Overview

Introduction

This solution employs a router/server computer with two network interfaces, one is a network uplink, the other connects to the NASes (wifi access points) with their respective supplicants.

Users authenticate using a username and password, and verify the identity of the RADIUS server using a certificate presented by the server (see more information under supplicant configuration).

As mentioned in the General information page, the NASes available do not support sending RADIUS accounting packets, they are only able to authenticate users against the response from a RADIUS server. This solution uses 802.1X to authenticate users and let them into the wireless network, and then Shorewall to perform the logging/accounting work. FreeRADIUS, along with a couple of scripts, logs the MAC address of all devices that connect along with the username they authenticated as, this information can be used to match information in Shorewall's logs to a user session. For improved security the firewall policy will be to disallow all connections except those originating from known IP addresses of known wifi clients. A script, shwl_add.sh, gets run by FreeRADIUS upon successful authentication of a supplicant, which runs an ARP scan to find the IP address of the device with the MAC address specified by FreeRADIUS, it then adds the IP address to shorewall's "whitelist", logs the event along with some useful information to a MySQL database and writes the current timestamp to a text file. In Access-Accept packets the Session-Timeout and Termination-Action attributes are sent, informing the NAS that after the specified amount of time the supplicant needs to repeat the authentication process or be disconnected. When the supplicant repeats authentication the mentioned script detects that the supplicant is already known and simply logs the event to MySQL and updates the timestamp in the text file. Another script, shwl_del.sh, is run by crontab at a regular interval, which goes through all the IP addresses present in shorewall's "whitelist" and checks the timestamp of the corresponding supplicant's last authentication to FreeRADIUS in the corresponding text file. If more than the specified amount of time has passed, this script assumes that the supplicant has disconnected from the wifi network and removes its IP from shorewall's "whitelist", logs the event to MySQL, and deletes the related text files.

A python script, script_launcher.py, serves as an intermediary script that is called by FreeRADIUS using its Rlm_python module and in turn launches shwl_add.sh.

The database backend containing user credentials for FreeRADIUS is MySQL. The passwords are stored in MySQL in NT hashes. Adding/deleting users can be accomplished by SQL queries which can be included in site specific custom user management scripts or as hooks to the standard 'adduser' and 'deluser' utilities (out of the scope of this guide). Updating of passwords is accomplished by a script, pam_to_mysql_update.sh, that gets run by the libpam-script PAM module during PAM stack execution, and updates the password in the MySQL database according to the same password the user chose for their system user account. An entry with the username field already filled and matching the system user account username needs to be already present in the MySQL database. Commands that expire or disable a user's system user account (or password) without deleting it (such as passwd -l) will not cause the credentials in the MySQL database to be disabled, thus it is necessary to take care (perhaps with a site specific lock user script) to also invalidate the same.

Shorewall dynamic zones are used to achieve the dynamic change of firewall rules set for the IP addresses added/removed to the "whitelist". The rules for the normal zone concerning the network connecting to the NASes/supplicants disallows everything, except what's needed for the RADIUS conversation between the NASes and FreeRADIUS (and perhaps, HTTP access to the NAS web interfaces). A dynamic zone is declared under the normal zone, whose rules allow network access, with logging. The 'shorewall add <dynamic_zone_name> <ip_address>' and 'shorewall delete <dynamic_zone_name> <ip_address>' commands can then be used by the shwl_*.sh scripts to change the applicable rules for the specified source IP address.

Sudo is installed and configured as it is required for some of the mentioned scripts to run commands as root or as a different user.

A package containing the mentioned scripts, as well as the empty MySQL schema for the database used by some of them, is attached to the Installation page.

Scripts

Common MySQL database schema

A MySQL schema is contained in the shwl_add_shwl_del_pmu.sql file in the attached archive. It is used by the shwl_add.sh, shwl_del.sh and pam_to_mysql_update.sh scripts to log events and errors. It contains a single table named 'event_log', it's format is as follows:

log_id	device_username	device_ip	device_mac	rad_attr_NAS-IP-Address	rad_attr_NAS-Port	rad_attr_Called-Station-Id	rad_attr_NAS-Identifier	rad_attr_Framed-MTU	rad_attr_NAS-Port-Type	rad_a Type
A unique identifier for the entry, auto-generated by MySQL	See the below sections on each script for all possible values	See the below sections on each script for all possible values	See the below sections on each script for all possible values	The value of the NAS-IP-Address RADIUS attribute, when applicable	The value of the NAS-Port RADIUS attribute, when applicable	The value of the Called-Station-Id RADIUS attribute, when applicable	The value of the NAS-Identifier RADIUS attribute, when applicable	The value of the Framed-MTU RADIUS attribute, when applicable	The value of the NAS-Port-Type RADIUS attribute, when applicable	The valu EAP-Ty RADIUS when ap

NOTE: The 'log_time' column indicates the time when the MySQL entry was logged (i.e. all the scripts use the MySQL NOW() function in the field for this column in all their queries), whereas the rad_attr_Event-Timestamp column contains the content of the Event-Timestamp RADIUS attribute, when applicable, which is expected to contain the timestamp of when the RADIUS server received the request.

NOTE: This database is used only by the above mentioned scripts. One more MySQL database is expected to be running on the server for FreeRADIUS, containing the user database, which will also be accessed by the pam_to_mysql_update.sh script to update user passwords stored in there.

shwl_add.sh

This script is intended to be run by script_launcher.py (below mentioned).

It reads 10 lines on stdin, as follows:

1	Username (RADIUS attribute: User-Name)
2	Supplicant MAC address (RADIUS attribute: Calling-Station-Id)
3	RADIUS attribute: NAS-IP-Address
4	RADIUS attribute: NAS-Port
5	RADIUS attribute: Called-Station-Id
6	RADIUS attribute: NAS-Identifier
7	RADIUS attribute: Framed-MTU
8	RADIUS attribute: NAS-Port-Type
9	RADIUS attribute: EAP-Type
10	RADIUS attribute: Event-Timestamp

The supplicant MAC address is used in the script, the remaining values are simply logged into the MySQL database. Some sanity check is performed on the username before the script continues. The supplicant MAC address is processed so as to obtain it in both of the following formats regardless of the format the NAS specified it in: filename friendly version: 0123456789ab, normal version: 01:23:45:67:89:ab. The supported input formats are: 01-23-45-67-89-ab, 01:23:45:67:89:ab, 0123456789ab, all case in-sensitive. The script then checks if a file named as the filename friendly version of the supplicant MAC address already exists (referred to by the script as an IP file). If it does not, the script runs an ARP scan on the configured network interface and in the resulting table, looks for the IP address matching the specified supplicant MAC address. In case no such device is found, it repeats the scan a configurable amount of times at a configurable interval before giving up. Once a matching IP is found a basic amount of sanity check is performed on the scan results (e.g. to detect multiple MACs being used by the found IP, or such), and then the event is logged into MySQL with "connect" mentioned in the 'event' column, the found IP is added to the configured shorewall dynamic zone, and a file named as the filename friendly version of the supplicant MAC address is created containing the device's IP, and a file named as the device's IP (referred to by the script as a timestamp file) is created containing the present timestamp. In case adding the IP address to the shorewall dynamic zone is not successful, the script waits a configurable amount of time, and then attempts a second time, after which it gives up (it was observed that it might happen that FreeRADIUS starts earlier in the boot process than Shorewall, and if this script is started during that timeframe the shorewall add command fails). In case the IP file is already present, the script simply logs the event to MySQL with "re-auth" mentioned in the 'event' column and re-writes the timestamp file with the present timestamp. In case an error is encountered, the script logs the error to MySQL mentioning "err-add-N" (where N is the error number) in the 'event' column, populating the remaining columns with their respective information as may be available at the time of the error, and exits immediately, returning the error number as exit code. In case access is available to the script's stdout and stderr, a description of the error message is also printed (and the script is quite verbose about what's happening), in case not, it is possible to look in the script's code for calls to the shwl_add_error_message_close() function, identify the call where the error number in question is passed to the function, and the error description can be found in the same function call. The configurable options can be found at the top of the script. The username is logged into the 'device_username' column, the IP address found during the scan is logged in the 'device_ip' column, the above mentioned normal version of the supplicant MAC address is logged in the 'device_mac' column, and the remaining RADIUS attributes are logged in the columns with the respective names.

shwl_del.sh

This script is intended to be run by crontab at a regular interval.

This script loops through all the IP addresses present in the configured shorewall dynamic zone, reads their corresponding timestamp file (see shwl_add.sh description) and checks whether the configured amount of time has elapsed since. If it has, it logs the event to MySQL specifying "expire" in the 'event' column, removes the IP from the shorewall dynamic zone, deletes the timestamp file and searches for files (although there should be only one) containing the IP address (to find the IP file, see shwl_add.sh description) and deletes them. In the unexpected case that no timestamp file is found for a given IP or it does not contain a valid timestamp, the same actions are taken as when the configured amount of time has elapsed, but "untracked" is mentioned in the 'event' column instead of "expire". In case an error is encountered, the script logs the error to MySQL mentioning "err-del-N" (where N is the error number) in the 'event' column, populating the remaining columns with their respective information as may be available at the time of the error, but continues execution. At the end, it returns the error number as exit code, or, in case there were multiple errors, 127. In case access is available to the script's stdout and stderr, a description of the error message is also printed (and the script is quite verbose about what's happening), in case not, it is possible to look in the script's code for calls to the shwl_del_error_message() function, identify the call where the error number in question is passed to the function, and the error description can be found in the same function call. The configurable options can be found at the top of the script. The IP address being processed is logged in the 'device_ip' column, a comma separated list of the IP files is logged in the 'device_mac' column, or the text "/// no-ip-files ///" in case no IP files were present, or the text "/// err-del-N ///" (where N is the error number) in case an error occurred while searching for the file(s), and the rad_attr_* as well as the 'device_username' columns are left empty. The text "/// n/a ///" may be present in the 'device_ip' and 'device_mac' columns if this information is not applicable.

script_launcher.py

This script is intended to be called by the FreeRADIUS Rlm_python module, which, in the intended configuration, calls its post_auth(attr) function passing a tuple of tuples containing the relevant RADIUS attributes and their values as the 'attr' argument.

This script looks for the values of the following items in the tuple of tuples:

User-Name	Mandatory
Calling-Station-Id	Mandatory
NAS-IP-Address	Mandatory
NAS-Port	Optional
Called-Station-Id	Mandatory
NAS-Identifier	Optional
Framed-MTU	Optional
NAS-Port-Type	Optional
EAP-Type	Optional
Event-Timestamp	Mandatory

The script executes the configured command in the background and then writes to the new process's stdin the values of the above attributes, one attribute on each line, in the order shown in the table. It then exits, leaving the new process running. If one of the RADIUS attributes mentioned above as "Optional" is missing, the script writes "None" in its place. If one of the "Mandatory" ones is missing an error occurs. In case an error is encountered, the script exits immediately, returning radiusd.RLM_MODULE_FAIL. (This means if an error occurs after the new process has been launched, e.g. because a mandatory RADIUS attribute is missing, the script exits leaving the new process running, doing nothing, waiting for its stdin to be populated). If no error occurs, the script returns radiusd.RLM_MODULE_OK at the end. The configurable options can be found at the top of the script.

pam_to_mysql_update.sh

This script is intended to be run by the libpam-script PAM module during the PAM auth and PAM password stacks execution. In the intended configuration, the libpam-script module checks the script's exit code and reports failure back to the PAM stack, causing the PAM operation to fail, in case an error occurred in the script.

It reads the following environment variables (set by libpam-script):

PAM_USER	The system user for whom the PAM operation is running
PAM_AUTHTOK	The user's password (in case of a password change operation, the new password)
PAM_SERVICE	The service that invoked PAM (e.g. sshd when the user is attempting to log in through SSH)

This script makes use of two MySQL databases, the one containing the above mentioned 'event_log' table for logging errors, and FreeRADIUS's database for updating passwords.

This script checks first whether it is running as root (the PAM stack does not necessarily run as root, but as the user as which the service that invoked it is running). If it is, it proceeds to encrypt the user's password (${PAM_AUTHTOK}) as an NT hash and update it in the 'value' column of the configured table in FreeRADIUS's MySQL database for the entry where the value of the 'username' column matches the user's username (${PAM_USER}). Due to the nature of the SQL query used, there needs to already be an entry in the table containing the matching username in the 'username' column. If it is not running as root, it uses a workaround to escalate its privileges. It uses the available credentials to SSH into localhost as the user in question, causing the SSH daemon to run a new instance of the PAM stack to verify the user's credentials (and sshd runs the PAM stack as root) and thus a new instance of the script, as root. Before doing so the script makes sure that the PAM stack has not already now been invoked by sshd, in order to avoid, in case sshd should ever decide to run the PAM stack as a non-root user, that an infinite loop occurs spawning endless processes of this script, sshd, PAM, etc.

In case an error is encountered, the script logs the error to the configured table in MySQL, in the 'event_log' table, mentioning "err-pmu-N" (where N is the error number) in the 'event' column, populating the 'device_username' column with the user's username, leaving the other columns empty, and exits immediately, returning the error number as exit code. The error is logged to MySQL only if the script is running as root. In case access is available to the script's stdout and stderr, a description of the error message is also printed (and the script is quite verbose about what's happening if cfg_verbose is set to 1), in case not, it is possible to look in the script's code for calls to the pam_to_mysql_update_error_message_close() function, identify the call where the error number in question is passed to the function, and the error description can be found in the same function call. The configurable options can be found at the top of the script.

This script has been tested with the 'passwd' system utility and the GNOME User Accounts applet, for the PAM password stack. It has been tested with 'su', GNOME desktop and sshd for the PAM auth stack.

In the intended PAM/libpam-script configuration, in the case of the script running SSH to start a second instance as root, in the case the second instance (running as root) fails and returns an error exit code, libpam-script will report failure to the PAM stack, causing the authentication to fail, thus the SSH login to fail, and the ssh command that was launched in the first instance of the script, resulting finally in the first instance of the script to also fail, the first instance of libpam-script, and thus the first PAM stack. Setting cfg_verbose=1 will cause the script's verbose output to appear on screen in cases where a service invoking the PAM stack allows, e.g. when running the 'su' command. During PAM password stack execution, the libpam-script module also shows a prompt "Current password:", which is visible when using command line utilities such as 'passwd', and is irrelevant to our purpose, this can be safely ignored and it is sufficient to press enter.

Sources

https://wiki.freeradius.org/guide/Basic-configuration-HOWTO

https://wiki.freeradius.org/guide/SQL-HOWTO-for-freeradius-3.x-on-Debian-Ubuntu

https://wiki.freeradius.org/modules/Rlm_python

https://wiki.freeradius.org/config/Certificates

http://deployingradius.com/documents/configuration/certificates.html

http://deployingradius.com/documents/protocols/compatibility.html