Compliance signalling

Signalling

A compliance check can have one of the following results:

- A compliant node becomes non compliant
- A non-compliant node becomes compliant
- A non-compliant node stays non-compliant
- A compliant node stays compliant

Whether an action should be undertaken after a policy check is stored in the policy's Signal_trigger mask.

- Bit 1: compliant to non-compliant
- Bit 2: non-compliant to compliant
- Bit 4: non-compliant to non-compliant
- Bit 8: compliant to compliant

This setting is available in the front-end when editing a policy using four separate check-boxes

There are four ways we can send signals, which is stored in the policy's Signal_type as a bit-map:

- Bit 1: A trap message
- Bit 2: A syslog message
- Bit 3: An email message
- Bit 4: A REST API 'post' call using Json

This setting is available in the front-end when editing a policy using four separate check-boxes allowing for multiple signal types for the same event.

These signal messages need setting-up for the environment and mostly requires the definition of the remote targets and priority. Configure these in the file **`etc/signal_cmpl.conf`** which is copied from `system/signal_cmpl.conf` when missing.

It contains a simple perl struct, containing a section for each of the four different signal types.

Traps

An SNMP (v2) trap has the following parameters that can be configured:

- **enterprise:** the traps's enterprise, a single number. We append this to “1.3.6.1.4.1.”.
- **specific:** the trap's specific number
- **varoid:** the trap's varoid
- **community** the trap community string
- **tmsg:** the compliance-event will create its own trap-message. Include your own message to

override it. The compliance-event message will still be included in the trap.
- **no_spoof:**optional setting to disable the spoofing of the snmp-trap with the node's ipv4 address if resolvable. Set to '1' to disable spoofing.
- **nmsaddr:** the addresses(es) to send the trap to. Add all trap-destination ip-addresses to this array.

The compliance-event message formats:

```
when compliant:
[CMPL] Node <Hostname> is compliant for policy <Policy_name>

when not-compliant:
[CMPL] Node <Hostname> is not compliant for policy <Policy_name> - Severity:
<Severity> see
<Server>/operate/cmpl_report.pl?Cmpl_node_id=<Cmpl_node_id>&Detail_level=1&c
hoice=policy_single&type=nodes for more details
```

The relevant entries in the `signal_cmpl.conf` file are:

```
# Available values:
# - trap
#    - enterprise        # default to '696'
#    - specific          # defaults to '0'
#    - varoid            # defaults to '1.0'
#    - community         # defaults to 'public'
#    - tmsg              # optional trap-message override. use undef to use
default cmpl messages
#    - messages (array)  # optional text lines to be included as individual
oids
#    - nmsaddr (array)   # a list of ip-addresses to send trap to. Defaults
to Common::traphosts

our $cmpl_signals = {
    'trap' => {
        'enterprise' => '696',
        'specific'   => '0',
        'varoid'     => '1.0',
        'community'  => 'public',
        'tmsg'       => undef,
        'no_spoof'   => 0,
        'messages'   => [
            "additional messages line one",
            "additional messages line two",
        ],
        'nmsaddr'    => [
            '127.0.0.1',
            '2.3.4.5',
            '3.4.5.6',
        ],
    },
    #
    # ... other signal-type definitions
    #
};
```

The trap will also include additional compliance variables associated with the event:

- Hostname
- Vendor_type
- Node_fqdn
- Node_ipv4_addr
- Node_ipv6_addr
- Policy_id
- Policy_name
- Nccm_id
- Cmpl_node_id
- Severity
- Status
- Server
- Report

The values for these variables include the variable name in the “<variable>: <value>” format.

By default the SNMP Trap messages (udp) are '**spoofed**'. Here the 'spoofing' refers to the 'faking' of the source ip-address of the Trap message by replacing the server address with the node address.

The SNMP Trap will use the node ip-address instead of the NetYCE server as the source if the node-fqdn can be resolved using the DNS in an ipv4 address. Otherwise the NetYCE server will be used as the source address:

- The node name must exist in YCE or CMDB and have an Node_fqdn defined
- If the Node_fqdn is an ipv4, use that address as the (spoofed) agent-address
- If the Node_fqdn can be resolved in an ipv4, use that address as the (spoofed) agent-address
- Otherwise the NetYCE server's ipv4 address will be the agent address - the current default assignment

To disable trap spoofing add “'no_spoof' ⇒ 1,” to this section of the trap signal config.

Syslog

The compliance-event can send a syslog message to one or more remote syslog receivers using a configurable syslog facility and priority.

The syslog message format:

```
when compliant:
  [CMPL] Node <Hostname> is compliant with policy <Policy_name>
when not-compliant:
  [CMPL] Node <Hostname> is not compliant with policy <Policy_name> -
```

```
Severity: <Severity> - see
<Server>/operate/cmpl_report.pl?Cmpl_node_id=<Cmpl_node_id>&Detail_level=1&c
hoice=policy_single&type=nodes for more details
```

All syslog messages are assigned a 'facility' (category) and a 'priority' (severity):

```
Valid 'facilities' are:
  kern user mail daemon auth syslog lpr news uucp cron authpriv ftp
  local0 local1 local2 local3 local4 local5 local6 local7

Valid 'priority' values are (ordered high to low):
  emerg alert crit err warning notice info debug
```

The relevant entries in the `etc/signal_cmpl.conf` file are:

```
# - syslog
#    - facility          # defaults to 'daemon'
#                        # one of:
kern|user|mail|daemon|auth|syslog|lpr|news|uucp|cron|authpriv|ftp|
#                        #
local0|local1|local2|local3|local4|local5|local6|local7
#    - priority          # defaults to 'warning' - needs mapping from
severity
#                        # one of:
emerg|alert|crit|err|warning|notice|info|debug
#    - priority_map      # translation hash of numeric 'Severity' to
'priority' string, overrides 'priority'
#    - nmsaddr (array)   # a list of ip-addresses to send syslog to.
Defaults to Common::traphosts

our $cmpl_signals = {
    'syslog' => {
        'facility'   => 'daemon',
        # translate the event severity to the desired priority
        'severity_map' => {
            # '6' => 'emerg',
            # '5' => 'alert',
            'High' => 'crit',
            'Serious' => 'err',
            'Medium' => 'warning',
            'Low' => 'notice',
            # '0' => 'info',
            # '1' => 'debug',
        },
        # the default priority
        'priority'   => 'warning',
        'nmsaddr'    => [
            '127.0.0.1',
            '172.17.10.20',
            '172.17.10.28',
        ],
    },
    #
    # ... the other signal-types
    #
};
```

The 'facility' is directly configured using its name. For the 'priority' a mapping can be made from the compliance Policy **Severity** to the desired syslog 'priority'. Since the Policy Severity can be freely configured and the number of severity can be altered, this mapping should be modified to match the used setup. When the mapping does not result in a match, the default 'priority' setting is used.

The default set of 'severity' values uses 'Low', 'Medium', Serious' and 'High' and can be altered using Custom data under 'Nccm_lookup':

The sample configuration shows that these are mapped to an appropriate syslog 'priority' by name.

Email

The compliance-event can send an email message to one or more email-addresses.

> Note: Sending email messages as notifications doesn’t scale well. There is significant processing overhead involved in sending emails compared to traps or syslog which will make sending hundreds or even thousands of mail messages undesirable at least. Not to speak of a mailbox being bombed that way. **This signalling type should be reserved for special cases or during development of new policies only.**

The relevant entries in the `etc/signal_cmpl.conf` file are:

```
# - email
#    - subject            # defaults to 'NetYCE compliance'
#    - mail_to            # array of mail addr. defaults to 'yce@<server-
fqdn>'
#                         #  may also be space or comma separated string of
addr
#    - mail_from          # defaults to 'yce@<server-fqdn>'
#.   - attach_report_body # if set to 1, it will attach the report text to
the email body, truncated to 500 lines
#    - messages (array)   # optional text lines to be included as email
trailer
#

our $cmpl_signals = {
    'email' => {
        'subject'   => 'NetYCE compliance',
        'mail_to'   => [
            'yce@nms.netyce.org',
        ],
        'mail_from' => 'yce@netyce.org',
        # optinal lines included in the email trailer
        'messages'  => [
            "additional messages line one",
            "additional messages line two",
        ],
    },
    #
    # ... other signal-types
    #
};
```

The `mail_to` configuration accepts either an array (set between '[..]' with individual quoted email-addresses) or as a string (a single quoted list of email-addresses). The use of the array format is recommended.

The `mail_from` configuration must contain a single quoted email-address.

The `subject` value is prepended to the compliance-event message. The resulting email subject has the format:

```
when compliant:
  <subject> [<Hostname>] [<Policy_name>|<Severity>]: Compliant";

when not-compliant:
  <subject> [<Hostname>] [<Policy_name>|<Severity>]: Not-Compliant";
```

The mail body will also include additional compliance variables associated with the event:

- Hostname
- Vendor_type
- Policy_id
- Policy_name
- Nccm_id
- Cmpl_node_id
- Severity
- Status
- Server
- Report

The values for these variables include the variable name in the “<variable>: <value>” format.

The mail body uses the format:

```
This is an automatically generated notification
message, please do not reply
----------------------------------------
```

```
<subject> <compliance message>
  ::
  :: ... list of compliance variables
  ::
You can view the policy details at:
<Server>/operate/cmpl_report.pl?Cmpl_node_id=<Cmpl_node_id>&Detail_level=1&c
hoice=policy_single&type=nodes

----------------------------------------
::
:: ... optional report body (set it with the attach_report_body setting)
::




::
:: ... optional trailing messages
::

Issued by <server> at <current-timestamp>
```

The resulting mail is sent as plain text to each of the addresses in the mail_to list.

REST-API

> Note: The configuration of the 'rest-api' signalling type has been changed as of build number 20210226. The section below clarifies the newer method. The newer method allows the user to specify the attribute names and their value of the REST POST to suit the needs of their environment.

The REST API call is a POST transaction in JSON format to a single remote host at a specific url.

Apart of a couple of mandatory attributes, the REST POST message can be customized to fair degree. The mandatory attributes pertain to the definition of the targeted host/url and its authorization. All other attributes in the JSON message must be specified and assigned a value. To create values (strings only) the configuration will use the provided strings as templates where <variables> will be substituted.

The default rest-api configuration demonstrates the use of all available <variables> and can be customized at will. If the current (default) configuration of the rest-api section does not match the section below, cut and paste it from here or copy it from the **`signal_cmpl.conf`** found in the directory `/opt/yce/system`. This file will be copied to /opt/yce/signal_cmpl.conf should it be found missing.

The REST call has the following mandatory parameters that can be set:

- host: The host ip or fqdn to post too. Has to include the transfer protocol (“http:*” or “https:*”) and the port number
- url: The URL path of the REST API to post too. Must have a leading `/`
- webtoken: Optional bearer AUTH's webtoken

- username and password: optional; basic authentication username and password

```
# - rest-api
#    - host              # host web address eg:
'https://server.netyce.org:8080', 'http://192.17.10.28:8080'
#    - url               # Rest-api url part of address:
'/api/v1/nms_events'
#    - username          # basic authentication username part or undef for
none
#    - password          # basic authentication password part or undef
#    - webtoken          # webtoken authorization
#    - attibutes         # include attibutes to be Rest-posted using the
available <variables>
#                          to create their values. Variables:
#                          <Hostname> <Cmpl_status> <Severity> <Policy_name>
<Report_summary> <Report>
#                          <Vendor_type> <Server> <Server_fqdn> <Policy_id>
<Nccm_id> <Cmpl_node_id>
#                          <Node_fqdn> <Node_ipv4_addr> <Node_ipv6_addr>
#-------------------------------------------------------------

our $cmpl_signals = {
    'rest-api' => {
        # mandatory host/url to Rest-post to using Json
        'host'     => 'http://nms.netyce.org:8080',
        'url'      => '/api/v1/nms_events/<Hostname>',

        # Rest authorization as required
        'username' => undef,
        'password' => undef,
        'webtoken' => undef,

        # customizable attributes (create desired attributes and values
using <Name> variables)
        'hostname'    => '<Hostname>',
        'status'      => '<Cmpl_status>',
        'severity'    => '<Severity>',
        'policy_name' => '<Policy_name>',
        'message'     => "[CMPL] Node <Hostname> compliant-status is
'<Cmpl_status>' with policy '<Policy_name>'",
        'summary'     => '<Report_summary>',
        'report_url'  =>
'<Server_fqdn>/operate/cmpl_report.pl?Cmpl_node_id=<Cmpl_node_id>&Detail_lev
el=1&choice=policy_single&type=nodes',

        # more attributes can be structured as fitting
        'details'     => {
            'vendor_type'  => '<Vendor_type>',
            'server'       => '<Server>',
            'policy_id'    => '<Policy_id>',
            'nccm_id'      => '<Nccm_id>',
            'cmpl_node_id' => '<Cmpl_node_id>',
            'report'       => '<Report>',
        },
    }
};
```

The example above also shows that attributes can be grouped forming a data structure.

Previous format

> below this point the older configuration method of the 'rest-api' signal is described.

The REST API call is a POST transaction in JSON format to a single remote host at a specific url. The REST call has the following parameters that can be set:

- host: The host ip or fqdn to post to. Has to include the transfer protocol (“http:*” or “https:*”) and the port number
- url: The URL path of the REST API to post to. Must have a leading /
- webtoken: Optional bearer AUTH's webtoken
- username and password: optional; basic authentication username and password

The relevant entries in the `signal_cmpl.conf` file are:

```
# - rest-api
#    - host              # host web address eg:
'https://devel6.netyce.org:8080', 'http://192.17.10.28:8080'
#    - url               # Rest-api url part of address:
'/api/v1/nms_events'
#    - username          # basic authentication username part or undef for
none
#    - password          # basic authentication password part or undef
#    - webtoken          # webtoken authorization
#    - parameters        # optional struct of data to include
#

our $cmpl_signals = {
    'rest-api' => {
        'host' => 'http://nms.netyce.org:8080',
        'url' => '/api/v1/nms_events',
        'username' => undef,
        'password' => undef,
        'webtoken' => undef,
        # optional struct of data
        'parameters' => '',
    },
    #
    # ... other signal-types
    #
};
```

The optional `parameters` value can be any perl-struct that is to be included in the POST. It is converted to JSON for this purpose.

The submitted JSON structure will have various attributes. The attribute `message` has the compliance-event message.

The format of this message:

```
when compliant:
  [CMPL] Node <Hostname> is compliant with policy <Policy_name>
when not-compliant:
  [CMPL] Node <Hostname> is not compliant with policy <Policy_name> -
Severity: <Severity> - see
<Server>/operate/cmpl_report.pl?Cmpl_node_id=<Cmpl_node_id>&Detail_level=1&c
hoice=policy_single&type=nodes for more details
```

The JSON will also include additional compliance variables associated with the event:

- Hostname
- Vendor_type
- Node_fqdn
- Node_ipv4_addr
- Node_ipv6_addr
- Policy_id
- Policy_name
- Nccm_id
- Cmpl_node_id
- Severity
- Status
- Server
- Report

The `parameters` attribute will have the structure and values from the config file.