

sec2xym

This is a perl module containing subroutines that are intended exclusively for calling from SEC. They provide native (eg cross platform) access to some specific interfaces with an upstream Xymon server.

Exported functions

`init_sec2xym($XYMSRV,$PORT,$LIFETIME,$SECSVC,$SECHOME,$opt_d,$opt_n)`

`sendToXymon($server,$port,$msg)`

Subroutine to communicate with Xymon

`refresh_config()`

Subroutine to fetch a named list of files from the Xymon download directory. Used as a way to keep SEC rule files (*.sr) in sync.

`refresh_logpath()`

Subroutine to maintain the named (static) file hard linked to the newest object in a given folder

`relink_logpath()`

Subroutine to maintain the named (static) file soft linked to the newest object in a given folder

`XymonStatusUpdate()`

Subroutine to simplify sending a status update to Xymon by providing default settings for target name and timestamp and providing flexibility in severity

`fake_signal_handler()`

Subroutine to provide access to native signal processing on platforms where signals not supported, eg Windows with native perl (ActiveState, Strawberry etc)

`XymonStatusModify()`

Subroutine to simplify sending a modify update to Xymon by providing default settings for target name and source and providing flexibility in severity

Running SEC

By default, there is a single runtime configuration file expected which should in addition have a .rc extension, eg

`\apps\sec\etc\SimpleEventCorrelator.rc`

It is referenced in our servers via the SECRC environment variable passed to it via the environment during startup although failure to find this file will not prevent the service from starting.

The runtime configuration file will be used to define :-

- inputs
- options
- further rules (optional)

A typical runtime configuration may look as follows :-

```
# SimpleEventCorrelator.rc
# this file (SECRC) is read first, these settings are treated as
# though they had been entered on the command line
# pull in additional conf files
--conf=etc/xymon.sr
--intevents
# define the log file to be scanned
--conf=etc/CEPoller.sr
--input=D:\apps\tomcat\current\logs\pollerLog.log=CEPOLLER
--reopen-timeout=61
# other useful settings
--debug=6
--dump=tmp/sec.dump
--conf=etc/01control.sr
--input=etc/control=CONTROL
```

Execute SEC with a command line that includes the --conf, --log and --pid options, eg :-

```
D:\apps\Perl64\bin\perl.exe D:\apps\sec\bin\sec.pl
--conf=etc/SimpleEventCorrelator.rc --log=log/SimpleEventCorrelator.log
--pid=log/SimpleEventCorrelator.pid
```

You should ensure that the log folder exists and is writeable.

SEC Rules files

By convention, SEC rules have an extension of .sr (sec rules). Perversely, they are referenced from the command line or runtime configuration using the token `conf=<rulename>`. Also by convention, rules files are located in the etc directory, eg

- `\apps\sec\etc\calendar.sr`

The order in which rules files are loaded by SEC at runtime is important, especially in respect of the Xymon Integration described below. For this reason, you will sometimes see that .sr files are named with a numeric identifier. In this way, a lexical sort will load them in the correct order with a token such as `conf=*.sr` in exactly the same manner as Unix system startup scripts. However, due to the way that SEC is started on Windows, this may not be appropriate and you are encouraged to nominate your inputs and rules in the SEC Runtime Configuration file as above.

Xymon Integration ruleset

This should be amongst the first rulesets to be loaded and it is used to enable the sec2pm module. This needs to be initialized with various information about the environment, especially :-

- the IP address of the upstream Xymon proxy through the SEC variable `%XYMSRV`
- the path on the Xymon server from where the configuration can be downloaded through the SEC variable `%XYMDL`.
- the default column name on the Xymon server where results are sent to through the SEC variable `%COLUMN`.

Note that this ruleset also has a dependency on the environment variables `SECHOME` and `SECSVC` passed to it by the environment during startup. These could be assigned directly in the rules file if you wish, but the rule may well then become specific to a single client server whereas the way it stands below, it is portable among many servers.

```
# Load and configure the Sec2Xym module
type=Single
ptype=RegExp
pattern=(SEC_STARTUP|SEC_RESTART)
desc=Load and configure the Sec2Xym module
context=SEC_INTERNAL_EVENT
action=assign %XYMSRV <ip_address_of_xymon_server>;\
assign %XYMDL <path/under_download_directory/on_xymon_server> ;\
assign %COLUMN sec ;\
```

```

assign %LIFETIME 30 ;\
eval %SECHOME ( $ENV {'SECHOME'} );\
eval %SECSVC ( $ENV {'SECSVC'} );\
eval %a (require 'lib/sec2xym.pm'); \
if %a (\
lcall %a %XYMSRV 1984 %LIFETIME %SECSVC %SECHOME 0 0 -> \
( sub { Sec2Xym::init_sec2xym(@_) } ) );\
logonly %a;\
) else ( eval %a exit(1) )

```

NB, the final 2 values (0,0) supplied to the init_sec2Xym() call above are verbosity and debug flags. They can be set and unset during operation using the SEC control ruleset described below.

SEC control ruleset

This ruleset is completely optional, but provides a valuable method for testing and debugging various features. The rules are designed to work in conjunction with a named file in the etc directory and by adding text to this file, SEC is configured to carry out chosen actions. As an example, 01.control.sr may contain the following (note this is not the complete file) :-

```

#####
#####

type = single
continue=dontcont
desc = test upstream Xymon server
ptype = regexp
pattern = ^ping
action = lcall %r %XYMSRV 1984 ping -> ( sub { Sec2Xym::sendToXymon(@_) } );\
logonly %r

#####
#####

type = single
continue=dontcont
desc = update configuration files stored on Xymon server
ptype = regexp
pattern = ^refresh

```

```
action = lcall %r %XYMDL/SimpleEventCorrelator.sr %XYMDL/control.sr
%XYMDL/xymon.sr -> ( sub { Sec2Xym::refresh_config(@_) } ) ; logonly %r
```

```
#####
#####
```

```
type = single
```

```
continue=dontcont
```

```
desc = toggle debug of Sec2Xym module
```

```
ptype = regexp
```

```
pattern = ^debug
```

```
action = lcall %r -> ( sub { Sec2Xym::toggle_debug(@_) } ) ; logonly %r
```

```
#####
#####
```

and by adding a line to the file \apps\sec\etc\control containing the word refresh at the beginning of the line it is possible to instruct the running SEC instance to use the refresh_config() function defined in sec2xym.pm perl module to update the configuration from the centrally stored settings.