MICROSENS

Application Note

Securing G6 Devices and Your Network - Security Tips

MICROSENS GmbH & Co. KG
Küferstr. 16
59067 Hamm/Germany

Tel. +49 2381 9452-0
FAX +49 2381 9452-100
E-Mail info@microsens.com
Web www.microsens.com

Summary

Generation 6 (G6) and later-based Ethernet Switches and Management Modules offer a vast number of features and options. For the casual operator it may not be so obvious which options should be used to safeguard the system and network against exploitation and misuse.

This document describes the parameters related to security issues and offers guidelines for setting them up effectively.

Table of Contents

1 Management Access Security

This section is concerned with protection of the management interfaces.
With full access to the management system it is possible to change all device settings including disabling other network protection mechanisms.
Therefore, management access protection is extremely important.

1.1 Physical Access

Device.Hardware.factory_reset_button = Disabled / Enabled
When unauthorized physical access to the device is possible, the factory button can be used to reset the software to factory defaults. Setting this parameter to disable turns the factory restore option off. By default this setting is enabled.

Note: You can define a custom configuration and assign this to be the default configuration which is loaded should the device be forced to the factory default setting.
This is accomplished with the following steps:
Management.Files.configuration.backup_to_folder = myfactory_config
Management.Files.configuration.factory_default_folder = myfactory_config

Device.System.serial_port = DISABLED
Each G6 device offers a local serial RS-232 console port. The advantage of such a port is the fact that it remains accessible even if all IP parameter are incorrect or unknown.
When unauthorized physical access to the device is possible, this port could be used to log into the console provided the user/password is known. Of course the same could be done via Telnet or SSH. Therefore, the additional risk here is low. Nevertheless, the serial port can be disabled for login access.

The serial port may be configured for use with building automation, to access other serial devices or under user programmable script control.
Any setting other then "CONSOLE" effectively disables the use as console and can be considered secure.

1.1.1 Disable Telnet when RS-232 console access is enabled simultaneously

Each G6 device offers an access to its command line interface through Telnet. To prevent access to the G6 device through Telnet but to still allow for the option to login through the RS-232 port execute both of the following commands as listed below:

Management.Cli.Enable_telnet = DISABLED
Management.Access.User[admin].enable_telnet_access = ENABLED

In addition the following command must be executed:

Device.System.serial_port = CONSOLE

This leaves the console open but Telnet is generally disabled.

1.2 Management Protocols

Device.System.permit_debug_access = Disabled / Enabled
G6 devices are Linux based. Access to the Linux system is possible via a secret login/password combination. Such access may be used by an authorized service technician in order to recover from severe problems or for advanced troubleshooting.
When set to disabled, this access is fully blocked.

Management.CLI.enable_telnet = Disabled
Management.CLI.enable_ssh = Enabled
The CLI console terminal, when invoked with admin privileges, provides full access to all features of the device. It can be accessed using the Telnet or SSH protocols. A typical windows program to connect is "PuTTY" which can freely be downloaded from the Internet.
While both protocols appear to work identical, SSH encrypts the management data traffic, while telnet does not.
It is therefore advisable to disable Telnet.
We recommend leaving the SSH access active in case you need remote access to the command line.

Management.WEB.protocol = HTTPS_SECURE
Similar to SSH, the HTTPS web access provides encrypted management data traffic.
It is therefore advisable to enforce the use of HTTPS. This automatically disables the insecure HTTP protocol.
When HTTPS is used the browser may display a certificate alarm indicating an unrecognized certificate. We encourage the use of a private company certificate.
Such a certificate can be downloaded and activated in the G6 device using the commands under this path:
Management.Files.certificate.

Management.SNMP.v1v2_config.enable_snmp_v1 = Disabled / Enabled
SNMP provides yet another management access protocol. Under G6 SNMP private MIBs for each parameter, even those of additionally installed apps, are provided. In addition numerous standard MIBs are supported in parallel.

Unfortunately, the popular (because easy to use) SNMP versions 1 and 2c provide close to none security at all. There is no login and only a clear text password (the community string) which in most systems is left at the default setting of "public".
From a security standpoint, we therefore recommend to disable SNMP V1 and V2c.

However, often general network monitoring software is used which often relies on these open SNMP V2c protocol. In order to increase safety, while still maintaining SNMP V2c, several additional measures can be taken in a G6 device as shown below.

Management.SNMP.v1v2_config.permit_v1v2_set_commands = Disabled
When set commands are disabled, it is not longer possible to change any setting via SNMP. This is the strongest filter that can be set.

Management.SNMP.v1v2_config.get_community = public
Management.SNMP.v1v2_config.set_community = private
Of course the well known community strings should be replaced by other strings than the indicated defaults. This is a weak measure, but at least effective against a casual "guest".

Management.SNMP.v1v2_config.snmp_v1v2_username = public
The SNMP user name is a unique G6 feature, which can be used to further limit the scope of SNMP.
SNMP v1/v2c has no concept of a user and thus no access right profiles apply. With G6 devices a default user name can be assigned.
For every SNMP v1/v2c transaction the system applies the restrictions that apply to the user(name) configured here. The default name "public" refers to the limitations defined for this default user "public". Details can be found under:
`Management.Acces.user[public].`
Also see the section: „Advanced topic: Precisely crafted access profiles" in this document for details.

The default settings imply that no write access is permitted and that only general parameters can be viewed. Thus most of the system parameter and status values are NOT available with this setting.
To make use of this powerful feature, either the settings of the public user can be modified, or a completely new username, for example "snmp" could be added, which only offers permission (read and/or write) to precisely those parameters that need to be available via SNMP.

Note: To fully permit SNMP v1 / v2c to all parameters (for testing) set the parameter:

Management.SNMP.v1v2_config.snmp_v1v2_username = admin
Management.SNMP.v1v2_config.permit_v1v2_set_commands = Enabled

Management.SNMP.v3_config.enable_snmp_v3 = Enabled
SNMP version 3 is the secure version of SNMP. While this protocol no longer justifies the "Simple" in its name, it is reasonably well protected.
In order to use SNMP v3 it is required to set several parameters and to match them with the setting of the SNMP browser.
These settings are associated with the user names and are explained in the following chapter.

1.3 User names and passwords

The management access protection scheme hinges on a user/password scheme.
The user names can be defined locally or may be sourced from a central database via RADIUS or TACACS+.
For this discussion we will focus on locally defined users only.

Management.Access.user[user].general_access_rights = NO_ACCESS
Management.Access.user[public].general_access_rights = NO_ACCESS
The default settings provide 3 users. Up to 32 users may be defined locally.
It is recommended to keep the user "user" and change its passwords from the default "microsens" or to disable unused user altogether as shown above. Disabling keeps the other user related settings for possible future use.

```
Management.Access.user[user].general_access_rights = READ_ONLY
```
Another effective way is to set the general access rights to READ_ONLY.

```
Management.Access.user[public].enable_telnet_access = Disabled
Management.Access.user[public].enable_ssh_access = Disabled
Management.Access.user[public].enable_web_access = Disabled
Management.Access.user[public].enable_snmp_access = Disabled
Management.Access.user[public].enable_nmp_access = Disabled
Management.Access.user[public].enable_ftp_access = Disabled
```
For each user there are options to disable certain access protocols.
Note: It is still possible to log into the CLI with an "NO_ACCESS" user. Some build-in commands (such as Logout and WhoAmI) are then still available but no harm can be done. To fully shut down the user, also disable the individual protocols as shown above.

```
Management.Access.user[admin].enter_password = your_password
```
Definitely, the default passwords should be overwritten!!
The password must be at least 3 characters. A short password can be convenient while testing but for a live environment a complex password provides better security.
Permitted characters:
Digits: 0-9
Letters: a-z, A-Z
Characters: # $ * ? ( ) ! . @ % = { } ~ + - , ^ _(underscore) (space)in the middle

Note: leading or trailing space characters in an entered password are removed prior to further processing of the password. Consecutive space characters embedded in a password are collapsed into one space character prior to further processing of the password.

```
Management.Access.user[admin].encrypted_auth_password = your_password
Management.Access.user[admin].encrypted_priv_password = another_password
```
When SNMP v3 is enabled, also the default values for the SNMP passwords should definitely be changed.
This step is easily forgotten!
SNMP v3 requires at least 8 character passwords. The authentication password and the privacy password must be different!

```
Management.Access.user[admin].snmp_v3_security_level = AUTH_PRIV
```
Be sure to set the security level to the highest setting, which requires authentication and provides privacy (encryption).
Setup the SNMP browser to use the same settings.
The SNMP v3 settings must be updated for each user.

Hint: The following syntax sets the security level for all users in one command:
```
Management.Access.user[*].snmp_v3_security_level = AUTH_PRIV
```

1.4 File transfers and updates

```
Management.Files.server.enable_tftp = Disabled
Management.Files.server.enable_ftp = Disabled
```

Management.Files.server.enable_sftp = Enabled
File transfers to the G6 device may be needed to update the firmware or to exchange configuration files.
It is recommended to disable TFTP.
FTP and SFTP both require login using the user/password scheme.
SFTP additionally encrypts the traffic using SSH and as such is the preferred file transfer protocol.

Note that G6 (and later) firmware update files are always encrypted *and* digitally signed.
With regard to a G6 device the download of firmware files is the most frequently used file transfer application. Thus FTP is sufficient for such kind of transfers.

To be able to *push data to* or *to pull data from* the device the desired server needs to be enabled.
Alternatively, a file transfer may be initiated from the device itself. In this case the servers are not needed and may be disabled (see example below).
Example to download a firmware update without a local server running:
Management.Files.firmware.download =
ftp://name:password@machine.domain:port/full/path/to/firmwarefile.msu
(one line)

1.4.1 Advanced topic: Precisely crafted access profiles

In most cases the standard default settings for admin and other users are sufficient.

The G6 system, however, can precisely be set up to permit access control down to each particular parameter.
This is accomplished with the help of "pattern". Similar patterns are combined in "views".
Any combination of views can further be combined in "groups". Finally, one or more of such groups are assigned to the user via the "associated_groups" parameter.
When creating your own profile, it is best to start with the pattern definitions.
For illustration there are a number of patterns, views and groups predefined in the system.
Feel free to edit or add additional items.

Let's explore an exisiting pattern:

```
Management.Access.pattern[PORT_NAMES].
name                    :  PORT_NAMES
dotstring               :  port.config.alias
access_rights           :  READ_WRITE
```

This pattern declares the parameter (dotstring): `(Device).port.config[*/*].alias` to be fully accessible for read and write. It does not imply any other rule.
Note: The first part "Device", as it would be typed in the CLI, is not part of the definition. This also applies to the other leading keywords such as "Management" or "Protocol".

Let's check out a view that uses the PORT_NAMES pattern:

```
Management.Access.view[ALIAS_PARAMETER].
name                    :  ALIAS_PARAMETER
associated_pattern[1]   :  USER_FIELDS
associated_pattern[2]   :  VLAN_NAMES
associated_pattern[3]   :  PORT_NAMES
```

This view contains our PORT_NAMES pattern plus two other patterns that each are concerned with user definable alias names. These are fairly uncritical parameter and can be allowed to be changed by a user with a lower privilege than the admin user.

Next we will see how this view is used:

```
Management.Access.group[limited group].
name                    :  limited group
associated_views[1]     :  ALIAS_PARAMETER
associated_views[2]     :  CLI_SETTINGS
associated_views[3]     :  TEST_FUNCTIONS
associated_views[4]     :  TIME_AND_LED
associated_views[5]     :  SNMP
```

The "limited_group" contains a number of views. Among these we find the "ALIAS_PARAMETER" view which in turn contains among others the `port.config.alias parameter` as read/writable.

Finally this "limited_group" is assigned to the user "user":

```
Management.Access.user[user].associated_groups[*]
[user].associated_groups[1]: limited group
[user].associated_groups[2]: public group
```

What is still missing is the default behaviour for any parameter parameter-dotstring that does not match any pattern. In the case of the default user "user" this is accomplished by the following pattern, which comes into play via the second assigned group, the public group.

```
Management.Access.pattern[DEFAULT_PATTERN].
name                    :  DEFAULT_PATTERN
dotstring               :  *
access_rights           :  READ_ONLY
```

The "*" in above pattern means "any dotstring". This is a special keyword.
To indicate, for example, anything under the vlan group just use "vlan". If several pattern definitions overlap, then the more precisely matching dotstring takes precedence.
Remember that the first keyword level (like Device.) is not part of the pattern.

Note that limitations defined via the "general_access_rights" override the outcome of the pattern matches. That is, if the "general_access_right" is set to "READ_ONLY", no parameter can be written regardless of any pattern indicating "READ_WRITE" privileges.

Important: The access profiles fully apply to CLI and SNMP access.
NMP and Web access profiling is under development. In the meantime only the "general_access_rights" are supported for these interfaces.

In summary, the advanced G6 access scheme permits flexible and precise profiling at the expense of being somewhat complex to be set up.

For most applications, simply setting the "general_access_rights" is a sufficient measure.

2 Port Security

All previous sections discussed protection of the device management itself.
The following sections are dedicated to protecting the networks against malicious data.

In some networks, the switch does not need to take any precautions. Based on the automatically learned MAC addresses, it simply *switches* any data traffic between its ports. All attached devices are deemed save and no special filtering is required.
If your network falls into this category you can stop reading here.

2.1 Port Access Control

The switch offers a wide range of port access control related features. These are grouped in the section "PACC" (PortACcessControl).

```
Protocol.PACC.port_config[*/*].authorize_mode = ALWAYS_AUTHORIZED
Protocol.PACC.enable_port_access_control = Enabled
```

Before turning on the PACC function ensure that all ports that are not using access control (especially the uplink) are set to ALWAYS_AUTHORIZED.

Attention: Ensure that the device uplink port and the port by which you are currently connected to the management will not be blocked, once PACC is generally enabled. Otherwise you may be locked out!

`Protocol.PACC.port_config[1/1].authorize_mode` (replace 1/1 with the desired port)
This port specific setting defines which, if any, authentication mode is selected.
The options allow for local authentication using the MAC address of the attached device or to use an external authentication server.
Also the protocol IEEE 802.1X may be used for authentication via RADIUS.
We assume that readers who use external authentication are sufficiently skilled to get going. Therefore, we will focus on the options of local authentication next.

Local authentication using the MAC address is selected with the following setting:
`Protocol.PACC.port_config[1/1].authorize_mode = VIA_MAC_TABLE`

Any time a new device (not already learned) sends data to the port, its new MAC is detected and verified against a table of valid addresses.
This table is defined and viewable under:
`Protocol.PACC.authorized_macs[*].mac_address`

There are two methods to populate this table. It can be done completely manually or semi-automatic using a feature called *MAC learning*.
It is a lot more comfortable to use MAC learning as usually the external device's MAC adresses are not known and are not easily obtained.
For this feature to work do this:

```
Protocol.PACC.port_config[1/1].learn_mac_now = 1
```

Above line will accept the next device from which is receives data on port 1/1.
Up to 9 MACs may be specified to be learned. There may be a long time gap between these.
The learning mode remains active until enough new MACs have been added or a port restart is executed.

The learning may also be stopped typing:
Protocol.PACC.port_config[1/1].learn_mac_now = 0

Please ensure that only valid devices will send data while learning to avoid accepting an illegal source as legitimate MAC.

Protocol.PACC.port_config[1/1].limited_number_of_macs = 10
With the limited_number_of_macs parameter the total number of MACs (=devices) that can operate on a given port can be limited. In many cases the number of devices is 1 or 2. When a second switch is cascaded, a lot more devices may come in.
The idea is to limit the number to the expected value as a precaution against a potential unfriendly additional device.
If the number of devices is unclear leave the parameter at the default value of 0.

2.2 Advanced topic: Optical power level monitoring

Device.SFP.config.delta_threshold = 2
Device.SFP.config.optical_delta_detect = Enabled
In some networks optical link interconnections are used from the switch to a central place. Optical links are very reliable and secure. However, it is potentially possible to tap such a line using special couplers. Also accidental bending of fiber cables in your fiber installation can lead to higher loss and in some cases to an increased transmission error rate.
With the optical_delta_detect function, fluctuations of the optical signal level are reported via events/traps, even if the changes are uncritical and do not lead to connection loss. This can be used as an indicator that something unexpected is happening with the optical cables.

3 ARP Security

This section discusses an advanced security feature offered for G6 based devices.
To appreciate the feature it is useful to take a quick look on how IP works first.

3.1 Sending a packet

When a computer or any other network node tries to reach a certain IP address, it must first find out the MAC address of the target device or gateway. Once the MAC address is known the data will be send to this MAC address.
To find out the MAC the Address Resolution Protocol (ARP) is used.

The computer sends out a so-called ARP request (a layer 2 broadcast frame), asking for the "owner" (and the corresponding MAC address) of the target IP address in question. Only the device with this IP address will respond with an ARP response in which it will return its MAC address. Once received, the computer will cache this IP/MAC relation for future packets in the so called ARP table. Finally, the computer can send its data.

If the IP target address is outside the local network, the ARP lookup will fail. So all packets targeted outside the local subnet are sent to the gateway for further forwarding to the target node.

To make the protocol more efficient, every IP device is permitted to send “gratuitous” ARP responses which contain the MAC/IP relation of the device. Other network members take note and cache this IP/MAC relation in an attempt to immediately know where to send to without an extra ARP inquiry.

3.2 Man-in-Middle-attack / ARP spoofing

There is no protection mechanism in the protocol that prevents a malicious (illegal) device from sending a wrong mapping to the network. It could e.g. send the correct IP of the gateway, but insert its own MAC address. From then on, all traffic to the gateway (all external traffic) would be sent to the attacker instead. By this trick an attacker can force all traffic to be rerouted through his machine for monitoring.

In the illegal device software could be installed that copies all traffic and then continues to send the data to the actual gateway. This way communication would still be working, but the data would be copied covertly (stolen). Even worse, the malicious software could alter the content of certain packets, before passing them along. All of this will happen unnoticed. This is called a “Man-in-the-Middle” attack.

The solution to detect such illegal ARPs that precede a “Man-in-the-Middle” attack is called “ARP inspection”.

3.3 Detection

In principle it would be easy to detect that an ARP is legal, if the proper and legal relations were all known. If an ARP does not match the already known legal relations, it must be false.

Most of the time, the situations gets worse when it comes up to manually maintain large and changing lists of MACs and IPs – the operators will not be pleased. Also, often the MACs are actually unknown and will be not at hand immediately.

Therefore an automatic solution is desirable. It can be used if the IP addresses are provided using DHCP.
The automatic detection process is called DHCP snooping.

DHCP snooping is a function whereby all DHCP traffic passing the switch is detected, evaluated and recorded. While the snooping function does not alter the DHCP traffic, it keeps track of all assigned IP addresses assigned via DHCP and the associated devices' MAC addresses. Thus over time it builds a table that contains all IP/MAC relations that where supplied by the server.
This also means that the switch has to keep track of DHCP traffic in real time.

Moreover, this table is saved to local flash memory so that the relations are immediately available even if the switch is rebooted.

3.4 ARP Inspection Configuration

The use of DHCP snooping and/or a manually managed access control list (ACL) is a prerequisite to use ARP inspection. In fact, in most networks there are some manually assigned IP addresses as well, so that a combination of DHCP snooping and an ACL is used.

First we will show how to set up a manual access control list.

```
Management.ACL.rules[My PC].
name                    :  My PC
description             :  Example to match one device
mode                    :  PERMIT
source_mac              :  00:13:77:B4:1C:D6
source_ip               :  10.100.89.232
source_mask             :  255.255.255.255
vlan_id                 :  1
```

This rule declares a valid MAC and IP relation.
Note that the source_mask has all bits set to 1. Unlike a subnet_mask, here this mask declares how many bits are verified.
For example a source_mask: 255.255.255.0 would match any IP starting with 10.100.89.??.
There are no wildcards for the MAC address.

The vlan_id is used only when the port receives vlan tagged data. Otherwise it is ignored.

Enter as many rules as needed.
The next step is to assign the rules which apply together for a given port to an ACL list:

```
Management.ACL.list[my_desk].
name                    :  my_desk
description             :  Every rule that applies to my access port
rules                   :  my pc, my phone
```

Above list contains two rules, separated by a comma.
Finally, the rule must be applied to a port. This is done with the next line:

```
Protocol.DHCP.arp_inspection_port_config[1/2].arp_acl_name = my_desk
```

Other parameter associated with ARP inspection are:

Protocol.DHCP.arp_inspection_port_config[1/2].acl_default_logic = DENY
The default logic declares what happens if none of the rules apply. In most cases the port should be closed to anyone not listed. This is accomplished with DENY.

There could be a situation where anyone is permitted, and only certain devices are specifically blocked. To set this up, the default logic is set to PERMIT and the rules must be defined with a mode of DENY when matched.

Protocol.DHCP.arp_inspection_port_config[1/2].source_mac_validation = Enabled
Protocol.DHCP.arp_inspection_port_config[1/2].dest_mac_validation = Enabled
Protocol.DHCP.arp_inspection_port_config[1/2].ip_range_validation = Enabled
These settings control further checking of the ARPs. In particular it is checked if the MAC declared with the ARP packet actually matches the MAC of the sending device. Any difference usually indicates a device that tries to impersonate another.
Additionally, both the IP address must be known to the switch and the corresponding MAC address must match those already snooped by the switch.

The ip_range_validation checks that no broadcast or other illegal ip values are being transferred in an attempt to confuse the network.

Protocol.DHCP.arp_inspection_port_config[1/2].arp_rate_limiting
To spy out a network or to create a denial-of-service (DoS) situation, an attacker could fire ARP requests into the network with every conceivable IP address. With ARP rate limiting such a condition is detected and the corresponding port is being shut down automatically. Usually, the default of 10 ARPs per seconds should be sufficient. This value should only be modified if the port shuts down under normal conditions.

Below is the final configuration for a typical access port:

```
Protocol.DHCP.arp_inspection_port_config[1/2].
enable_arp_inspection    :  Enabled
arp_rate_limiting        :  10
inspection_database      :  ARP_ACL
arp_acl_name             :  my_port
acl_default_logic        :  DENY
source_mac_validation    :  Enabled
dest_mac_validation      :  Enabled
ip_range_validation      :  Enabled
```

Protocol.DHCP.arp_inspection_port_config[1/2].inspection_database = BOTH
In order to use automated DHCP controlled MAC-IP relations, the inspection_database parameter may be set to DHCP or BOTH.

In order for the DHCP part to work, the parameters explained in the following chapter have to be set up.

Finally, turn on the global function!
Protocol.DHCP.enable_arp_inspection = Enabled

4 DHCP Security

4.1 Rogue DHCP Servers

DHCP is the protocol with which devices can automatically obtain an IP address, if not manually configured. DHCP is most often used when for example a PC is connected to the IP network. The PC will ask for an IP address, and the DHCP server will supply an IP address out of a pool of permitted addresses.

4.2 Risk

A DHCP server is actually just a piece of software that can run on every PC. Some techies run such a server for their home network. When such a PC, with a running DHCP server is accidentally or willingly connected to a user port of a network, problems can arise.

It may happen that other legitimate devices on the network will ask for their IP and gateway and the PC of our techie will respond (when quicker than the regular DHCP server). Suddenly wrong IP addresses are handed out and the subsequent communication will fail. More serious, a purposefully inserted bad DHCP server could download malicious scripts or supply fake gateway addresses.

4.3 DHCP Snooping

The most effective way to eliminate the threat of an illegal DHCP server is to filter out all DHCP response traffic on untrusted user ports.
The feature of listening in on the DHCP traffic is called "snooping".
DHCP snooping is also required for ARP inspection to populate the table with valid MAC/IP relationships.

4.4 DHCP Snooping Configuration

Protocol.DHCP.snooping_port_config[1/2].dhcp_filtering = BLOCK_AND_EVENT
Once an illegal DHCP server is detected, one of two possible actions can be performed. Either the DHCP message is removed from the data stream ("DROP_AND_EVENT") or the whole port is blocked for all traffic ("BLOCK_AND_EVENT"). The latter action has the stronger effect and will inhibit a potential intruder completely.
In either case a management event is raised to inform the operator.

Protocol.DHCP.snooping_port_config[1/2].snooping_trust = AUTO
It is very important to apply DHCP snooping on ALL ports with a risk potential.
A clear distinction is made between trusted and untrusted ports.
In the "AUTO" setting, the general rule on the switch is that link ports are deemed trusted while access ports are defined as untrusted.
A trusted port must also be the port where the correct and legal DHCP server is reachable. This is needed as a reference for address verification and ARP inspection.

When the port assignment is used in a non-standard way and the port role is not properly adapted, the AUTO setting will fail. Use the TRUSTED or UNTRUSTED setting when in doubt.
The port role can be viewed here:
Device.Port.downlink_ports
Device.Port.uplink_ports

The port role itself is defined here:
Device.Port.config[1/2].role

Protocol.DHCP.snooping_port_config[1/2].dhcp_rate_limiting = 10
To create a denial-of-service (DOS) situation, an attacker could fire DHCP messages into the network. With DHCP rate limiting such a condition is detected and the port is shut down automatically. Usually, the default of 10 DHCP requests per seconds should be sufficient. Only modify this value if the port shuts down under normal conditions.

Protocol.DHCP.snooping_port_config[1/2].mac_address_verification = Enabled
When a packet is received on an untrusted interface, and the source MAC address and the DHCP client MAC address does not match and this feature is enabled, the packet is dropped.

Protocol.DHCP.snooping_port_config[1/2].accept_ingress_option82 = Disabled
Normally incoming DHCP request incoming with Option 82 set will be discarded. Enable this feature when DHCP Option 82 is used and permitted on that port.

Protocol.DHCP.snooping_port_config[1/1].enable_dhcp_snooping = Enabled
Finally, when all port setting are done, enable the portwise feature.

When all ports are set up, generally turn on DHCP snooping!
Protocol.DHCP.enable_dhcp_snooping = Enabled

Note: There is no need to disable and re-enable DHCP snooping (or ARP inspection) when changes are made. It is just good practice to initially setup everything before the first use, in order to minimize the risk to accidentally lock yourself or someone else out.

To view the MAC/IP relations learned (and used for ARP inspection) look here:
Protocol.DHCP.snooping_table[*].

Disclaimer

Document ID: AN-15001_2015-11-05