AAA feature for SONiC

Overview 2

AAA Protocols 2

Requirements 3

Assumptions 4

Limitations 4

References 4

AAA Design 5

AAA Component Interactions 5

High level flow of user authentication & Authorization 5

Authentication 6

Fallback mechanism for Authentication / Local Authentication 8

Privilege level to User Role mapping 9

Categorization of Shell Commands 9

Authorization 11

Role based Authorization 11

TACACS+ based Authorization 11

AAA Configuration Model 12

CLI for AAA 13

TACACS+ Configuration Model 13

Global Parameters 13

Server Configuration 14

CLI for TACACS+ 15

# Overview

Authentication, Authorization, and Accounting (AAA) is a framework for intelligently controlling access to the system. AAA is designed to enable network operators to dynamically control effective network management and security.

Authentication service provides a way to identify the user, typically by having the user provide his/her credentials (a valid username and password) before access to the system is granted for that user. The process of authentication is based on each user having a unique set of criteria for gaining access. The AAA server compares user authentication credentials with those stored in the database corresponding to that user. If the credentials match, the user is granted access to the network. If the credentials are at variance, authentication fails and network access is denied.

Authorization service determines whether the user has the authority to perform specified action on the system based on the privilege level configured for the user. Simply put, Authorization is the process of enforcing policies determining what the user is allowed to perform on the system. Usually, Authorization occurs within the context of Authentication for the given user. Once the user is authenticated, the user may be authorized for different activities he/she tries to execute on the system.

Accounting service provides a mechanism to collect information and logs for auditing & reporting purposes. Accounting is carried out by logging of session statistics and usage information and is used for authorization control, billing, trend analysis, resource utilization, capacity planning activities, etc.

Authentication, Authorization, and Accounting services are often provided by dedicated AAA Server that provides these functions.

# AAA Protocols

TACACS+ and RADIUS are the two best known types of AAA Protocols.

There are inherent differences between TACACS+ and RADIUS that make them suitable for particular type of different situations. TACACS+ is a proprietary protocol developed by Cisco, while RADIUS is IETF and RFC standard based. Another difference is that TACAS+ operates in TCP environment while RADIUS operates in UDP environment. Further, TACACS+ can control the authorization level of users, while RADIUS cannot.

*SONiC provides AAA functionality by implementing TACACS+ protocol.*

*Support for RADIUS protocols is not proposed for SONiC at this time and therefore, is beyond the scope of this document.*

TACACS+ (Terminal Access Controller Access Control System Plus) provides Authentication, Authorization and Accounting services, used most commonly for administration access to network devices like routers, switches and terminal servers.

TACACS+ separates out the authorization functionality, so it enables additional flexibility and granular access controls to control which user can run what commands on specified devices. Each command entered by the user is sent to TACACS+ server for Authorization, which then checks the command against an authorized list of commands for the specific user or group. TACACS+ can define policies based on user, device type, location or time of day, etc.

TACACS+ can also be used for authorization and accounting using a different method like Kerberos for authentication. This is possible because TACACS+ separates authentication and authorization (unlike RADIUS).

# Requirements

1. AAA can be enabled / disabled by the user. When disabled, the local server is the default mechanism for authentication & authorization.
2. When multiple TACACS+ servers are configured, user shall explicitly specify the priority of the server. The lower numerical value indicates higher priority within the server group.
3. When multiple TACACS+ servers are configured, the user configured server priority determines the order in which the servers are attempted for authentication and/or authorization. The subsequent servers are attempted only when previous server times out.
4. All user login through console, management interface & in-band must get authenticated by TACACS+ server or local authentication mechanism based on the configuration knob that determines the order of preference.
5. Separate configuration knobs may be required to distinguish order of preference for authentication for user login through console and other uses.
6. Command authorization shall also follow same order of preference for the server as Authentication
7. AAA must have a default fallback server mechanism (such as local server) for both authentication & authorization. A configuration knob shall be provided to disable this fallback mechanism. The default is on.
8. The fallback mechanism shall get invoked either when no TACACS+ servers are configured or when none of the servers are reachable
9. Accounting is NOT a requirement at this time
10. Authorization of command via REST is a desired feature, but not a must.

# Assumptions

1. The CLI implementation is a wrapper on top of RESTful interface which in-turn interfaces with Configuration Manger as described in SONiC Configuration Model design
2. The users configured on TACACS+ server are remote users, need not be configured locally on the switch
3. Local authentication is the default fallback authentication method
4. TACACS+ server should be able to provide role-based and user-based authorization information for a user
5. REST logic will run in the same docker contest as the configManager.
6. When user login to the box, user will land at Linux shell (which has RBAC capabilities built into it. Please refer to [Categorization of Shell Command](#_Categorization_of_Shell) for details.
7. Current python-based CLI will need to be migrated to the new sonic-cli being developed as part of SONiC Configuration Model

# Limitations

1. Accounting is not supported by the design proposed in this document, other than basic accounting related to session duration
2. Support for Role-based based Authorization will be implemented first. TACACS+ server based Authorization is best effort for implementation at this time (depending on use case requirement)

# References

1. SONiC Configuration Management
2. Open Source *pam\_tacplus*: <https://github.com/jeroennijhof/pam_tacplus>

# AAA Design

## AAA Component Interactions

The AAA component in SONiC intercepts the commands from REST Interface and/or Linux Shell before Configuration Manager processes them as shown in the following diagram:



AAA services are implemented leveraging existing Open Source components from *pam\_tacplus* implementation (see References). The PAM libraries will run in the same instance as host Linux OS in SONiC.

## High level flow of user authentication & Authorization

A high level flow of user, AAA components & TACACS+ server interactions during authentication and authorization process is shown in the following diagram.



## Authentication

A detailed flow of user authentication mechanism using TACACS+ server and/or fallback mechanism to local authentication is shown in the following diagram:



The sequence of steps involved in the authentication process is

1. User SSH to the switch, given login prompt
2. SSHd accepts connection request and processes it by asking for username and password credentials
3. SSHd checks user SSH keys against authorized SSH keys. PAM function pam\_authenticate( ) is invoked to verify if the user supplied username/password pair is valid
4. PAM loads pam\_tacplus( ) configuration module as defined in /etc/pam.d/sshd, and calls pam\_sm\_authenticate( )
5. This function sends encrypted packet to the TACACS+ to validate the user credentials. The subsequent steps fall under the following scenarios:
   1. Scenario-1: TACACS+ server is reachable and has the user configured with the same credentials as those provided by the user – PAM libraries would return PAM\_SUCCESS to the application (SSHd)
   2. Scenario-2: TACACS+ server is unreachable
      1. PAM libraries would continue to next configured TACACS+ server configured, the authentication process continues at step-5
      2. When NONE of the configured TACACS+ server is reachable, PAM libraries would invoke [Fallback mechanism](#_Fallback_mechanism_for)
   3. Scenario-3: TACACS+ server is reachable, but does not have user account or has different user credentials – the TACACS+ server would return Authentication Failure message. Based on this, the PAM libraries would return PAM\_AUTH\_ERR to the application (SSHd).
      1. User is asked to enter his password again. Maximum of 3 password retries are allowed.
6. Once user is successfully authenticated, the PAM libraries fetch information about the privilege level associated with the user. Based on the privilege level association (as defined in [Privilege to user Role mapping](#_Privilege_to_User)), the user is provided access to Role Based Access Controlled shell.
7. Before user gets the shell, PAM calls a function pam\_sm\_open\_session( ). This results in sending an accounting START packet to the server. Among other things it contains the user login and the time session started
8. When the user logs out, pam\_sm\_close\_session( ) sends STOP packet to the server indicating the whole session is closed.

### Fallback mechanism for Authentication / Local Authentication

Fallback mechanism for Authentication (aka Local Authentication) is the authentication mechanism that automatically gets invoked when NONE of the TACACS+ servers are reachable when user attempts to get authenticated.

The sequence of steps involved in the Fallback mechanism for Authentication is

1. The steps 1 through 4 as explained in [User Authentication procedure](#_User_Authentication_procedure) will remain the same
2. PAM libraries would invoke user authentication using /etc/passwd and /etc/shadow files. When the user is successfully authenticated PAM\_SUCCESS would be returned to the application. Otherwise, PAM\_AUTH\_ERR would be returned.

NOTE:

*Please note that when TACACS+ server sends failure message for a user authentication request, the Fallback mechanism for Authentication DOES NOT get triggered.*

## Privilege level to User Role mapping

The mapping between privilege level & user role are defined in the following table:

|  |  |  |
| --- | --- | --- |
| User Role | TACACS+ Privilege Level | Description |
| admin | 15 | Has access to all commands including capability to reboot, user add/delete |
| netops | 14 | Has access to configure commands except user add/delete |
| operator | 1 | Has access to monitor commands |

Role with higher privilege level has access to all the capabilities under a lower privilege level, plus additional capabilities.

If the privilege level of the user is not among the well defined privilege levels as explained in this section, then the user is put onto the next lowest privilege level. For example, if user privilege level returned from TACACS+ server is between 2-13, the user is put onto a privilege level of 1.

## Categorization of Shell Commands

The Role Based Access Controlled shell (RBAC shell) offers various operations through shell commands that the user is allowed to perform on the switch based on the privilege level of the user.

These shell commands can be categorized into different sets, as shown below. Each set of commands are further associated with appropriate privilege levels.



*It is imperative that when the commands are installed during SONiC initialization, they are installed with appropriate privilege level.*

The categories of commands that require high privilege level (level 15) are

1. SONiC Configuration commands – SONiC CLI / API that allow user the change the configuration of a module/ system. For example, add / delete bgp neighbor
2. SONiC Clear commands – SONiC CLI / API that allow user to cleat or reset state of operation. For example, “clear bgp neighbor”
3. Docker commands – Docker commands that allow user to change the context from current shell to a docker instance
4. Linux commands that require sudo access – Commands like “ifconfig” that allow user to change the administration state of an interface

Rest of the categories of commands can be access by all users; lower privilege level (level 1) is sufficient to invoke these commands. These categories include

1. SONiC Show commands – SONiC CLI / API that allow user to execute monitor/show commands
2. Linux commands that do not require sudo access

## Authorization

Authorization is achieved in two approaches

1. Role based Authorization (also, referred as Privilege level based Authorization)
2. TACACS+ based Authorization

The configuration knob determines the authorization approach selected by the user.

### Role based Authorization

In Role based Authorization, PAM libraries determine whether to allow a command based on the user role / privilege level provided by the TACACS+ server during authentication or local user group privilege association for the user.

### TACACS+ based Authorization

TACACS+ based Authorization requires every command to be authorized before being allowed to execute. It provides granular level of authorization as deep as command options.

The sequence of steps involved in the TACACS+ based authorization process is

1. TACACS+ based command authorization module gets activated as soon as user was presented with RBAC shell
2. PAM libraries would send encrypted packet for each command entered by user to TACACS+ server along with username information to validate if the command is authorized for the user. The subsequent steps fall under the following scenarios:
   1. Scenario-1: TACACS+ server is reachable and is configured to allow this command for this user – PAM libraries would return TAC\_PLUS\_AUTHOR\_STATUS\_PASS which allows command to be executed on the switch
   2. Scenario-2: TACACS+ server is reachable and is configured to NOT allow this command for this user – PAM libraries would return TAC\_PLUS\_AUTHOR\_STATUS\_FAIL which does not allow the command to be executed on the switch by this user
   3. Scenario-3: TACACS+ server is unreachable
      1. If the authorization fallback is configured, PAM libraries would invoke [Privilege level based Authentication](#_Privilege_level_based)
      2. Otherwise, the user is not allowed to execute this command (or any other command) until the TACACS+ server is reachable

NOTE:

It is important to note that the authorization function asks only the TACACS+ server that originally authenticated the user. It will not move to other servers from the list of configured servers when the original server becomes not reachable (for security reasons).

# AAA Configuration Model

The following table describes the AAA configuration model.

AAA table:

AAA configuration attributes global to the system

AAA {

enable : string

Enable / Disable AAA functionality in the system

Default: enabled

Authentication\_protocol : string

Defines default authentication protocol group.

Enum:

[ tacacs+, local ]

default: local

Authentication fallback : string

Defines authentication mechanism fallback mechanism

Enum:

[ enable | none ]

default: enable

authorization\_protocol : string

Defines default authorization protocol group.

Enum:

[ tacacs+, local ]

default: local

authorization\_fallback : string

Defines authorization mechanism to fallback to privilege level based authorization when enabled.

Enum:

[ enable | none ]

default: enable

## CLI for AAA

CLI configuration commands for AAA built on top of RESTful interface would look like:

aaa enable

aaa authentication protocol tacacs+

aaa authentication fallback enable

aaa authorization protocol tacacs+

aaa authorization fallback enable

The following set of Show commands relevant for AAA will be supported:

show user

show user privileges

show aaa

show aaa authentication

show aaa authorization

# TACACS+ Configuration Model

## Global Parameters

The following table describes the TACACS+ configuration model with attributes that are common across all the servers in the system.

TACACS+ Global Table:

Defines TACACS+ configuration attributes global to the system. Only one instance of the table exists in the system.

TacacsPlusGlobal {

timeout : integer<1 – 60>

The timeout value specifies the number of seconds to wait for a response from the TACACS+ server before moving to next server.

Default: 5

passkey : string

The server passkey is the shared secret used for encrypting the communication between TACACS+ client running on the switch and the server. The secret can be of length up to 32 characters.

default: “”

authentication\_type : string

The authentication type specifies the method used for authenticating the communication messages between the TACACS+ client and the server

Enum:

[ pap, chap, mschap ]

default: pap

*Any of the global parameters can be over-written by a fine grained per server setting by configuring the corresponding attribute for the server.*

## Server Configuration

The following table describes the configuration model for TACACS+ server.

TACACS+ Server Table:

Defines TACACS+ Server configuration in the system. The table is indexed using the address and tcp\_port number.

TacacsPlusServer {

address : ip-address

Defines the IP address of the TACACS+ server. It can be either IPv4 or IPv6 address.

priority : integer <1 – 64>

The priority of the TACACS+ server among the list of TACACS+ servers configured.

tcp\_port : integer <1 – 65535>

The TCP port number that the server is configured to listen on.

default: 49

timeout : integer<1 – 60>

The timeout value specifies the number of seconds to wait for a response from the TACACS+ server before moving to next server. The default value is taken from the TacacsPlusGlobal Table

passkey : string

The server passkey is the shared secret used for encrypting the communication between TACACS+ client running on the switch and the server. The secret can be of length up to 32 characters. The default value is taken from the TacacsPlusGlobal Table

authentication\_type : string

The authentication type specifies the method used for authenticating the communication messages between the TACACS+ client and the server. The default value is taken from the TacacsPlusGlobal Table

Enum:

[ pap, chap, mschap ]

## CLI for TACACS+

CLI configuration commands for TACACS+ built on top of RESTful interface would look like:

tacacs-server timeout <1 – 60>

tacacs-server passkey <shared-scret-of-length-up-to-32-characters>

tacacs-server authentication type [ pap | chap | mschap ]

tacacs-server authorization fallback

tacacs-server <ipv4 | ipv6>

priority <1-64>

port <1 – 65535>

{ key <passkey> }

{ timeout <1-60> }

{ authentication-type [pap | chap | mschap] }

The following set of Show commands relevant for AAA will be supported:

show tacacs-server

show tacacs-server detail

show user privileges

show aaa