RADIUS and Authentication

Chris Wilson
Aptivate Ltd, UK
AfNOG 2010

Based on a presentation by:
Frank A. Kuse, 12/05/2009

The Password Problem

- Many services require access control based on user identity
- Most services have their own specialised password database and authorization mechanism
- Most users have access to more than one service
- Password proliferation → password hell!
  - Ever forget to delete a user after they left?
  - Ever had a password database compromised?
- Authentication and authorization services solve all of these problems

Components of the Solution

- Four services that relate to users
- Authentication
  - Is this user really who they claim to be?
- Authorization
  - What is this user allowed to do?
- Accounting
  - What did this user actually do (in the past)
- Directory
  - What users do I have, and what do I know about them?

Authentication

- Is this user really who they claim to be?
- Requires presentation of an identity and credentials, such as:
  - Plain text password
  - Challenge response (hashed password)
  - Digital certificate and challenge signature
- Often confused with authorization

Authorization

- What is this user allowed to do?
  - Use a specific IP address
  - Log into POP3 and IMAP
  - Receive 1 Mbps download speed
- RADIUS can base authorization on rules:
  - Their authenticated user name
  - Time of day
  - Physical location
  - Number of simultaneous logins
  - Current date (and activation/expiry date)

Common Solutions

- LDAP
  - Authentication, Authorization and Directory
  - Microsoft, Netscape and Red Hat directory services
- Kerberos
  - Authentication (with bells on)
  - Part of Microsoft's Active Directory implementation
  - Commonly combined with LDAP
- RADIUS
  - Authentication, Authorization and Accounting
  - Commonly used at ISPs and for 802.1x security

Less Common Solutions

- DIAMETER
  - Designed by IETF to replace RADIUS
  - Better proxying, session control and security?
- TACACS
  - Cisco proprietary, very few implementations
- NIS
  - Old Sun standard, obsolete, insecure
- NIS+
  - Newer Sun standard, more difficult to administer

What is RADIUS?

- Remote Authentication Dial-In User Service
- Latest version defined by RFC 2865
- Network Protocol (like HTTP, FTP, SSH)
- Used for:
  - Authentication: is this user really who they claim to be?
  - Authorization: what are they allowed to do?
  - Accounting: recording what they did

Why use RADIUS?

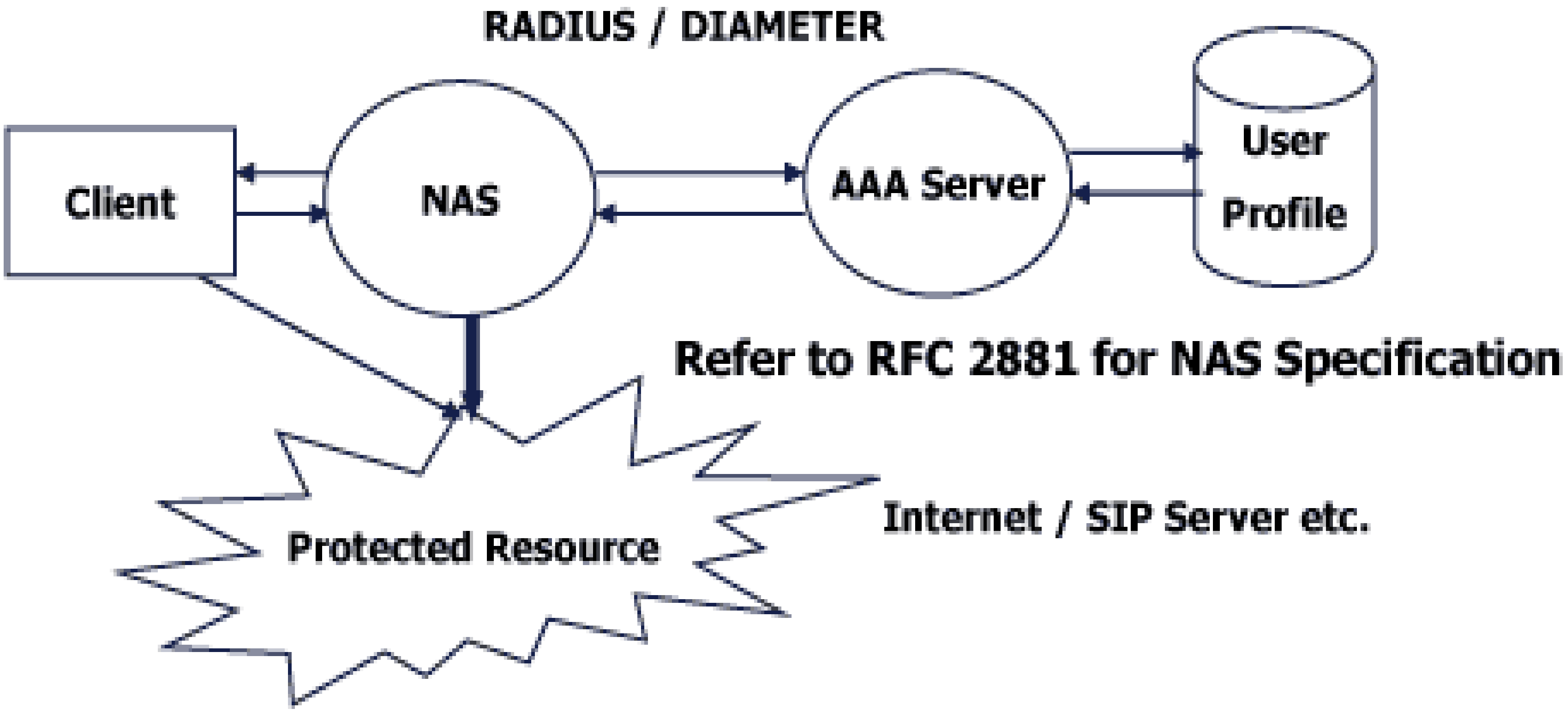
- Many services can authenticate against a RADIUS server:
  - PAM, and any Unix service that uses it, including SASL
  - Wired Ethernet switches and wireless access points
  - ADSL DSLAM (head end), PPP (e.g. L2TP)
- Create user accounts just once for all services
- Change passwords just once for all services
- Easily delete users after they leave
- Give users the same password for all services
- Easier to secure a single password store

RADIUS vs LDAP

	RADIUS	**LDAP**
Origin	Dial-in access control	Directory Services
Authentication	Yes	Yes
Authorization	Attributes	Group Memberships
Accounting	Yes	No
Directory	No	Yes
Scalability	High	Medium
Complexity	Low	High
Security	High	Medium

Conclusion: Horses for courses, or use both!
(RADIUS can authenticate against an LDAP backend)

Basic Architecture of RADIUS



Basic Architecture for NAS/RADIUS/AAA

Conventions

- File names and technical terms are in *italics*
- Commands to type are shown in monospaced bold italic purple type:
  - ***`cat /etc/monospaced/bold/italic/purple`***
- Long command lines are wrapped, but with a single bullet point at the start:
  - ***`cat /usr/local/etc/foo/bar | less | more | grep | sed | awk > /usr/local/tmp/foo/bar`***
- Text that is output by a program, or should already be in a file, is shown in plain monospaced type:
  - `sshd_enable="YES"`

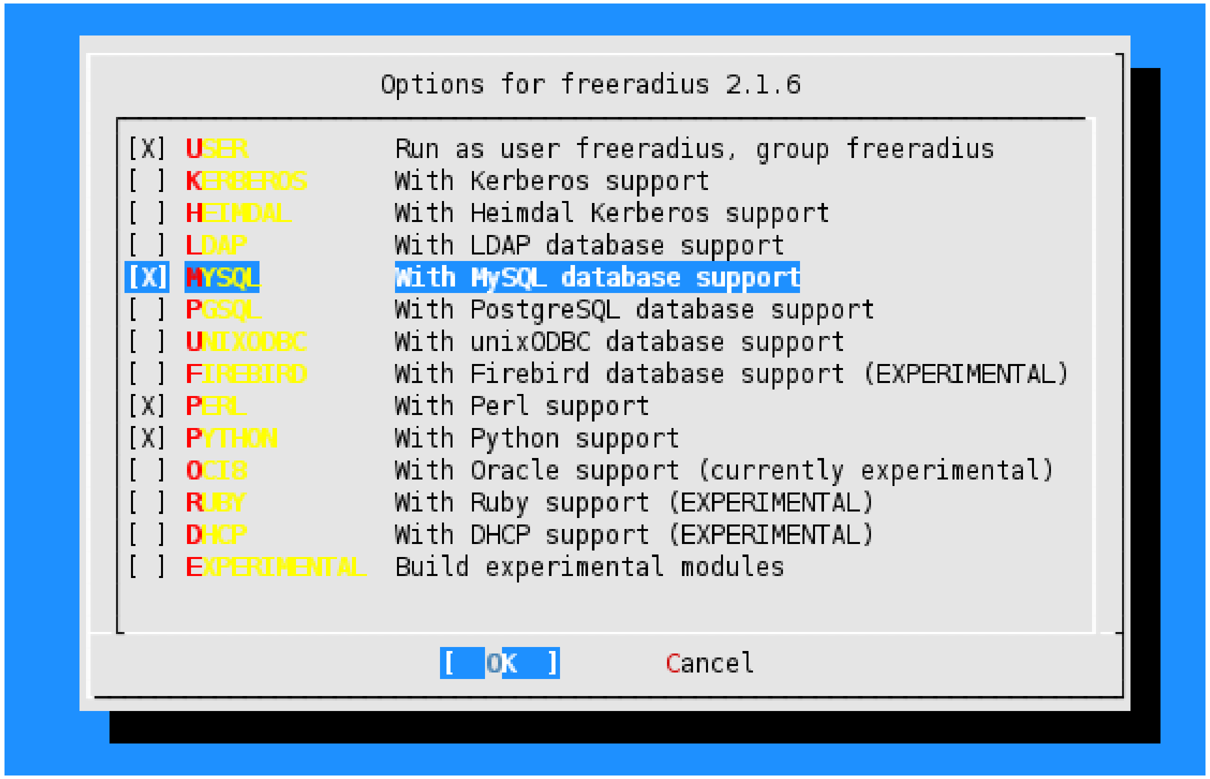
AfNOG

Installing Dependencies

- Install dependencies from packages:
  - ***`sudo pkg_add -r gmake mysql50-server autoconf262 libtool`***
  - `Fetching ftp://.../gmake.tbz... Done.`
  - `...`
  - `Fetching ftp://.../libtool.tbz... Done.`
- You can ignore errors like:
  - `pkg_add: package '...' or its older version already installed`

Installing FreeRADIUS (1)

- Configure the FreeRADIUS port:
  - ***cd /usr/ports/net/freeradius2***
  - ***sudo make config***
  - Highlight MySQL and press [Space] to select it



```
                    Options for freeradius 2.1.6

 [X] USER          Run as user freeradius, group freeradius
 [ ] KERBEROS      With Kerberos support
 [ ] HEIMDAL       With Heimdal Kerberos support
 [ ] LDAP          With LDAP database support
 [X] MYSQL         With MySQL database support
 [ ] PGSQL         With PostgreSQL database support
 [ ] UNIXODBC      With unixODBC database support
 [ ] FIREBIRD      With Firebird database support (EXPERIMENTAL)
 [X] PERL          With Perl support
 [X] PYTHON        With Python support
 [ ] OCI8          With Oracle support (currently experimental)
 [ ] RUBY          With Ruby support (EXPERIMENTAL)
 [ ] DHCP          With DHCP support (EXPERIMENTAL)
 [ ] EXPERIMENTAL  Build experimental modules

                 [  OK  ]        Cancel
```

AfNOG

Installing FreeRADIUS (2)

- Press [Tab] then [Enter] to save the settings
- Compile the port:
  - `make deinstall clean install clean`
- You should see a lot of output, finishing with:

```
For more information, and contact details
about the security status of this software,
see the following webpage:
```

```
http://www.freeradius.org/
```

- If you see anything else, please ask for help

Installing FreeRADIUS (3)

- Edit */etc/rc.conf* and add the line:
  - `radiusd_enable="YES"`
- Start the FreeRADIUS server now:
  - `sudo /usr/local/etc/rc.d/radiusd start`
  - `Starting radiusd.`

Testing FreeRADIUS

- Edit */usr/local/etc/raddb/sites-available/default* and comment out all the lines that say just:
  - `unix`
- Restart the FreeRADIUS server:
  - ***`sudo /usr/local/etc/rc.d/radiusd restart`***
  - `Stopping radiusd...`
  - `Starting radiusd.`
- Test that it responds properly:
  - ***`sudo radtest bob SEKret localhost 0 testing123`***
  - `rad_recv: Access-Reject packet ...`

Debugging radiusd

- *radiusd* will not start if there is a mistake in the configuration files
- Either check the system logs:
  - `sudo tail /var/log/radius.log`
- Or stop radius and start it in debugging mode:
  - `sudo /usr/local/etc/rc.d/radiusd stop`
  - `sudo /usr/local/etc/rc.d/radiusd debug`
  - (check that it starts, fix any errors, run your tests)
  - `sudo /usr/local/etc/rc.d/radiusd start`

Adding Users

- Edit */usr/local/etc/raddb/users* and add the following lines at the top:
  - `bob       Cleartext-Password := "SEKret"`
  - `afnog     Cleartext-Password := "success!"`
  - Be careful not to put spaces before the user names
- Restart *radiusd* (this is important!)
  - `sudo /usr/local/etc/rc.d/radiusd restart`
- Test the new users:
  - `sudo radtest bob SEKret localhost 0 testing123`
  - `rad_recv: Access-Accept packet ...`

Changing the Secret

- Edit */usr/local/etc/raddb/clients.conf* and change:
  - `secret = testing123`
- (the only uncommented one) to something like:
  - `secret = `***`eymu5ml`***
- Restart *radiusd* (this is important!)
- Test that the secret has been changed:
  - ***`sudo radtest bob SEKret localhost 0 testing123`***
  - `rad_recv: Access-`**`Reject`**` packet ...`
  - ***`sudo radtest bob SEKret localhost 0 eymu5ml`***
  - `rad_recv: Access-`**`Accept`**` packet ...`

Networking the Service

- Add the following lines to */usr/local/etc/raddb/clients.conf*:

  -
    ```
    client sse {
            ipaddr = 196.200.219.0
            netmask = 24
            secret = newpassword
    }
    ```
- Restart *radiusd*
- Ask your neighbour to test your server, using your own hostname instead of *pcXX*:

  -
    ```
    sudo radtest bob SEKret
    pcXX.sse.ws.afnog.org 0 newpassword
    ```

  -
    ```
    rad_recv: Access-Accept packet ...
    ```

Storing Users in a SQL Database

- The *users* flat file is not scalable:
  - Need to restart *radiusd* whenever users added or changed
  - Difficult to manage with thousands of users
  - Easy to make a mistake which prevents *radiusd* from starting (and therefore breaks your authentication)
  - Difficult to share between multiple servers (for redundancy)
- In production it makes sense to use a SQL database instead, for example MySQL
- The following instructions are based on: http://wiki.freeradius.org/SQL_HOWTO

Starting MySQL Server

- We already installed the MySQL server
- Enable MySQL by adding this line to */etc/rc.conf*:
  - `mysql_enable="YES"`
- Start the MySQL server:
  - `sudo /usr/local/etc/rc.d/mysql-server start`
  - `Starting mysql.`
- If it fails to start, check the error log file:
  - */var/db/mysql/pcXX.sse.ws.afnog.org.err*

Creating MySQL Database

- We need to:
  - Create the database
  - Add a user account and password for *radiusd*
- Run the following commands:
  - ***`mysql -uroot`***
  - `Welcome to the MySQL monitor...`
  - `mysql>` ***`CREATE DATABASE radius;`***
  - `mysql>` ***`GRANT ALL ON radius.* TO radius@localhost IDENTIFIED BY "radpass";`***
  - `mysql>` ***`exit`***

MySQL Passwords

- Our database has no root password!
- To set one:
  - ***mysqladmin -u root password***
  - Now you will need to add the `-p` option to every mysql command
- You can also change the password for the radius user:
  - Run the `GRANT` command again with a different password
  - Edit */usr/local/etc/raddb/sql.conf* and change the `password` setting to match

Linking FreeRADIUS to MySQL

- Create the tables for Radius:
  - `sudo cat /usr/local/etc/raddb/sql/mysql/schema.sql | mysql -u root radius`
  - Should not give any output if successful
- Edit */usr/local/etc/raddb/radiusd.conf* and uncomment the following line:
  - `$INCLUDE  sql.conf`
- Edit */usr/local/etc/raddb/sites-available/default*:
  - Uncomment all the lines that say just "`sql`"
- Restart *radiusd*

Creating a User in MySQL

- Log into MySQL:
  - `$ mysql -u root radius`
- Create a user entry:

  -
    ```
    mysql> INSERT INTO radcheck SET
        UserName = "fred",
        Attribute = "Cleartext-Password",
        Op = ":=", Value = "wilma";
    ```
  - `Query OK, 1 row affected (0.00 sec)`
- Log out of MySQL:
  - `mysql> exit`

Testing the User in MySQL

- Check that we can authenticate as our new user:
  - ***sudo radtest fred wilma 127.0.0.1 0 eymu5ml***

  -
    ```
    Sending Access-Request ...
    User-Name = "fred"
    User-Password = "wilma"
    NAS-IP-Address = 196.200.223.1
    NAS-Port = 0
    ```
  - `rad_recv:` **`Access-Accept`** `packet ...`
- Success!
- If it doesn't work, stop *radiusd* and run:
  - ***/usr/local/sbin/radiusd -X***

User Reply Items in MySQL

- Add an entry into the *radreply* table for each extra *reply item* for Fred:

  -
    ```
    mysql> INSERT INTO radreply SET
      UserName = "fred",
      Attribute = "Framed-IP-Address",
      Op = ":=", Value = "1.2.3.4";
    Query OK, 1 row affected (0.00 sec)
    ```

  -
    ```
    mysql> SELECT * FROM radreply;
    +----+----------+-------------------+----+---------+
    | id | username | attribute         | op | value   |
    +----+----------+-------------------+----+---------+
    |  1 | fred     | Framed-IP-Address | := | 1.2.3.4 |
    +----+----------+-------------------+----+---------+
    1 row in set (0.00 sec)
    ```
- When Fred logs in, this *reply item* will be sent to the NAS.

Group Membership in MySQL

- Add an entry into the *radusergroup* table for each group that Fred is a member of:

  -
    ```
    mysql> INSERT INTO radusergroup SET
      UserName = "fred", GroupName = "users";
    Query OK, 1 row affected (0.00 sec)
    ```

  -
    ```
    mysql> SELECT * FROM radusergroup;
    +----------+-----------+----------+
    | username | groupname | priority |
    +----------+-----------+----------+
    | fred     | users     |        1 |
    +----------+-----------+----------+
    1 row in set (0.00 sec)
    ```
- When Fred logs in, any reply items for the Users group will be sent to the NAS, as well as his own.

Group Reply Items in MySQL

- Add an entry into the radgroupreply table for each extra reply item for the group:

  -
    ```
    mysql> INSERT INTO radgroupreply SET
        GroupName = "users",
        Attribute = "Service-Type",
        Value = "Framed-User", Op = ":=";
    ```

  -
    ```
    Query OK, 1 row affected (0.00 sec)
    ```
- When any user in the Users group logs in, including Fred, this reply item will be sent to the NAS.

Configuring a client

- We have a working RADIUS server!
- What can we do with it?
  - Configure a NAS device or 802.1x switch or access point
  - Will use RADIUS for several examples during the week
- Many services on FreeBSD and Linux use Pluggable Authentication Modules (PAM)
  - Allows you to query many different types of password databases
  - Supports RADIUS!

PAM – Part 1

- Configure the *ssh* service on our machine to authenticate against our RADIUS server
- Services that use PAM have configuration files in */etc/pam.d*
- Edit */etc/pam.d/sshd* and add the following `pam_radius` line, between `pam_ssh` and `pam_unix`:
  - `# auth sufficient pam_ssh.so no_warn try_first_pass`
  - ***`auth sufficient pam_radius.so try_first_pass`***
  - `auth required pam_unix.so no_warn try_first_pass`

AfNOG

PAM – Part 2

- Edit the file */etc/radius.conf*, which probably doesn't exist yet
- Add the following line:
  - `auth 127.0.0.1 eymu5ml 1`
  - `eymu5ml` is the better secret you picked

PAM – Part 3

- Create a user called *fred* (has to exist for *ssh* to allow logins) but with a blank password:
  - `sudo adduser`
  - `Username: fred`
  - `Full name: RADIUS test`
  - `Use password-based authentication? [yes]:`
  - `Use an empty password? (yes/no) [no]: yes`
  - `...`
  - `OK? (yes/no): yes`
  - `adduser: INFO: Successfully added (fred) to the user database.`
  - `Add another user? (yes/no): no`

PAM – Part 4

- Once we've done that we should be able to *ssh* in:
  - `ssh fred@pcXX.sse.ws.afnog.org`
  - `RADIUS Password: wilma`
  - `Last login: Mon May 24 23:11:36 2010 from 196.12.158.76`
- Ask your neighbour to try logging in to your machine as *fred*

Web Management Interface

- daloRADIUS is:
  - “an advanced RADIUS web management application aimed at managing hotspots and general-purpose ISP deployments. It features user management, graphical reporting, accounting, a billing engine and integrates with GoogleMaps for geo-locating.”
- You should find *daloradius-0.9-8.tar.gz* already in your home directory
  - If not, download it from: http://sourceforge.net/projects/daloradius/
- The following instructions based on: http://bit.ly/28Zfy3

Installing PHP for Apache

- Install PHP 5 from ports (to enable the Apache module):
  - `cd /usr/ports/lang/php5`
  - `sudo make install clean`
  - Enable the *Apache* option
- Edit */usr/local/etc/apache22/Includes/php5.conf* and add the following lines:
  - `DirectoryIndex index.php index.html`
  - `AddType application/x-httpd-php .php`
  - `AddType application/x-httpd-php-source .phps`

Installing PHP Extensions

- Install the GD and MySQL PHP extensions:
  - `sudo pkg_add -r mysql50-client t1lib`
  - `cd /usr/ports/lang/php5-extensions`
  - `make install clean`
  - Enable the GD and MYSQL options
  - Don't enable bundled PCRE
- Install PEAR extension:
  - `sudo pkg_add -r pear pear-DB`

Enabling Apache

- Edit /etc/rc.conf and add the following line:
  - `apache22_enable="YES"`
- Start Apache now:
  - `sudo /usr/local/etc/rc.d/apache22 start`
- And check that you can browse to http://localhost

Installing daloRADIUS

- Unpack the *tar.gz* file:
  - ***tar xzvf daloradius-0.9-8.tar.gz***
- Move it to the Apache data directory:
  - ***sudo mv daloradius-0.9-8 /usr/local/www/apache22/data/daloradius***
- Make it writable by the Apache user:
  - ***sudo chown -R www:www /usr/local/www/apache22/data/daloradius***
  - ***sudo chmod u+w /usr/local/www/apache22/data/daloradius/library/daloradius.conf.php***

Configuring daloRADIUS

- Create the database tables:
  - ***`mysql -u root radius < /usr/local/www/apache22/data/daloradius/contrib/db/mysql-daloradius.sql`***
- Now edit */usr/local/www/apache22/data/daloradius/library/daloradius.conf.php* and change the following lines:
  - `$configValues['CONFIG_DB_USER'] =` ***`radius`***
  - `$configValues['CONFIG_DB_PASS'] =` ***`radpass`***
  - `$configValues['CONFIG_DB_TBL_RADUSERGROUP'] = '`***`radusergroup`***`';`

Testing daloRADIUS

- Open daloRADIUS in your browser:
  - http://localhost/daloradius/
- Log in as user ***`administrator`***, password ***`radius`***
- Go to Management → New User
- Note that the *Username Authentication* panel is offset to the right
- We can fix this by editing */usr/local/www/apache22/data/daloradius/css/1.css* and changing:
  - `#contentnorightbar ul {`
  - `        margin:15px 0 16px 20px;`

AfNOG

Adding a User with daloRADIUS

- Go to Management → New User
- Create a new user, for example **`john`**, with a password of your choice
- Create a UNIX user for *john* with useradd, with a blank password, as before
- Try logging in with ssh

Configuring User Information

- The *users* file is a flat text file on the RADIUS server
- Stores authentication and authorization information for all users authenticated with RADIUS
- For each user, you must create an entry that consists of three parts:
  - the user name
  - a list of *check items* (restrictions)
  - a list of *reply items* (settings)
- The SQL database stores these in separate tables

User Information Example

- Franko Clear-Password := 'testing12'

```
Franko	Clear-Password := 'testing12'
	Service-Type = Framed-User,
	Framed-protocol = PPP,
	Framed-IP-Address = 255.255.255.254,
	Framed-IP-Netmask = 255.255.255.255,
	Framed-Routing = None,
	Framed-MTU = 1500
```

- *Clear-Password* is the last check item, because it doesn't end with a comma
- *Framed-MTU* is the last reply item, because it doesn't end with a comma

User Name and Check Items

- *User name* is the first part of each user entry. Consists of up to 63 printable, non-space, ASCII characters. Must be quoted if it contains spaces.
- *Check items* are listed on the first line of a user entry, separated by commas.
  - For an access request to succeed, all check items in the user entry must be matched in the access request.
  - For PAP authentication, the Cleartext-Password attribute must be assigned with the `:=` operator, which always matches

Password Expiration

- To disable logins after a particular date:
  - Specify the date of expiration using the Expiration check item
  - The date must be specified in “Mmm dd yyyy” format
  - Eg. Franko Cleartext-Password := "test12", Expiration := "May 12 2009"
- Try it out!

Reply Items

- Give the NAS information about the user's connection or *authorizations*, e.g.:
  - Whether to use PPP or SLIP
  - Which IP address to assign to the user
- If authentication succeeds:
  - All check items in the user entry are satisfied by the access-request, and
  - The assigned password matches the one supplied by the user
- Then the RADIUS server sends the reply items to the NAS to configure the connection.

Sending Additional Reply Items

- Add the following lines to */usr/local/etc/raddb/users*:

  -
    ```
    Franko Cleartext-Password := 'testing12'
        Service-Type = Framed-User,
        Framed-protocol = PPP,
        Framed-IP-Address = 10.11.12.13,
        Framed-IP-Netmask = 255.255.255.240,
        Framed-Routing = None,
        Framed-MTU = 1500
    ```
- Restart *radiusd* and test with *radtest*:
  - ***sudo radtest Franko testing12 localhost 0 eymu5ml***

  -
    ```
    rad_recv: Access-Accept packet ...
    Service-Type = Framed-User ...
    ```

Shared Secrets

- The entire security of RADIUS relies on the secret!
- From RFC 2865:
  - The secret (password shared between the client and the RADIUS server) SHOULD be at least as large and unguessable as a well-chosen password. It is preferred that the secret be at least 16 octets. This is to ensure a sufficiently large range for the secret to provide protection against exhaustive search attacks. The secret MUST NOT be empty (length 0) since this would allow packets to be trivially forged.
- Long random strings are probably a good idea
  - ***`dd if=/dev/random bs=32 count=1 | sha1`***

What have we achieved?

- We have a free RADIUS server that answers authentication queries using flat files or a MySQL database
- We can deploy new services (for example SMTP AUTH) without having to populate them with user credentials.

What more could we do?

- Query an LDAP, Kerberos or Active Directory database for user authentication
- Add RADIUS authentication to a NAS or Access Point
- Replicate the password database across multiple machines for redundancy
- Restrict logins based on time of day, NAS IP, etc.
- Generate accounting data, so that we could bill for timed access to resources
  - E.g. at a wireless hotspot or a hotel

Where to Get Help

- The FreeRADIUS website
  - http://www.freeradius.org/
- FreeBSD PAM module
  - http://www.freebsd.org/doc/en/articles/pam/
- PAM RADIUS man page
  - http://www.freebsd.org/cgi/man.cgi?query=pam_radius&sektion=8
- AfNOG Mailing List
  - http://www.afnog.org/mailinglist.html
  - Please subscribe to this list!

FIN

Ack?