SecSIP User Documentation

Alexandre Boeglin, Abdelkader Lahmadi, Olivier Festor

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

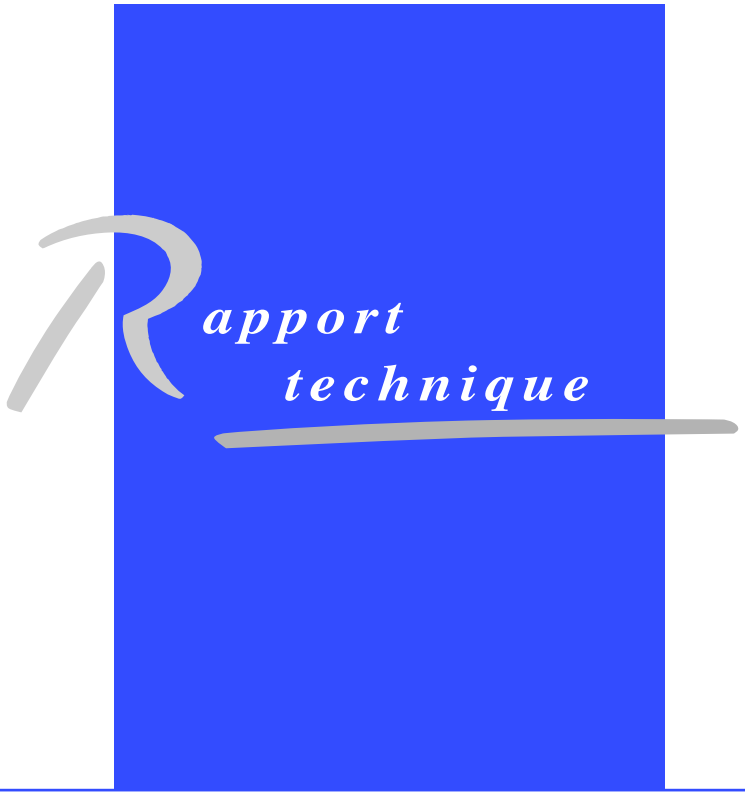
SecSip User Documentation

Alexandre Boeglin — Abdelkader Lahmadi — Olivier Festor

N° 0394

July 2010

Networks and Telecommunications



ISSN 0249-0803 ISRN INRIA/RT--0394--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
NANCY - GRAND EST

SecSip User Documentation

Alexandre Boeglin , Abdelkader Lahmadi , Olivier Festor

Theme : Networks and Telecommunications
Équipe-Projet Madynes

Rapport technique n° 0394 — July 2010 — 20 pages

Abstract: This document is the user's guide of the SecSip software, a stateful SIP firewall. The report contains the installation guidelines, details the usage of the binaries delivered in the distribution, and explores the web-based manager interface that comes with it.

Key-words: VoIP, SIP, firewall, SecSip, configuration, usage

Centre de recherche INRIA Nancy – Grand Est
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

Documentation utilisateur de SecSip

Résumé : Ce document est le manuel d'utilisateur du logiciel SecSip, un pare-feu SIP à conservation d'états. Le rapport contient les directives d'installation, détaille l'utilisation des binaires fournis dans la distribution, et explore l'interface de gestion qui l'accompagne.

Mots-clés : VoIP, SIP, pare-feu, SecSip, configuration, utilisation

Contents

Chapter 1

Introduction

1.1 Introduction

This document is a guide for setting up and running SecSIP, a proactive context-aware protection system for SIP networks.

It assumes some familarity with Linux. The instructions are based on the current release of SecSIP, running on a Ubuntu 10.04 distribution of Linux. It includes details on how to compile, install and deploy SecSIP. It also presents the web interface to author and deploy VeTo rules supported by the SecSIP engine. VeTo rules syntax and semantics are provided in a separate document.

Chapter 2

Installation

SecSIP is written using the C language and distributed in source code form. Therefore, the main steps in installing it are the prerequisite software needed and then compiling the source code. To install prerequisite software, we recommend the use of a recent version of Ubuntu Linux, for which we have verified that all the needed packages are already available. The compilation is performed automatically using configure and Makefile scripts like many other Linux applications. The following subsections cover these tasks in more detail.

2.1 Prerequisites

The recommended platform for using SecSIP is a 32-bit x86 version of Ubuntu Linux, version 10.04 (code named Lucid Lynx); we used such a system in preparing these instructions.

It is possible to use SecSip with other Linux distributions, but one may need to compile some of the prerequisite software from the source beforehand.

SecSip requires a C compiler and depends on a few libraries. In order to compile it, they will need to be present on the build system. All the needed packages also exist in Debian Linux, so the process there should work in almost the same way.

Following is a list of Ubuntu / Debian packages that fulfill these dependencies, along with the version we used:

- **gcc** (4:4.4.3-1ubuntu1) to build the runtime,
- **libpopt-dev** (1.15-1) to parse command-line arguments,
- **libpcre3-dev** (7.8-3build1) to process regular expressions,
- **libpcap0.8-dev** (1.0.0-6) to capture packets,
- **libnetfilter-queue-dev** (0.0.17-1) to capture packets,
- **libcsoap-dev** (1.1.0-16.1) to process SOAP requests,
- **libnanohttp-dev** (1.1.0-16.1) to provide an HTTP SOAP binding, and to serve manager interface through HTTP,

- **libxml2-dev** (2.7.6.dfsg-1ubuntu1) as SOAP requests and responses consist of XML,
- **openjdk-6-jre-headless** (6b18-1.8-4ubuntu3) to build the manager interface files (but any recent JRE should do).

Having a JRE (Java Runtime Environment) is not a direct dependency: it is just used to compress the manager interface javascript files, and a compressed version is already shipped with SecSip. It will only become necessary in the event that the javascript files are modified, or after the `make clean` or `make distclean` commands have been issued.

These packages can be installed using the following command on a Debian-based system:

```
sudo apt-get install libpopt-dev libpcre3-dev libpcap0.8-dev \
    libnetfilter-queue-dev libcsoap-dev libnanohttp-dev \
    libxml2-dev openjdk-6-jre-headless
```

2.2 Compiling

The SecSip source code can be retrieved, either as a package from `http://secsip.gforge.inria.fr/`, or from the public subversion repository of the project, hosted at `https://gforge.inria.fr/projects/secsip`.

To get the sources from the subversion repository, use the following command:

```
svn checkout svn://scm.gforge.inria.fr/svn/secsip/trunk secsip
```

To compile SecSip, simply use the following commands:

```
./configure --with-nfqueue
make
```

If the configure script asks to do so, make a symbolic link using the next command, then re-launch the compilation (`./configure --with-nfqueue` and `make`):

```
sudo ln -s /usr/include/pcap-bpf.h /usr/include/net/bpf.h
```

2.3 Installing

SecSip can be installed unsing the following command:

```
sudo make install
```

By default it will be installed in `/opt/secsip`.

- `/opt/secsip/bin` contains the binary,
- `/opt/secsip/etc` contains the rules file,
- `/opt/secsip/libexec/(api|io|lib)` contains shared libraries used by the binary,
- `/opt/secsip/libexec/manager` contains files of the manager interface.

Chapter 3

The SecSip Runtime

3.1 Command-Line Arguments

The SecSip binary accepts the command-line arguments detailed thereafter:

```
Usage: secsip [-S|--soap] [-H|--http] [-p|--port=0-65535]
[-u|--user-agent=IP:port/max] [-r|--io-receive=pcap:ethX]
[-s|--io-send=raw:ethX] [-d|--rules-files-path=path]
[-f|--rules-file=filename] [-l|--loglevel=0-7] [-6|--ipv6]
[-C|--show] [-b|--background] [-?|--help] [--usage]
```

- `[-S|--soap]` activates the SOAP server, which allows to read and write the configuration used by SecSip via a remote SOAP client,
- `[-H|--http]` activates the HTTP server, which serves files of the manager interface — using this argument implies the activation of the SOAP server as well, as it serves as the backend of the manager,
- `[-p|--port=0-65535]` allows to specify on which port the SOAP and HTTP server will listen (port 10000 is used when this parameter is absent),
- `[-u|--user-agent=IP:port/max]` enables the SecSip User Agent, and allows to specify on what address:port to listen and how many simultaneous connections to allow,
- `[-r|--io-receive=pcap:ethX]` allows to specify how packets should be received from the network (see sections **??** and **??**),
- `[-s|--io-send=raw:ethX]` allows to specify how packets which haven't been filtered should be sent back to the network (see sections **??** and **??**),
- `[-d|--rules-files-path=path]` allows to specify the path to the directory that contain the rules file,
- `[-f|--rules-file=filename]` allow to specify the filename of the rules file,
- `[-l|--loglevel=0-7]` changes the log level, from 0 (nothing) to 7 (noisy),
- `[-6|--ipv6]` makes SecSip use IPv6,

- `[-C|--show]` shows SecSip configuration parameters, then returns control to the shell,
- `[-b|--background]` starts SecSip as a background process, in order to use it as a daemon,
- `[-?|--help]` shows the long version of the help message,
- `[--usage]` shows the short version of the help message.

3.2 Capturing and Filtering Traffic using LibPCAP and Raw Sockets

To capture traffic using libpcap, SecSip has to be invoked like this:

```
/opt/secsip/bin/secsip [...] -r pcap:eth0 -s pcap:eth1
```

In this case, packets will be captured on the eth0 interface, and those that are not filtered will be sent on the eth1 interface.

Note that due to the nature of libpcap and raw sockets, the machine running SecSip MUST NOT be configured to route packets from eth0 to eth1.

Libpcap just captures a copy of the incoming packet, and does not destroy (or drop) the original one, and if a packet were to be routed to eth1 by usual means, SecSip's filtering would be completely ineffective, and each non-filtered packet would be duplicated by the raw socket on the "io-send" interface.

3.3 Capturing and Filtering Traffic using NQFueue

NFQueue is currently the preferred way to capture and process packets, when using Linux. NFQueue allows a userland application to take a decision on a packet (either accept it or drop it).

NFQueue is integrated into NetFilter; when a packet is sent to a NFQueue target, the kernel "blocks" it and waits for a verdict from the application handling the queue, and then drops it or continues processing it accordingly (or drops it immediately if the queue is not handled).

To send packets to a NFQueue, a rule like this is required:

```
iptables -A FORWARD -p udp --dport 5060 \
    -j NFQUEUE --queue-num 1
```

This sends all UDP packets that pass through this rule, and have a source and destination port equal to 5060, to the NFQueue number 1, which SecSip uses.

SecSip must then be launched like this, to handle this queue:

```
/opt/secsip/bin/secsip [...] -r nfqueue -s nfqueue
```

In the NFQueue case, the machine running SecSip can simply be configured as a "classical" firewall, that just delegates some of its decisions to a NFQueue application.

3.4 SecSip as a Daemon

SecSip can be used as a daemon. The `-b` or `--background` command-line argument allows it to be run in the background. To start and stop it when the system boots or halts, a sample init script is provided within the sources at `scripts/initscript`.

Chapter 4

The SecSip Manager

4.1 The SOAP Binding

SecSip provides a SOAP binding, which is used as a backend to the manager, and can also be used independently.

It can be activated using the `-S` or `--soap` command-line argument, and the `-p` or `--port` parameter can then be used to specify on which port the HTTP server will be listening.

The SOAP binding offers three operations:

- **getConfig** allows to retrieve a set of rules,
- **setConfig** allows to modify a set of rules on the firewall,
- **copyConfig** allows to copy a set of rules onto another one.

And SecSip supports three configurations, which mimic NETCONF's:

- **startup** is the set of rules that will be active when SecSip boots, and it is a "carbon copy" of the rules file,
- **running** is the set of rules that is currently active in SecSip,
- **candidate** is an "work in progress" set of rules, that can be made active by copying it over the startup or runnign configuration.

Modifying the **startup** config (or copying onto it), will actually also modify the rules files to reflect these changes. And modifying or copying onto the **running** config will also instantly update the active set of rules in the firewall.

Complete details about the SOAP binding can be found in the `secsip.wsdl` and `secsip.xsd` files of the `libexec/manager/resources/` folder.

4.2 The Manager Interface

The manager interface is a web based config editor, that allows to view and modify SecSip rules. It uses the same XML Schema as the SOAP binding to ensure that the generated rules are valid.

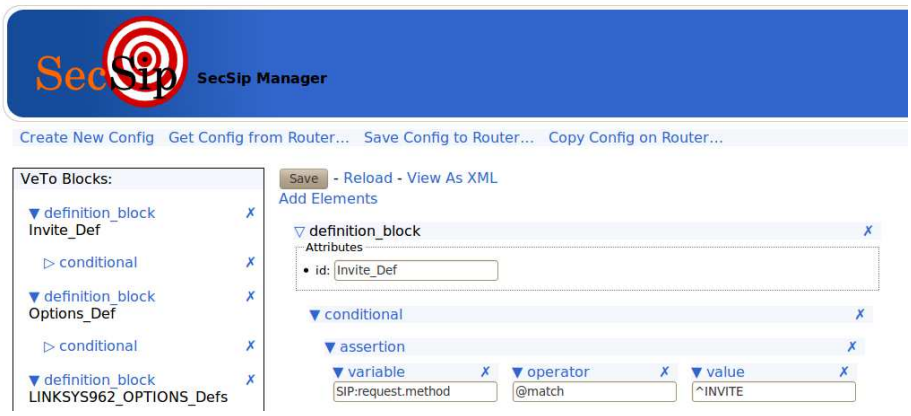


Figure 4.1: The Manager Interface

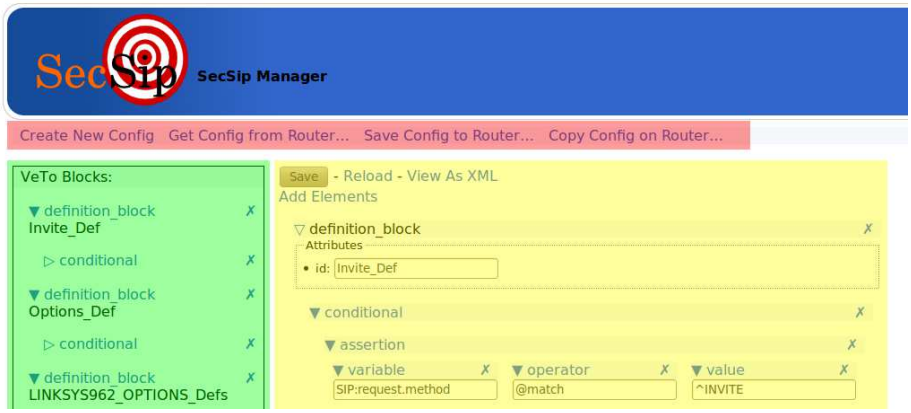


Figure 4.2: The Interface Elements

It can be activated using the `-H` or `--http` command-line argument, and it will listen on the same port as the SOAP binding (10000 by default, unless specified using the `-p` or `--port` command-line parameter).

The manager interface looks like figure 4.1, and is composed of three elements, as seen in figure 4.2:

- the **Menu** (top, in red),
- the **VeTo Blocks List** (bottom left, in green),
- the **Individual Block View** (bottom right, in yellow).

4.2.1 The Menu

The menu is mostly used to trigger SOAP operations. It allows to create a new, blank configuration, to get a configuration from SecSip and start working on it, to send the working configuration to SecSip, and to copy one of the SecSip configurations onto another one, without any impact on, or input from the working configuration.

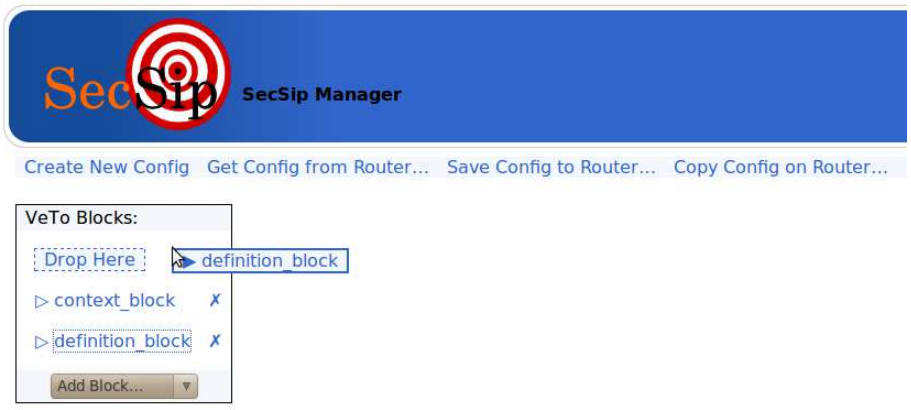


Figure 4.3: The VeTo Blocks List

Just as when using the SOAP binding directly, sending the working configuration as **startup** or copying another configuration onto it will affect the rules file, and those two actions on the **running** configuration will affect the currently active set of rules.

4.2.2 The VeTo Blocks List

The VeTo blocks list (figure 4.3) allows to add blocks using the "Add Block..." drop-down list at its bottom, to delete blocks and block elements by clicking on the 'X' on the right of their label, and to re-organize blocks by draging and dropping their label.

When blocks contain elements that are visible in the list, they can also be expanded and collapsed by clicking on the triangle on the left of the label.

And finally, by clicking a label, the corresponding element will be displayed in the individual block view.

4.2.3 The Individual Block View

The block view allows to visualize and modify an element that has been selected in the VeTo blocks list.

There are three actions available at the top of the block view:

- **Save** allow to save all changes made to the currently viewed element,
- **Reload** allows to discard all changes that haven't been saved yet,
- **View As XML** or **View In Designer** allows to switch between the two available views, the designer view and the XML view.

When the set of rules contains errors, like missing elements or references to non-existent blocks, these will also be shown in the block view, as seen in figure 4.6.

The Designer View

The designer view can be seen in figure 4.4, and it provides the following features:

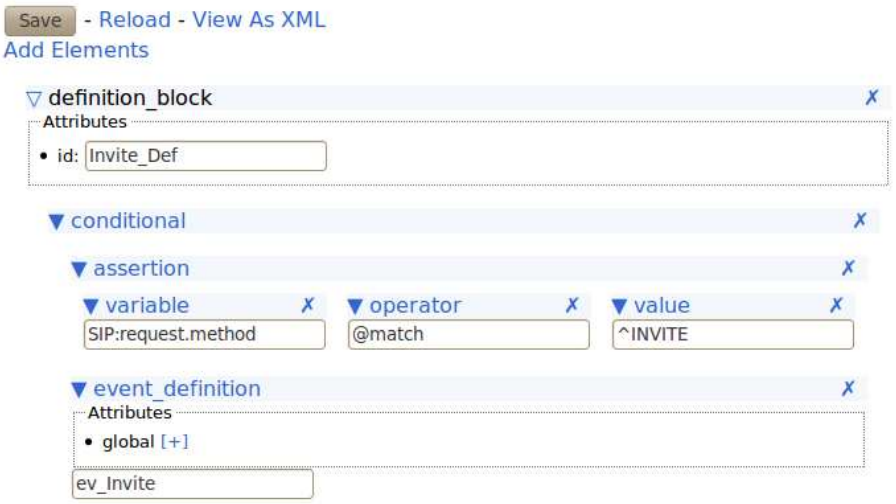


Figure 4.4: The Individual Block View — Designer

Figure 4.5: The Individual Block View — XML

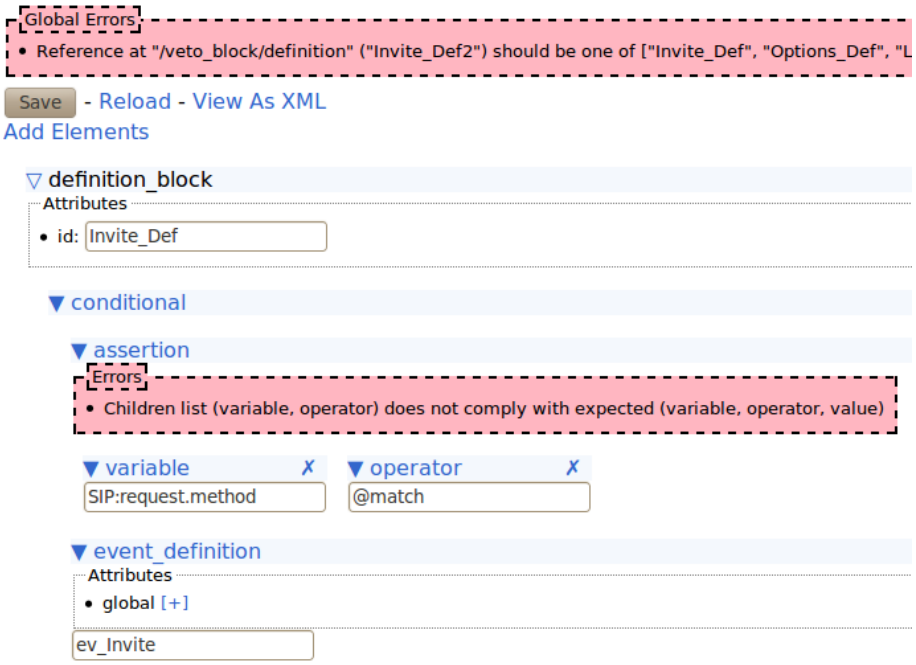


Figure 4.6: The VeTo Block View — Error Messages

- moving the mouse over elements will highlight the innermost one,
- clicking the 'X' at the right of an element's label will destroy this element,
- clicking the arrow at the left of an element's label will expand or collapse this element,
- clicking on an element's label will "zoom" on this element, and make it the newly active element,
- saving changes can be accomplished by clicking the "Save" button, or by pressing the "Enter" key while a field is focused,
- the "Add Elements" button will insert drop-down lists in the view everywhere where an element can be inserted, while keeping the edited element valid, and then the "Hide Add Buttons" button will remove these lists.

The XML View

The XML view, represented in figure 4.5, allows to edit the currently selected element as an XML tree. It is mostly present for debugging purposes, as the designer view is a bit more usable.

When focused, the background of the text area will turn yellow, and when editing is finished and the text area loses focus, it will either turn back to white or red (in the event that the XML is malformed).

Chapter 5

Use Case

In this chapter, we will go through a concrete example of setting up SecSip, and adding rules for processing a vulnerability. We'll assume that SecSip has already been built and installed on the system, following instructions from chapter 2.

5.1 NetFilter Rules

This example will use NFQueues, which require us to setup a queue that will hold the packets for inspection.

The queue can be created using the following commands:

```
iptables -A FORWARD -p tcp --sport 5060 --dport 5060 \
    -j NFQUEUE --queue-num 1
iptables -A FORWARD -p udp --sport 5060 --dport 5060 \
    -j NFQUEUE --queue-num 1
```

Those two rules will send all TCP and UDP packets using the standard SIP ports, that would be routed by this machine, to the queue number 1 (Which is used by SecSip by default).

5.2 Starting Up The SecSip Daemon

Next, the SecSip daemon has to be started, using the following command:

```
/opt/secsip/bin/secsip -S -H -p 8080 -r nfqueue -s nfqueue \
    -l 0 -b
```

The `-S -H -p 8080` arguments will make the SOAP binding and the manager interface available on the port 8080, the `-r nfqueue -s nfqueue` arguments will instruct SecSip to capture and release packets, and the `-l 0 -b` arguments will make SecSip start in the background with no log output.

5.3 Setting Up The Rules

Once SecSip is started, the set of rules provided as an example with its source code will be made active. In our case, we want to setup a set of rules that will protect against a

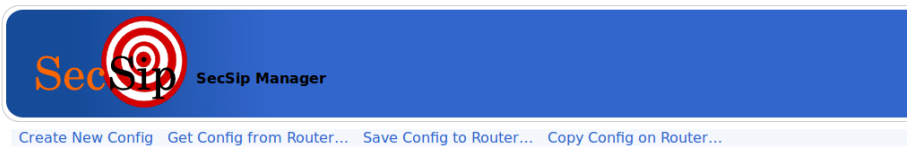


Figure 5.1: The VeTo Block View (No Config Loaded)

specific vulnerability: a buffer overflow in the date field of an OPTIONS message can cause certain SIP hardphones to crash.

First, we will review the set of rules that are currently active. To do so, simply point a web browser to the address of the machine running SecSip, on port 8080, like this: `http://192.168.1.1:8080/`. This should lead to a screen similar to figure 5.1.

Now, it is possible to review the configuration that is currently active on the router by selecting `Get Config From Router...` -> `Running` from the menu. And we can see that it constains the same blocks as the config file in `/opt/secsip/etc/secsip.rules`.

5.3.1 The Definition Block

For the purpose of this example, we will start with a new, blank config. Simply click `Create New Config` in the menu, and an empty VeTo blocks list should appear. Select `Definition Block` from the `Add Block...` drop-down list to create a new block, which should be appended to the list.

We can now start filling the block. First, click on the block name in the list, then on the `Add Elements` in the menu that just appeared on top of the block. What we want to do here, it to add an event that will be triggered each time an OPTIONS message is captured. So, we'll have to make it conditional.

First, add a conditional element inside the definition block, then add an event definition element in the conditional element we just created. The structure of the block is now complete, so we can click on the `Hide Add Buttons` in the menu, and start filling the block's fields.

We will name the block "Options_Def", and the event "ev_Options". We want to match OPTIONS messages, so the (variable, assertion, value) triplet will have to be set to ("SIP:request.method", "@match", "^OPTIONS"). Once everything is filled, we can click the `save` button. In the end, the block should look like in figure 5.2.

5.3.2 The VeTo Block

Adding a VeTo block follows the same steps as the definition block. In our case, the purpose of the VeTo block is to check for a buffer overflow in the date field of each message that have triggered the "ev_Options" event, and drop the packet.

We'll start like the definition block, by adding it form the `Add Block...` menu, then clicking the `Add Elements` option oin the menu.

This time, we'll add an assertion element in the rule element, followed by a drop element (in the same rule element).

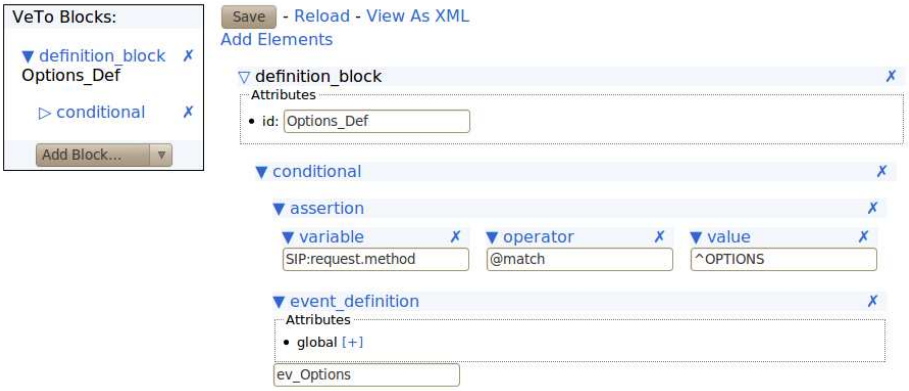


Figure 5.2: The Example Definition Block

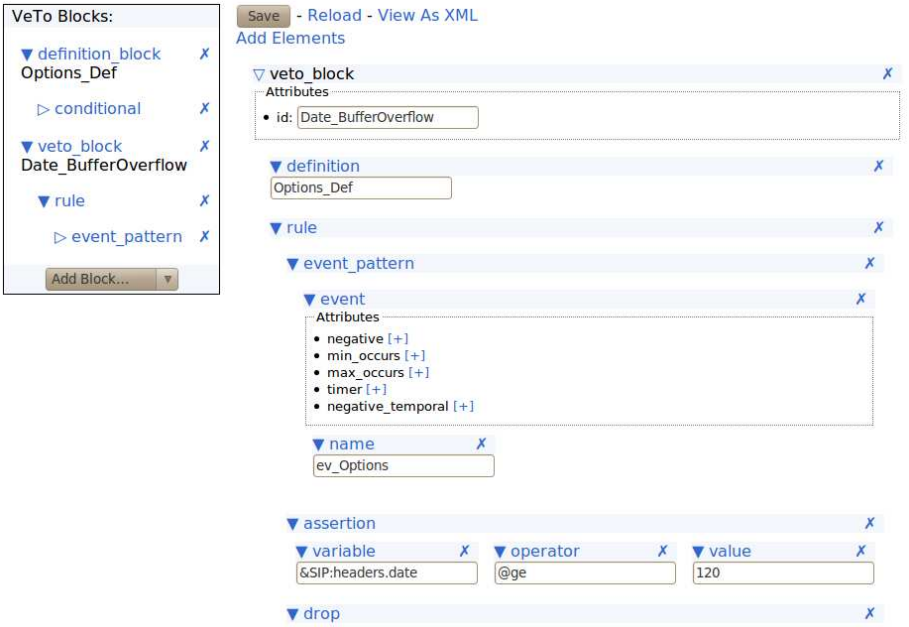


Figure 5.3: The Example VeTo Block

The block name will be set to "Date_BufferOverflow", the definition element should reference our previously created block, so it'll be set to its name, "Options_Def". The name of the event in the rule has to be set to the one that we defined, "ev_Options". Then, the (variable, assertion, value) triplet will have to be set to ("&SIP:headers.date", "@ge", "120"). 120 was chosen, as it is big enough to hold any valid date, and small enough not to cause a buffer overflow.

Once saved, the VeTo block should look like in figure5.3.

5.3.3 Saving The Configuration

Finally, selecting `Save Config to Router...` -> `Running` will send the configuration to SecSip through the SOAP binding, and cause it to use it as its active set of rules. From now on, SecSip will drop any packet that tries to exploit this vulnerability.

It is also possible to make this configuration permanent, by choosing `Save Config to Router...` -> `Startup`. This will send the configuration to SecSip through the SOAP binding, which will then write it to its configuration file. We can check that it has correctly been written in VeTo language form, by viewing the file `/opt/secsip/etc/secsip.rules`.

From now on, each time SecSip is (re)started, it will use this set of rules to initialize its filtering engine.

Centre de recherche INRIA Nancy – Grand Est
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-0803