeradius in combination with 802.1X can bring authentication to a wireless LAN. This combination can restrict the validity of insecure WEP keys (Wired Equivalent Privacy) for a client to half an hour, for example. But this requires PEAP (the Protected Extensible Authentication Protocol) or TLS mode, as MD5 hashes are incapable of generating keys. There is a good (if not state-of-the-art) HOWTO at [2].

To be able to use PEAP, admins need to create a certificate for the Radius server. For Windows clients, the certificate needs a specific OID (Object ID) as its purpose. There is a script called *CA.all* in the *scripts/* directory of the Freeradius sources that generates sample certificates for the server and clients. The client certificate is only required for use with EAP/TLS.

Easy Administration with TinyCA

If you prefer a GUI-based approach to certificate management, why not try TinyCA [3] version 0.6.4 or later? This tool allows administrators to add the

required OID (for example 1.3.6.1.5.5.7.3.1 for the server) in the advanced usage dialog for the certificate.

The next step is to copy the new server certificate (the script creates a certificate called *srv-cert.pem*) and the CA root certificate, *root.pem*, to the *certs* directory below the Radius server configuration directory. If you will be using TinyCA, you additionally need to copy the file with the private key. Windows users can install the CA certificate *root.der* by double-clicking on the file. This also applies to the Radius server certificate, *srv-cert.p12* (in PKCS#12 format).

Now set up the *tls* and *peap* sections of the Radius server EAP configuration as described at [2]. The lines you need are included in the default configuration but disabled. If you launch the Radius service in debug mode (*radiusd -X*), the Radius service will output verbose messages, which can be a big help when troubleshooting.

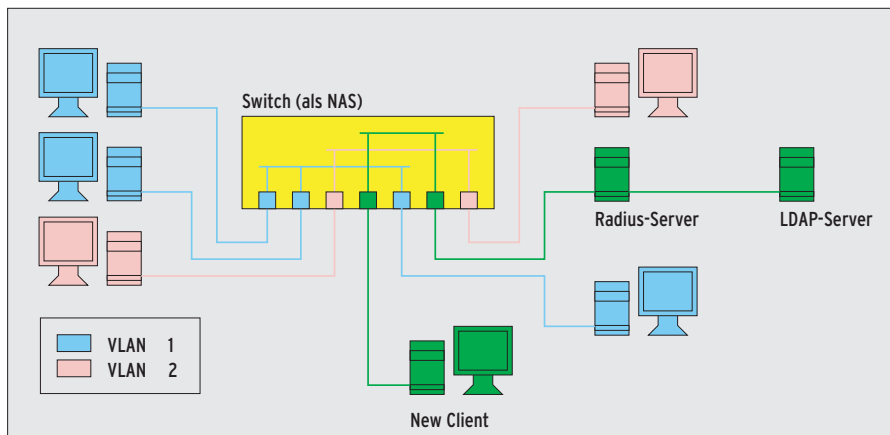


Figure 2: New clients first have to authenticate against the NAS (that is the switch). The switch acts as a proxy to the Radius server, which in turn accesses a LDAP directory to retrieve user credentials.

The administrator needs to enter the password on the switch (this is “testing123” in our case) when configuring the Radius server. Freeradius will accept CIDR (Classless Inter-Domain Routing) notation for whole networks: 192.168.200.0/24.

Authenticating Users

The *users* file specifies the user authentication type. The following entry should be fine for initial testing:

```
testuser Auth-Type := Local, ↗
User-Password == "tpasswd"
Reply-Message = "Hello, %u"
```

You will want to run the Radius server in debug mode initially. *radiusd -X* tells the server to output error messages and warnings on the console. The *radtest* command line program provides a useful test client tool that can help you test your configuration. Assuming the settings mentioned so far, the command line for the test client is as follows: *radtest testuser tpaswd localhost 0 testing123*. The last three parameters reference the Radius server – to allow this to work, you need to add *localhost* as a valid Radius client to *clients.conf*. Freera-

Listing 1: Attribute Mapping

```
01 replyItem Tunnel-Type
   radiusTunnelType
02 replyItem Tunnel-Medium-Type
   radiusTunnelMediumType
03 replyItem
   Tunnel-Private-Group-Id
   radiusTunnelPrivateGroupId
```

dius should respond by outputting *Access-Accept*.

VLAN in the Response

Of course, the Radius protocol can give you more than a simple message as a response. It can give you the number of a VLAN, for example, allowing the switch to evaluate the number and accept the client into the required VLAN. To allow this to happen, you need to add the Radius response to the test user configuration. Note that return values need to be comma separated and indented with space characters.

```
testuser Auth-Type := Local, ↗
User-Password == "tpaswd"
Reply-Message = "Hello, %u",
Tunnel-Medium-Type = IEEE-802,
Tunnel-Private-Group-Id = 1,
Tunnel-Type = VLAN
```

This tells the switch to assign VLAN 1 to the *testuser* after authenticating the users. Administrators can simply add manual configurations for the remaining users. Of course, this approach is impractical for larger networks.

OpenLDAP

OpenLDAP [5] is a good choice for a Freeradius back-end (see Figure 2). You’ll find a good HOWTO on combining Freeradius and OpenLDAP by Dustin Doris [6]. The Freeradius documentation also has an example in a text document called *doc/rlm_ldap*.

However, there is a pitfall in the LDAP schema; *Radius-LDAPv3.schema* (which is also located below the *doc* directory)

Listing 2: LDIF for VLAN 2

```
01 dn:uid=vlan_02,ou=profiles,
   ou=radius,dc=domain,dc=de
02 uid: vlan_02
03 radiusTunnelMediumType:
   IEEE-802
04 radiusTunnelType: VLAN
05 radiusTunnelPrivateGroupId: 2
06 objectClass: radiusprofile
07 objectClass: top
```

is not structural, that is, it only works in conjunction with another schema, for example, *inetorgperson.schema*. In contrast to this, the schema from [6] is designed specifically for use with Radius, but it does not understand the kind of extensions typically needed for daily operations.

To allow Freeradius to understand the responses related to the VLANs returned by the LDAP database, we need to add mappings (*ldap.attrmap*) to our Radius configuration (Listing 1).

Figure 3 shows the structure of the LDAP directory. The profiles (*ou=profiles*) contain VLAN configurations; Listing 2 gives you the LDIF file for *uid=vlan_02* (see line 1). For the user configuration (*ou=users*), we simply need a reference to the profile with the matching VLAN (Listing 3, line 6). Now, the Radius server simply needs to know that it can access LDAP for user management. The *radiusd.conf* configuration handles this.

The example in Listing 4 tells Radius to log on to the LDAP server, *ldap.domain.de*, (line 3) with the configured *identity*, and *password* (lines 4 and 5), and to authenticate the user *filter* with a password of *password_attribute* (lines 7 and 10). If this works, the LDAP server returns the parameters in the matching profile (mapped as *radiusProfileDn* for the user).

A new *DEFAULT* entry in the Radius user management (file: *users*) ensures

Listing 3: VLAN Reference

```
01 dn:uid=testuser2,ou=users,
   ou=radius,dc=domain,dc=de
02 uid: testuser2
03 userPassword: password
04 objectClass: radiusprofile
05 objectClass: top
06 radiusProfileDn:
   uid=vlan_02,ou=profiles,ou=radius,dc=domain,dc=de
```

Listing 4: Radius Configuration

```

01 modules {                                ${raddbdir}/ldap.attrmap
02     ldap {                                10     password_attribute =
03         server =                          userPassword
04         "ldap.domain.de"                  11     }
05         identity =                        12     }
06         "cn=freeradius,ou=admins,ou=ra-   13
07         dius,dc=domain,dc=de"           14     authorize {
08         password = secret                 15         preprocess
09         basedn =                           16         ldap
09         "ou=users,ou=radius,dc=domain,   17         eap
09         dc=de"                             18         suffix
07         filter =                          19         files
07         "((&(uid=%{Stripped-User-Name:-  20     }
07         %{User-Name}})(objectclass=rad-  21
07         iusprofile))"                     22     authenticate {
08         start_tls = no                     23         eap
09         dictionary_mapping =              24     }

```

that authentication attempts using EAP will work:

```

DEFAULT Auth-Type == EAP
Fall-Through = yes

```

We still need to prepare the clients for 802.1X authentication. The Open 1X Project at [7] has developed software for Linux to handle this. Windows 2000 SP4 and WinXP SP1 have native authentication functions for this purpose. Windows 2000 additionally needs to launch the *Wireless Configuration* service.

Windows Client

There is an *Authentication* field in the network connection properties. Enter EAP with the required type (MD5, PEAP or TLS) in this field. Users can additionally specify how the system should respond if user credentials are not available, which is the case before logging in.

If you select this option, the computer will attempt to log in to the network using its client name.

In contrast to the EAP/MD5 option, PEAP and TLS provides additional options. Users have to specify the CA certificate the client should accept. Windows can also use the account name and password from the Windows login for PEAP authentication; this entry is located in the advanced authentication options.

However, in this case the system uses a Domain/Username combination. If you are not sure, the exact variant will be logged in the Radius server logfile after the first login attempt. To be able to understand this format, Radius needs a few hints. If you are interested in learning more about this subject, try the book on Radius at [8].

Latecomers...

Microsoft 802.1X clients have a basic problem. They first log on to their own domain and then authenticate against the network. But the network is not available to the client while it is trying to log on to the domain, and this causes the login attempt to fail. The domain server would need to reside in an open standard VLAN to allow the Windows supplicant to work, but this contravenes basic security.

There are a few third party tools that solve this problem by providing own 802.1X supplicants. These clients typically support far more granular configuration – allowing the client to be integrated into the domain login process. The login process then starts by authenticating the user against the network based on 802.1X, and then goes on to handle the normal Windows login. However, some programs have an unfinished look to them, so test before you buy.

Linux-based Clients and Servers

You'll find a mature and 802.1X software tool for Linux clients at [7]. This tool should make it easy to give Linux computers access to your network.

The Freeradius server has an enormous range of options for authenticating users and configuring access based on your networking requirements. The example we have looked at in this article is just one of many approaches. Because this approach uses a directory service for user management, the security design is also suitable for large-scale networks where the danger of internal attacks is particularly severe. ■

INFO

- [1] Interopnet Labs, "What is 802.1X?": http://www.ilabs.interop.net/WLANSec/What_is_8021x-lv03.pdf
- [2] Freeradius and Windows XP: <http://text.broadbandreports.com/forum/remark,9286052~mode=flat>
- [3] TinyCA: <http://tinyca.sm-zone.net>
- [4] Freeradius: <http://www.freeradius.org>
- [5] OpenLDAP: <http://www.openldap.org>
- [6] Freeradius and OpenLDAP: <http://doris.cc/radius/>
- [7] Open source implementation of 802.1X: <http://www.open1x.org>
- [8] Radius – Securing Public Access to Private Resources, by Jonathan Hassell; O'Reilly, 2002.

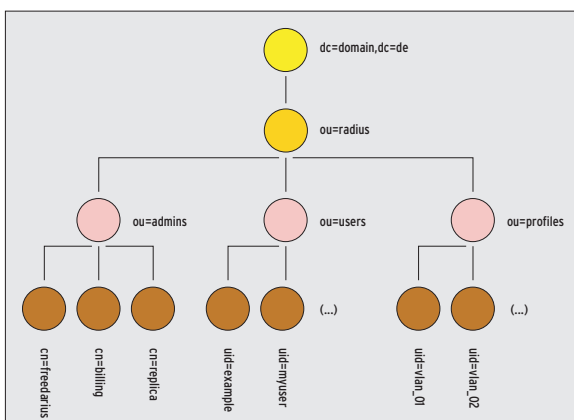


Figure 3: This LDAP directory structure contains user admin and other profiles for the VLAN configuration. In each user profile is a *radiusProfileDn* entry that points to the matching VLAN profile.

Michael Schwartzkopff works for Multinet Services GmbH as a security and networking consultant (specializing in SNMP). He caught the Linux bug way back in 1994 after working with a Yggdrasil distribution.

