RADIUS and FreeRADIUS

Chris Wilson, Aptivate Ltd.
Presented at AfNOG 2014

Based on “FreeRADIUS Install and Configuration” by Frank A. Kuse

Download this presentation at:
http://afnog.github.io/sse/radius

Ingredients

- Theory
  - What is RADIUS
  - Why use RADIUS
  - How RADIUS works
  - User databases
  - Attributes
- Practical
  - Installing FreeRADIUS
  - Adding RADIUS users
  - Authenticating services that use PAM

What is RADIUS?

- Remote Authentication Dial In User Service
- Authentication
  - “Who are you?”
- Authorization
  - “What services am I allowed to give you?”
- Accounting
  - “What did you do with my services while you were using them?”

Why RADIUS?

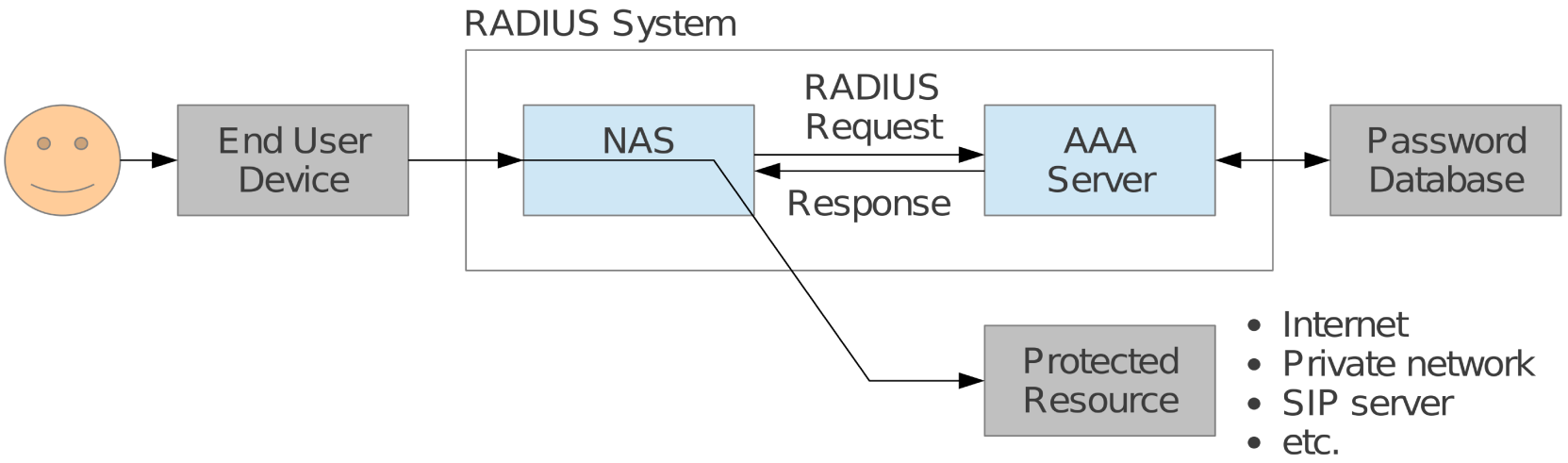
- What are the alternatives?
  - LDAP, Kerberos, Active Directory
- Advantages of RADIUS:
  - Lightweight and efficient
  - Supported by many clients, e.g. 802.1x, switches and routers
- Disadvantages of RADIUS:
  - Limited attribute set, limited use for desktop authentication

How does RADIUS work?

- Authentication
  - Password authentication, plain text and hashed
  - Lookup in various user databases: passwd, SQL, text
- Authorization
  - Using a set of rules or other templates
- Accounting
  - Measuring, communicating and recording resources accessed by user
- See Wikipedia for list of RFCs

RADIUS Architecture

- RADIUS protocol is between NAS and AAA server
- NAS controls access to protected resource



What does RADIUS do?

- NAS sends an Authentication-Request to AAA server
  - user name
  - password hashed with secret shared
  - some client specific information
- AAA server receives an Authentication-Request
  - consults password databases:
  - looks up the username and client-specific info
  - retrieves unhashed password, and other Check Items
  - hashes and compares with request contents
  - sends an Access-Accept or Access-Reject packet

Why do we need RADIUS?

- Many services require password authentication!
- Users don't want to remember many passwords
- Easier to change password regularly or if compromised
- Easier to secure a single password database
- Enables user-password auth with 802.1x
- Alternative to TACACS for network equipment
- Used for PPP authentication in ISPs (PAP/CHAP)

RADIUS message types

- Access-Request
- Access-Challenge
- Access-Accept
- Access-Reject
- Accounting-Request
- Accounting-Response
- Status-Server (experimental)
- Status-Client (experimental)

RADIUS attributes

- Name=Value
  - User-Name
  - User-Password
  - NAS-IP-Address
  - NAS-Port
  - Service-Type
  - NAS-Identifier
  - Framed-Protocol
  - Vendor-Specific
  - Calling-Station-ID
  - Called-Station-Id

RADIUS users database (file)

- Flat text file
  - Easy to understand and edit
  - Alternatives include Kerberos, LDAP and SQL
- Each user entry has three parts:
  - Username
  - List of check items (requirements)
  - List of reply items (assignments)

User entry example

```
Franko      Password = 'testing12'
            Service-Type = Frame-User,
            Framed-protocol = PPP,
            Framed-IP-Address = 192.168.1.4
            Framed-IP-Netmask = 255.255.255.0
```

- Username is Franko (case sensitive!)
- Check items (first line, all must match Access-Req):
  - password = testing12
- Reply items (indented lines):
  - Service-Type, Framed-IP-Address...

User name and check items

- Username
  - First part of each user entry
  - Up to 63 printable, non-space, ASCII characters
- Check Items
  - Listed on the first line of a user entry, after username
  - Multiple items are separated by commas
  - Entry only matches if all check items are present in the Access-Request and match
  - *Fall-Through* = *Yes* allows server to try other entries
- First line (user name + check items) must not exceed 255 characters.

Operators in user entries

- The “=” and “==” operators mean different things in *check items* and *reply items*!
- In check items:
  - Use “=” for server configuration attributes (Password, Auth-Type)
    - Sets the value if not already set (set without override)
  - Use “==” for RADIUS protocol attributes
    - True if value is present and has the same value, never sets
- In reply items:
  - Use “=” for RADIUS protocol attributes
  - Do not use “==”, it is never valid

The Auth-Type check item

- Used to specify where (how) to lookup the password:
  - Local (in the users file)
  - System (query the OS, /etc/shadow or PAM)
  - SecurID
- Defaults to Local
- Example:

```
Franko      Auth-Type = Local, Password = 'test123'
```

Password expiration

- Disable logins after a particular date
- Use the *Expiration* check item:

```
Franko Password="test12", Expiration="May 12 2009"
```

- Date must be specified in "Mmm dd yyyy" format!
- Use the *Password-Warning* check item to warn the user *before* their password expires:

```
VALUE    Server-Config  Password-Expiration 30
VALUE    Server-Config  Password-Warning    5
```

Checking the NAS IP address and port

- NAS-IP-Address check item
  - Matches a particular NAS (by IP address)
  - Will only match if the user connected to (Access-Request came from) that specific NAS.
- NAS-Port-Type check item
  - Will only match if the NAS reports that the user connected to a specify the type of port
  - Options include: Async, Sync, ISDN
- NAS-Port check item
  - Will only match if the NAS reports that the user connected to a specific port (ethernet or serial)

Reply items

- If all check items in the user entry are satisfied by the access-request, then:
- Radius server sends an Access-Accept packet to the NAS, containing the reply items
- Gives information to the NAS about the user
  - For example, which IP address to assign to them

Reply items

- Service Type
  - Must be specified
  - Login-User → User connects via telnet, rlogin
  - Framed-User → User uses PPP or SLIP for connection
  - Outbound-User → User uses telnet for outbound connections.
- Framed-User is by far the most used now
- Simple example:

```
Franko          Auth-Type = System
                Service-Type = Framed-User
```

The Service-Type reply item

- Service Type
  - Must be specified
  - Login-User → User connects via telnet, rlogin
  - Framed-User → User uses PPP or SLIP for connection
  - Outbound-User → User uses telnet for outbound connections.
- Framed-User is by far the most used now
- Framed-User requires a Framed-Protocol:

```
Franko  Auth-Type = System
        Service-Type = Framed-User
        Framed-Protocol = PPP
```

The Framed-IP-Address reply item

- Specifies the user's IP address to the NAS
- Set to 255.255.255.255 to force the NAS to negotiate the address with the end-node (dial-in user)
- Set to 255.255.255.254, or leave out, to force the NAS to assign an IP address to the dial-in user from the assigned address pool

```
Franko  Auth-Type = System
        Service-Type = Framed-User
        Framed-Protocol = PPP
        Framed-IP-Address = 192.168.1.4
```

Netmask and Route reply items

- Use *Framed-IP-Netmask* to specify a netmask for the user's IP address
  - The default subnet mask is 255.255.255.255
- Use *Framed-Route* to add a route to NAS routing table when service to the user begins
  - Three pieces of information are required:
    - the destination IP address
    - gateway IP address
    - metric
  - For example:
    - `Framed-Route = “196.200.219.0 196.200.219.4 1”`

Accounting records

- FreeRADIUS writes to its Detail log file
- Typically *Start* and *Stop* accounting records

```
Tue May 12 14:12:14 2009
        Acct-Session-Id = “25000005”
        User-Name = “franko”
        NAS-IP-Address = 196.200.219.2
        NAS-Port = 1
        NAS-Port-Type = Async
        Acct-Status-Type = Start
        Acct-Authentic = RADIUS
        Service-Type = Login-User
        Login-Service = Telnet
        Login-IP-Host = 196.200.219.254
        Acct-Delay-Time = 0
        Timestamp = 838763356
```

Accounting attributes

- Acct-Status-Type attribute
  - indicates whether the record was sent when the connection began (Start) or when it ended (Stop)
- Acct-Session-Id attribute
  - ties the Start and Stop records together, indicating that it's the same session

What is FreeRADIUS?

- The premier open source RADIUS server
- Similar to Livingston RADIUS 2.0
- Many additional features
- Free!

Practical exercise overview

- Build and install FreeRADIUS
- Configure and start FreeRADIUS
- Test authentication using FreeRADIUS
- Convert a service to authenticate using RADIUS

Installing FreeRADIUS

- Where in the ports collection is FreeRADIUS?
  - cd /usr/ports/net/freeradius
  - sudo make install
- Select any options you might need (none)
- Watch it build and install:
  - init script in `/usr/local/etc/rc.d/radiusd`
  - config files in `/usr/local/etc/raddb`
- Add to /etc/rc.conf: `radiusd_enable="YES"`
- Run `sudo /usr/local/etc/rc.d/radiusd start`

Configuring and debugging

- You should review the configuration files carefully
- Debugging mode is extremely useful:
  - `sudo /usr/local/etc/rc.d/radiusd stop`
  - `sudo radiusd -x`
- Output should end with:
  - `Ready to process requests.`
- Server is now running in debugging mode
  - Leave it running, and open another window/session on the server to run more commands

Testing the default configuration

- FreeRADIUS should now respond to RADIUS requests
- Test by running:
  - `sudo radtest test test localhost 0 testing123`
  - What happens?
- Try a local user that does exist, with password:
  - `sudo radtest afnog afnog localhost 0 testing123`
  - What happens?
- You should see the server receive the access-request and respond with an access-reject *in both cases*

Completing Unix authentication

- Unix authentication is not working!
- We don't know why!
- Look carefully at the debug output
  - `rlm_unix: [afnog]: invalid password`
  - This means that the account has an invalid password, OR the password field could not be read
- This is a permissions issue
  - Change `user = freeradius` to `user = root` in */usr/local/etc/raddb/radiusd.conf*
  - Restart `radiusd -x` and try again
  - `rad_recv: Access-Accept packet ...`

Change the default shared secret!

- We've been using the default shared secret, `testing123`
  - Not very secret, so let's change it!
- Edit */usr/local/etc/raddb/clients.conf*
  - Find the section `client 127.0.0.1`
  - Change `secret = testing123` to `secret = afnog`
- The secret can be up to 31 characters in length
- For monitoring purposes, we need the same secret on all the machine and that is “afnog”
- Restart `radiusd -x` and test with the new secret

Secret (digression)

- 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.
- How to generate a new, secure random key:
  - `dd if=/dev/random bs=16 count=1 | base64`

Creating RADIUS users

- So far we have only shared our Unix password database using RADIUS
- Add this line to */usr/local/etc/raddb/users*:
  - `john  Password == "Smith"`
  - Add it **early** in the file, before the first DEFAULT line
- Restart `radiusd -x`
- Test using the radtest command:
  - `sudo radtest john Smith localhost 0 afnog`
  - `rad_recv: Access-Accept packet …`
  - Success!

Configuring a client

- Now that we have the server working we can configure a client to query the server
- We could configure a NAS device, if we had one
- Many authenticated services on FreeBSD (and Linux) use PAM to authenticate users
  - Pluggable Authentication Modules
  - Allows any service to query many different password databases
  - By default just queries the system password database, */etc/master.passwd*
  - The pam_radius module queries a RADIUS server (AAA) for authentication

Using PAM with RADIUS (part 1)

- Configure the SSH service on our machine to authenticate against our RADIUS server
  - Keep a root shell open, in case you break it!
  - Edit */etc/pam.d/sshd*
  - Find the line: `auth required pam_unix.so`
  - Add another line before it:
  - `auth sufficient pam_radius.so`
- Try connecting with SSH to your machine
  - Try with user `afnog`
  - Do you notice any difference in the password prompt?
  - Now try with user `john` – this will fail, but why?

Using PAM with RADIUS (part 2)

- What's wrong with authenticating as RADIUS user?
  - `tail /var/log/auth.log` may give you a clue
  - The configuration file */etc/radius.conf* is missing
  - PAM doesn't know which RADIUS server to use, or with what shared secret
- Create the file */etc/radius.conf*, adding this line:
  - `auth 127.0.0.1 afnog`
- SSH requires that the user exists on the local system
  - Otherwise you'll see: `Invalid user john from ...`
  - Create the user by running: `pw useradd john`
- Now try again with SSH

What have we achieved?

- FreeBSD RADIUS server answers authentication requests:
  - Unix password files/database
  - Flat text file (`users` file)
- SSH login authentication using RADIUS passwords
- We can deploy new services without having to create separate password databases

What more could we do?

- Store credentials in:
  - a database (MySQL, PostgreSQL)
  - LDAP
  - Kerberos
- Integrate with network access control (802.1x)
- Generate accounting data
  - so that we could bill for timed access to resources
  - for example a wireless hotspot or a hotel network
- Generate reports from accounting data

Bibliography

- FreeRADIUS website
  - http://www.freeradius.org/
- FreeBSD PAM
  - http://www.freebsd.org/doc/en_US.ISO8859-1/articles/pam/index.html
- PAM RADIUS man page
  - http://www.freebsd.org/cgi/man.cgi?query=pam_radius&sektion=8