# Managing signatures for IDS in a distributed environment - A study of a signature management system

David Ormbakken Henriksen


Master's Thesis
Master of Science in Information Security
30 ECTS
Department of Computer Science and Media Technology
Gjøvik University College, 2012

Avdeling for
informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Department of Computer Science
and Media Technology
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

# Managing signatures for IDS in a distributed environment - A study of a signature management system

David Ormbakken Henriksen

2012/06/30

# Abstract

There has been a 36 percent increase in detected computer attacks from 2010 to 2011 according to the latest Symantec threat report. An increase in the total number of attacks from the previous year to the next is a trend that has been going strong for a while. A natural consequence and a parallel trend to this trend is that the human work of intrusion detection becomes more resource demanding each year. This in turn demands from the security practitioners that they become more efficient in their daily work, and routine work is were the biggest potential gain in efficiency can be found.

Intrusion detection systems (IDS) are a big part of intrusion detection work and human IDS-work consists of a lot of routine work. In particular management of signatures. Managing signatures for intrusion detection systems is an ongoing process, which currently is too inefficient. This thesis will show how human efficiency can be improved in this process by the use of a dedicated management system.

# Sammendrag

Ifølge den siste trussel rapporten fra Symantec har det vært en 36 prosent økning i oppdagede dataangrep fra 2010 til 2011. Dette er en trend som har vart i mange år nå og en trend som ser ut til å fortsette de kommende årene. En naturlig konsekvense av denne trenden og en parallel trend til denne trenden er at det menneskelige arbeidet med inntrengingsdeteksjon blir mer ressurskrevende vært år. Dette igjen krever av menneskene som jobber med inntrengingsdeteksjon at de blir effektive i sitt daglige arbeid og rutine arbeid er hvor det største potensialet i økt effektivitet kan bli hentet.

Inntrengingsdeteksjons- systemer (IDS) er i dag en stor del av inntrengingsdeteksjons- arbeid og dette arbeidet består av mye rutinearbeid. Da spesielt i arbeidet som omfatter forvaltning av signaturer. Arbeidet med forvaltning av signaturer er en gjentagende prosess som i dag er for ueffektiv. Denne masteroppgaven vil vise hvordan menneskelig effektivitet kan bli forbedret i denne prosessen ved hjelp av et dedikert forvaltningssystem.

# Contents

# List of Figures

# List of Tables

# Preface

The author of this thesis has since 2009 until recently been working with intrusion detection and incident response at the Norwegian Armed Forces Critical Infrastructure Protection Centre (CIPC) located at Jørstadmoen, just outside Lillehammer city. The thesis topic and much of the work done in this thesis is based on experiences gained from working daily with intrusion detection systems at this centre.

**Acknowledgements**
Firstly, I would like to thank my supervisor, Professor Slobodan Petrovic, for his valuable input during this thesis work and for always being available. Secondly, I would like to thank my previous colleagues at the Critical Infrastructure Protection Centre for taking the time to test the prototype and for answering the questionnaire. In relations, I would also like to thank previous colleagues and friends, not working at this centre any more, for their support and constructive discussions. Lastly, I would like to thank my opponent, Kristian Nordhaug, for his honest feedback on my written report and work.

# 1 Introduction

## 1.1 Topic covered

Looking at recent years research, concerned with intrusion detection systems (IDS), there has been done a lot of research, but only a small amount of this research actually addresses the human side of intrusion detection work [1]. The fact of the matter is that an IDS's successfulness in detecting intrusions depends largely on human interaction in the different phases of intrusion detection work.

The well-known problem with signature based IDS is the huge number of false-positives that they generate [2] [3] [4]. Looking past the technological solutions to this problem (where there has been done a lot of work) and looking at what IDS-operators can do to address this problem, two things come to mind: configuration of network variables and tuning of signatures. Configuring network variables is mainly a one time job, while managing signatures is an ongoing process, which becomes more resource demanding each year. The acknowledged open source IDS (OSIDS) Snort [5] had approximately 3000 signatures available for use in 2005 and in 2010 the number of signatures available had increased to approximately 15,000, which is an increase of 5 times in 5 years and there is nothing that indicates that this won't keep up. We can imaging the additional complexity added when managing more than one IDS, where all the IDS have their own customized signature set. In addition customizing the signature sets requires deep knowledge and skills about the IDS and the defended network [6]. It becomes clear that management of signatures in IDS is a challenging task [7].

## 1.2 Keywords

Information security, usable security, intrusion detection, distributed networks, signature management, security tools, snort, python

## 1.3 Problem description

The ever increasing threat against information assets requires ever more attention from information security professionals [8]. Monitoring the network and analysing abnormalities looking for malicious traffic is a time consuming process that never ends. It is therefore vital that the detection tools being used are adapted to the security practitioner's needs and way of working. The most widely deployed IDS/IPS (Intrusion prevention system) in the world, with over 4 million downloads and nearly 400,000 registered users [5], Snort, suffers from poorly implemented/lack of management of detection signatures. IDS-operators spend an unnecessary amount of time, or do not spend the necessary time on the basis of lack of time, in the signature management process. Snort, like other OSIDS (e.g. Suricata [9], Bro [10]) was not primarily developed with ease of use in mind. The management of rules is therefore text/commando-line based, complex and very manual, which in turn results in a process that is unnecessarily cognitive demanding and

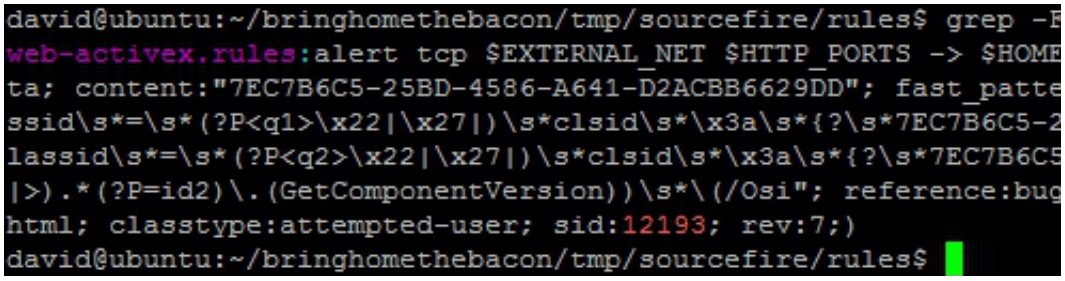


Figure 1: Example of the cognitive load when performing signature syntax lookup

time consuming (see Figure 1 for an example of the cognitive load). The complexity also makes it harder for new operators to gain an understanding, making the learning curve unnecessarily steep. There is no built-in scalability either when it comes to the signature management process. It is today very common to have more than one sensor with an IDS deployed within a company. All these sensors are then usually connected to a centralized server from where they are controlled. It should then not be necessary to do some of the same actions multiplied with the number of sensors, but currently it is. In conclusion, the human work of managing signatures for intrusion detection systems have the potential to be done more efficiently than it is today by introduced a dedicated tool into the IDS signature management process.

## 1.4 The signature management process

The term “signature management process” is used sporadically in research papers (e.g. [7]), but the actual process has not really been explained in any paper. As it is important to understand this process, for the understanding of this thesis, this chapter will try to give an explanation. The following explanation is primarily based on the author’s experiences with IDS work.

### 1.4.1 Explanation of the process

As the different superior phases of human intrusion detection work have already been defined in previous research, it is natural to explain the signature management process in that context. There are four phases in intrusion detection work. The first three where identified and named: monitoring, analyse and response phase by Goodall in 2004 [11] and supplemented with a fourth phase named pre-processing by Thompson in 2006 [12]. The pre-processing phase is the first phase. All actions related to installation, configuration or maintenance of an IDS can be said to belong to this phase. The second phase is the monitoring phase. This is the phase where possible malicious network traffic is detected and alerted. According to Thompson [12], initial analysis should also be looked at as part of this phase, but this paper, for simplicity sake, does not do that. After some network traffic has been detected as suspicious the IDS-operator moves into the analysis phase. The analysis phase is where the decision whether the incident is real (true-positive) or false (false-positive) is to be taken. According to Goodall [11] an IDS-operator will only enter the response phase if a real attack has been identified. What Goodall failed to realize is that even a false-positive may need some kind of response, which becomes clear when you take the signature management process into account. Reading Thompson’s research [12],

the false-positive response can be understood to be part of the pre-processing phase, but again for simplicity sake, response to a false-positive is in this paper part of the response phase. Having explained the different phases, this chapter will precede with a simple explanation of the actions taken in regards to the signatures in the different phases. All these actions are what ultimately makes up the signature management process (see Figure 2 for a graphical representation). From Figure 2 it is possible to see that the IDS signature management process is actually two somewhat independent processes, but this thesis will keep on referring to it as one process.

#### 1.4.1.1 Pre-processing phase

Before the IDS becomes active on the network for the first time, it is in this phase important to disable the signature sets that are superfluous, violate the privacy of the users or that detect traffic patterns that don't violate the company's policies. This could for example be signature sets that detect the use of instant messaging services, torrent services or VoIP. This will reduce the amount of alerts in the monitoring phase and let the operators focus on the incidents that really matter. It is in this phase common to disable/enable signature sets and not disable/enable single signatures, as that is to time demanding. IDS placed in different parts of the network may have different signature sets enabled or disabled depending on the policy for that network and depending on other detection systems that are already in place. After finishing this task, the IDS-operator has to distribute the correct signatures too the corresponding IDS. Disabling/enabling signature sets is more or less a one time task, while writing custom signatures and downloading and updating signatures are the never ending tasks of this phase.

#### 1.4.1.2 Monitoring phase

The monitoring phase doesn't involve any human interaction in regards to signatures.

#### 1.4.1.3 Analysis phase

When an alert is produced the IDS-operator needs to know why. In all simplicity the alert was produced because the signature matched some traffic passing trough the network, but the operator needs to know if that traffic is harmful or could be harmful to the company's assets. The first thing the IDS-operator needs to look at, to make it clear, is the relevant signature syntax. From the signature syntax the operator can usually extract the following information: a short description of the signature, internet references, the actual rule syntax that triggered the event and a signature category. The operator will further use this information in correlation with information like network packet data and netflow data to be able to make a decision. The more information, the easier it will be for the operator to make his decision. As the signature based actions in this phase are done for each alert, it is safe to say that they are a daily part of IDS work.

#### 1.4.1.4 Response phase

If the IDS-operator finds the alert to be false-positive, the operator may decide: that the signature is not good enough and thereby disable it or modify it, that the signature is good enough but what it detected is not of interest and suppress/threshold it or that the signature will not trigger on the same traffic again and therefore no action is needed. Suppression gives the operator the possibility to block IP-addresses or IP-ranges for that signature. Thresholding lets the operator choose how "aggressive" the signature should be. He could then for example choose to extend

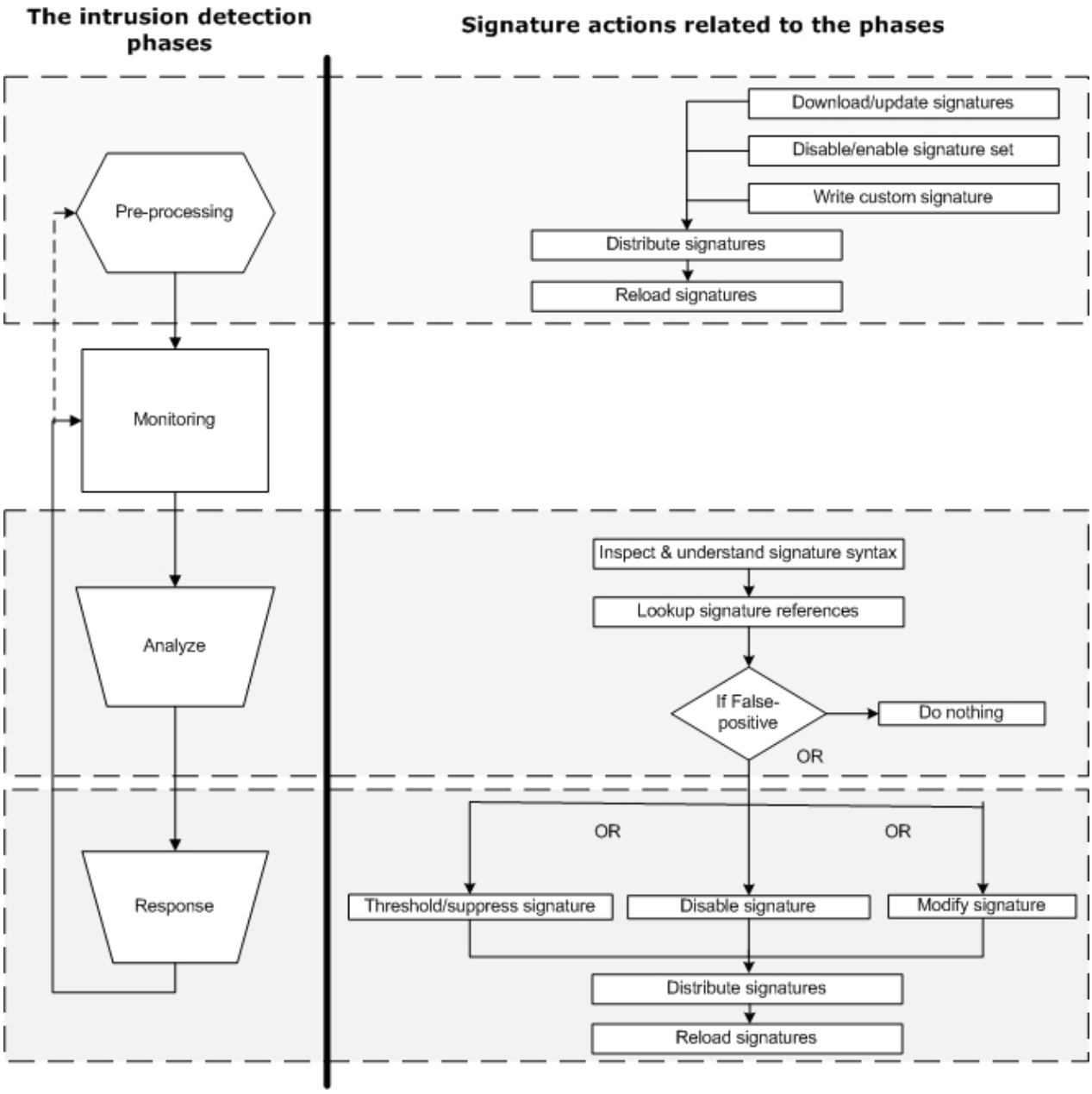


Figure 2: The signature managment process

the amount of traffic allowed to pass before the same alert triggers again. If the operator chooses to respond he also has to choose which IDS he wants to apply the response to and he has to distribute the changes to desired IDS. Bearing the problem of a huge amount of false-positives in mind, it is easy to see that an IDS-operator will often perform signature based actions in this phase.

## 1.5 Justification, motivation and benefits

The ones that have worked or work daily with IDS will at least agree upon one thing, and that is that there is a lot of routine work involved in the day to day operations of an IDS. The quality of this routine work depends solely on the skill set of the individual operator, but the time spent doing this work does not. Even though it is obvious, for the IDS-operators, that the signature management process can be improved, it is a matter of taking time to improve the process versus using that time to do the process. As mentioned earlier time is already something security practitioners are in short supply of, which means that they implement partial solutions, but don't have the time to do it right. Through interviews conducted with 9 IDS-operators in 2008 [1], the time consuming process of signature management is mentioned by many of the interview objects as an obstacle to the usefulness of IDS. The reason was that reducing the occurrences of false-positive alerts where partly done by filtering signatures and filtering signatures was very time consuming, time they already had too little of. The need for more efficient signature management has also been requested in many forum posts on the Internet over the years (e.g. [13], [14] and [15]).

This research aims at showing how much time can be spared by improving this process and at giving the IDS-operators a tool, which if taken into use, could make their life easier. The motivation behind this research is primarily based upon the authors first-hand experiences from working daily with different intrusion detection systems in a complex environment.

## 1.6 Research question

Based on the previous discussion, the following research question has been defined:

- To what extent would a graphical tool improve human efficiency in the IDS signature management process, in terms of time, compared to the existing manual method?

## 1.7 Claimed contributions

This master thesis will provide the following contributions:

- Definition of the IDS signature management process.
- Definition of a method to create a scalable and efficient system for managing signatures for Snort and other OSIDS like Suricata that use the same signatures.
- An implementation of a usable prototype based on the method mentioned above.

# 2 Related work

As stated earlier there has not been done much research that addresses the human side of intrusion detection work. It is therefore very little research that specifically targets the signature management process, as this is primarily a human based process in IDS work. The majority of the research that we have found concludes that introducing tailored tools into IDS work is a solution that has the potential to improve the human efficiency in the targeted process, for example:

[12] states that one of the most challenging tasks with intrusion detection work is that the IDS-operators have to manually integrate information from a variety of tools and resources which adds a cognitive burden on the operators. A complete tool could lighten some of this burden. [1] points out that much research has focused on improving the usability perspective by providing tools with a visual interface for the monitoring and analysis phase of intrusion detection work, but that the pre-processing and response phase have been neglected in this area. [7] acknowledges the need to automate the IDS signature management process by the use of a tool. The reason is that this process, if a larger signature set is introduced, would be too inefficient if done manually.

When it comes to creating tools, previous research states that graphical user interfaces are preferred ahead of the command-line interface (CLI), but only if the graphical interface completely removes the need to interact with the CLI to complete the desired task. In other words, as long as all the details needed are being presented [12] [16].

The next section will explain the related work with regard to existing tool-support for IDS-operators in the signature management process.

## 2.1 Existing open source tools

On the official support Website for Snort there are currently listed two dedicated signature management tools. They are: Oinkmaster [17] and Pulled Pork [18].

Oinkmaster is the one of the two listed above that has been around the longest as it was released in 2001. This third party tool is programmed in Perl and is based on the command-line interface. Current version is 2.0. Oinkmaster supports the update and enables/disables action in the pre-processing phase, and the change and disable/enable rule action in the response phase. Currently it is not being maintained.

Pulled Pork is an official tool in the way that it was developed and is maintained by one of the members of the Sourcefire team. It was released in 2008/2009. As Pulled Pork is currently maintained, many environments that previously used Oinkmaster have now started using this tool instead [19]. Pulled Pork supports the same actions as Oinkmaster in the signature management process, which are: update rules, enable/disable rule-set and enable/disable/change rule. Pulled Pork is also programmed in Perl and is command-line based. The current version is 0.6.1.

An even newer script, which is not on the official support website, was released in 2011. It

is called Polman [20] and is a third-party contribution. In contrast to Oinkmaster and Pulled Pork, Polman acknowledges that many environments have more than one sensor and therefore have the need for custom rule-sets for different sensors, instead of using the same rule-sets on all sensors. Polman has the ability to disable/enable rule-sets and rules for different sensors. It does this by creating a new database for each sensor, which contains the same rules, but not necessarily the same configuration. Polman supports: update rules, enable/disable rule and rule-set. This tool is as the two others written in Perl and is purely command-line based. The latest version is 0.3.2 and this tool currently seems to be maintained.

Also worth mentioning is an application called Snorby [21]. Snorby is a graphical user interface based tool created to support the monitoring phase of intrusion detection work, which from Figure 2, one can see is not a phase that contains any signature actions, but Snorby has partial support for looking-up signature syntax and signature references, which are signature actions in the analysis phase. The developer behind Snorby is aware of the need for better signature support (like tuning), as it was requested on his feedback forum for over a year ago, but as this tool was not initially created to support the signature management process, he replied that it was a hard task to implement it now. According to his Web page it is a feature to come, but the progress is going slow. Current version of Snorby is: 2.5.1.

## 2.2 Related work conclusion

Currently there are no tools for Snort that supports the whole signature management process, nor are there any tools that provide a graphical user interface for this process. As mentioned in the introduction, many IDS-operators create partial solutions and the tools currently available can be looked at as such solutions. Figure 3 gives an overview of the actions in the signature management process that are currently supported.

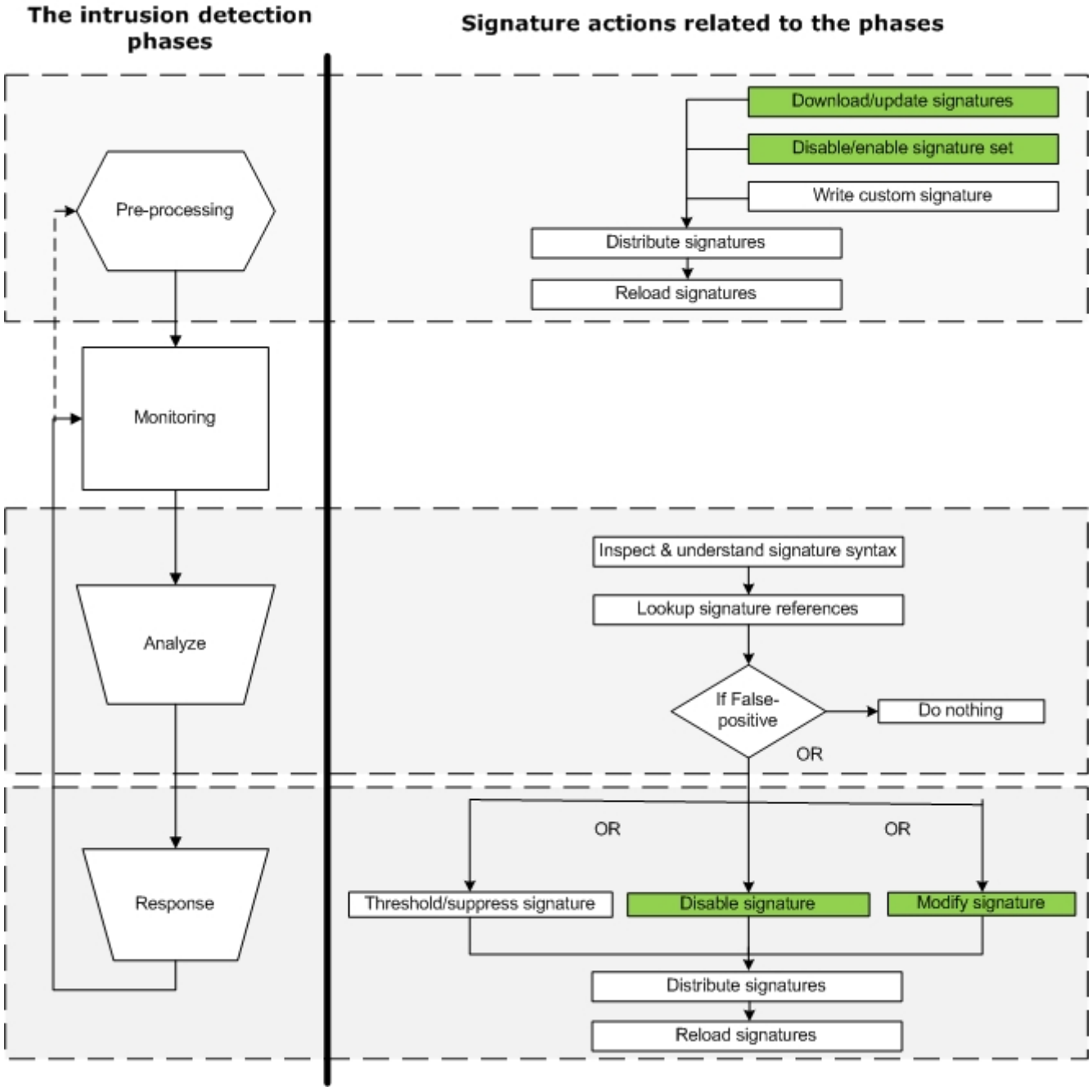


Figure 3: Actions in the signature management process currently supported (green colour)

# 3 Methods

This thesis has been a quantitative research study based on existing methods. In all simplicity the course of events in this study has been: Identify the different actions of the signature management process and then uncover related work related to this process. Further this information along with the authors experience and input from previous colleagues has been used to develop a prototype, which in turn has been used for testing.

To test for improved efficiency, keystroke-level modelling has been used. In addition the prototype has been installed at the Critical Infrastructure Protection Centre, where the IDS-operators working there have tested it and provided feedback through a Web-based questionnaire. The main reason for doing this second testing was to get more information that could supplement the findings in the keystroke-level modelling experiment, but also to get feedback on the overall usability aspect, as efficiency is not the only factor of usability that will be taken into account when a company/person chooses whether to take a tool into use or not.

## 3.1 Keystroke-level modelling

As mentioned above, keystroke-level modelling (KLM) was used in this thesis to test for improved efficiency in terms of time. The reason for using this method is that it is fast, relatively simple and that it measures exactly what we want to measure (time used in human-computer interaction). This method also produces results that are easily comparable. The KLM framework is well known and has been used by many researcher throughout the years. The time estimates that are provided within the framework are well tested and we can therefore assume that they are reasonable accurate. The paper [22] was used for guidance when executing the experiment.

## 3.2 Questionnaire

When using a questionnaire to measure something, the most difficult part is to frame the questions with regard to validity. The questions used in this questionnaire were therefore taken from the article [23] that offers a framework called USE-questionnaire where an assortment of already framed questions/statements for measuring usability can be chosen among. This framework also proposes an answering scale to be used together with these questions and this scale where therefore used together with the selected questions.

# 4 The signature management system

This chapter will describe the requirements set, the design choices taken and the actual implementation of the signature management system.

## 4.1 System requirements

The main purpose of this system is to make the process of signature management more efficient. The signature management system is to be based on the signature management process outlined in Figure 2 and explained in Section 1.4.

### 4.1.1 User interface

The system shall provide a graphical user interface (GUI) to the user. The main purpose of this GUI is to act as a layer of abstraction, hiding some of the underlying complexity and at the same time reducing the cognitive load put on the users in this process. The GUI is key to improving efficiency, so the design of it needs to be thoroughly thought through. User navigation should be as effective as possible and user assistance should be offered in form of text based instructions where deemed appropriate. Most of the actions that are part of the signature management process should be able to be carried out through the GUI. In other words not all actions need to be implemented as this is a prototype. For actions that could be sensor specific, the user shall be presented with a drop down list from where he has the option to choose which sensor to apply the action to. For example the action of disabling a signature could be a sensor specific action and for this action the user should then be presented with a "choose sensor" option. As the signature management process can be said to be two independent processes, the system should present the user with two different views. One view that is signature set based and supports the pre-processing phase and one view that is single signatures based and supports the analysis and response phase. The system should also have a sensor view and have functionality to add sensors into the system. Considering the amount of information this system will handle, some kind of information filtering needs to be implemented in the GUI. The system should offer a possibility for free text searching and for information sorting.

### 4.1.2 Scalability and Performance

The system should be able to scale to support an unlimited number of sensors. Depending on the number of sensors some loss in performance must be expected, but this system should be able to handle at least five sensors without any notable loss in performance for the user. Further the system should be able to support an unlimited number of signatures, but maximum 20.000 signatures without any notable loss in performance for the user. Notable loss in performance is in this context meant as more than 5 seconds in which the user needs to wait before he can continue with his use of the system.

### 4.1.3 Communication

The system must be able to connect to sensors so that it can deliver new/updated signatures and send commands, so that it can tell the IDS software on that sensor to take the new signatures into use. This system must also be able to download signatures from specified sources and regularly update them afterwards. When updating it is important that the system keeps track of changes so that changes made by the user do not get overwritten. The communication of distributing signatures and updating signatures could take place on an internal network only or over the Internet. All communication should be logged by the system. The GUI should present the user with a distribution and updating button. This way the user can manually update the system and sensors when needed. The system should also make sure that signatures are automatically updated and distributed, so that the sensors have the newest signatures at all times without any human interaction.

### 4.1.4 Compatibility and environment

The system is expected to be installed on a centralized server, either dedicated or existing, within a distributed network, from where it can communicate with all relevant sensors within the network and the internet. The system shall be compatible with the signature management process of Snort. This will also make the system compatible with Suricata, as Suricata uses the same signatures as Snort.

### 4.1.5 Previous work

When designing the system previous work should be taken into consideration. Either previous work could be implemented as a part of this system or ways of doing things in previous work could be done the same in this system.

## 4.2 Design choices

This section will describe the choices made during the design phase of this system. The choices that were made resulted in a very portable system with few dependencies and easy installation.

### 4.2.1 Platform

The Snort IDS was originally developed for Unix-like platforms (e.g. Linux and BSD). Although Snort now is compatible with Windows, most third-party support tools have been and are being developed for Unix-like platforms. For this system the platform of choice is Linux and the operating system (OS) to be developed in is Ubuntu as this is the OS the author is the most familiar with. This means that the system can be installed on all Linux platforms and most of the other Unix-like platforms.

### 4.2.2 Programming language

Many programming languages can be used to implement this system (e.g. Perl and Ruby), but the programming language of choice is Python. This is the programming language the author has the latest experiences with and the best knowledge of. There will also be used some JavaScript on the client side of this application, see Section 4.2.3.

#### 4.2.3 User interface

This system will have a Web-based user interface. The advantage of a Web-based interface is that it only needs to be installed once and is thereafter easily accessible by all computers on the internal network and the Internet independent of the computer's OS. To display the data to the user the DataTables plug-in [24] built on the jQuery Javascript library [25] will be used. This plug-in provides an easy way for retrieving data for further organizing and displaying that data in a Web based dynamic table. It also has built in column sorting and searching, which is a requirement of this system. With performance in mind this system will use server-side processing.

#### 4.2.4 Web framework

Since this system is to be programmed in python and demands high performance Tornado [26] is the web server/framework to be used. Tornado is a relatively new web framework written in Python. It is know for high performance and is therefore often used for real-time applications [27].

#### 4.2.5 Database

SQLite version 3 [28] will be used as the database to where this system will organize and store all of its detection signatures and other related information. SQLite is a very fast and minimalistic database. The advantage of this database is that it does not require any installation to work and that it is just a single file on the hard-drive, which makes backup of the system very easy.

#### 4.2.6 Communication

For distribution of signatures this system will use Rsync [29]. OpenSSH (Secure Shell) [30] will be used for sending commands. Both of these applications are installed in Ubuntu by default.

#### 4.2.7 Architecture

Figure 4 shows the architecture of this system. The arrows in this figure indicates the flow of data. The architecture is a result of the specifications given in Section 4.1 and the design choices made above. This system will mainly consist of three scripts:

- update.py
- distribute.py
- web.py

The reason for dividing them up this way and not include all code in one script is the requirement that states that the action of updating and distribution of signatures should be done automatically and be possible to trigger manually. Separating them this way fulfils that requirement. The scripts purpose and functionality will be discussed in the next sections.

##### 4.2.7.1 update.py

For downloading of signatures the already existing script Pulled Pork was considered to be used, to save some time in the development phase. Unfortunately this script has some shortages that force the need to customize it for use with this system, which seems to be just as time consuming. Some of these shortages are that Pulled Pork does not bind the rule set name to the signatures

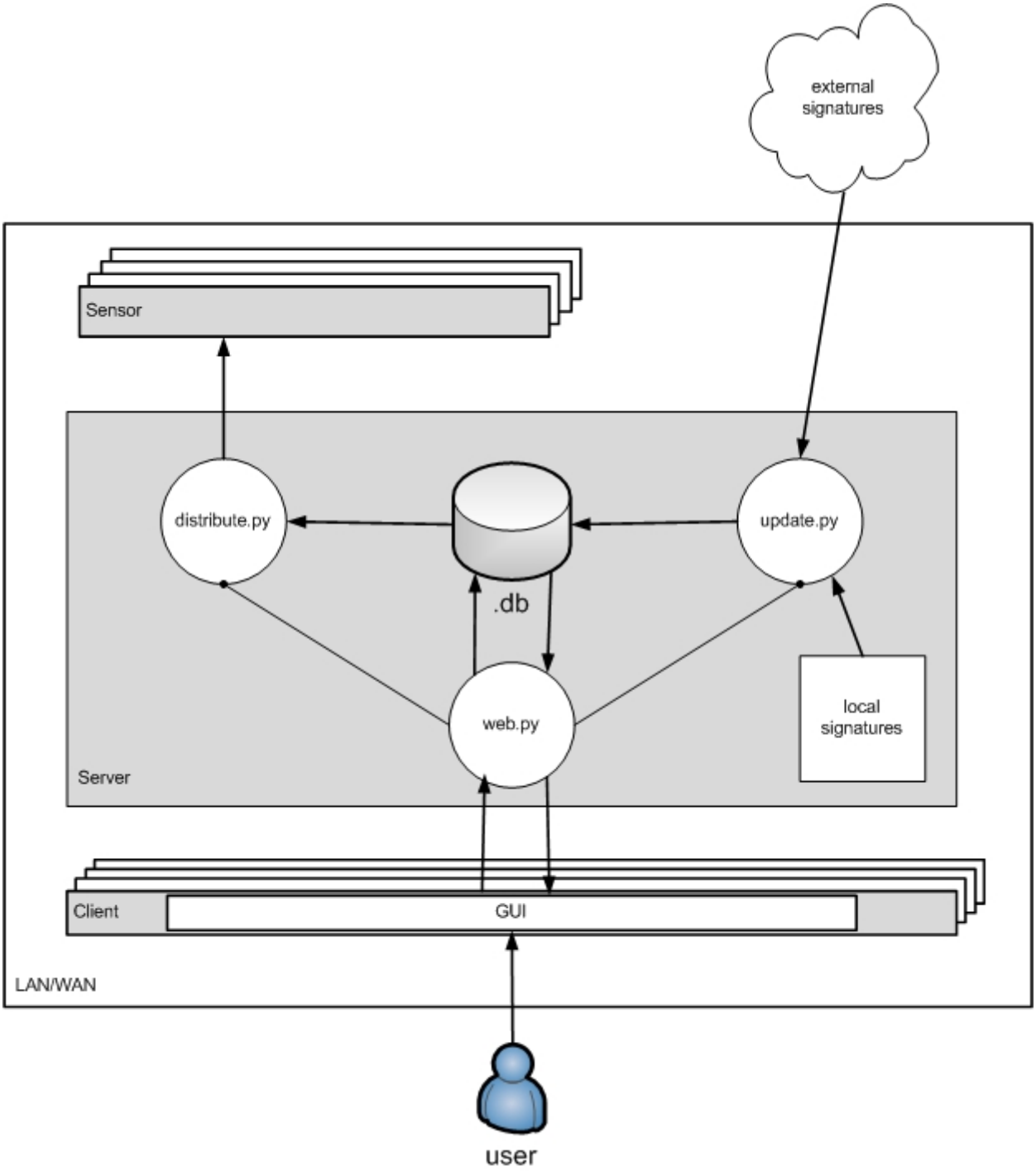


Figure 4: System architecture

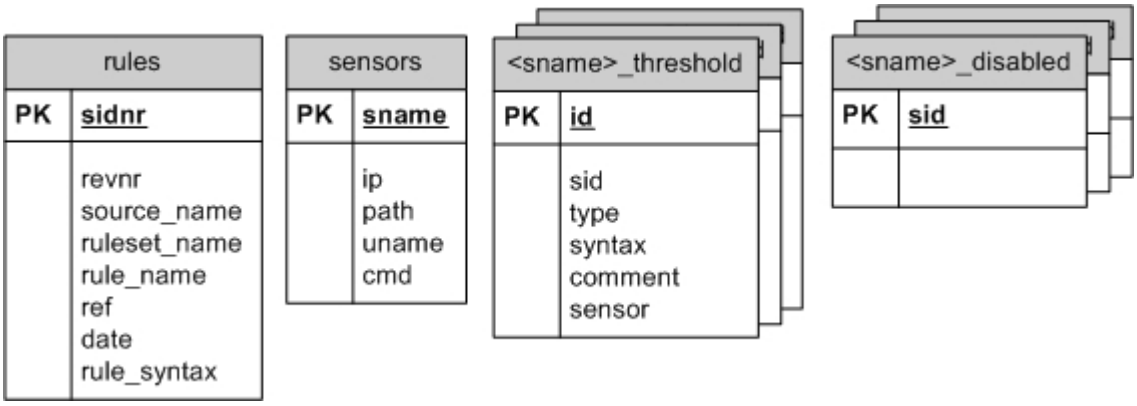


Figure 5: Database tables

(which only leaves class type) and does not keep the downloaded files on disk after use (compatibility with Snorby among other things). The update script will therefore manage downloading, extraction and database insertion of signatures from local and external sources. All actions this script does will be logged and signature archives, if possible, will be checked for freshness by the use of md5 before potentially downloading them (feature of Pulled Pork).

#### 4.2.7.2 distribution.py

This script will handle distribution of signatures and other relevant files to sensors. Distribution.py will extract and read information from the database to know which sensors are supposed to have which signatures and which tuning rules. The extracted information will be stored in sensor specific files on local hard drive before being distributed. All actions this script does will be logged to a file.

#### 4.2.7.3 web.py

This is the main script that will run on top of Tornado and provide for communication between the user, the database and the two other scripts. This script will extract data from the database and pass it along to the jQuery script running in the user's browser. This script will also handle all actions available for the user, such as insertion of data into the database, manipulation of data in the database or running the distribution or update script.

#### 4.2.7.4 Database

Figure 5 shows the tables with values that will be used in this system's database. There will be one table containing signature information, one table containing sensor information and two tables for each sensor that will contain tuning information. The last two tables will be generated at the same time a new sensor is registered by the user and deleted at the time the user deletes the corresponding sensor.

## 4.3 Implementation

To be able to prove to what extent this system would improve efficiency a prototype was developed.

The implemented system meets the specifications outlined in Section 4.1 and is designed according to the choices described in Section 4.2. In the implementation phase of the update.py script there where some choices that needed to be made that were not considered earlier in the process:

- Signatures that are disabled by default (commented out) in the signature sets downloaded from source will be ignored and thereby not imported into this systems database. There is probably a good reason why these signatures already have been disabled and with performance in mind the choice to ignore them was made.
- When two signatures have the same identification number (SID) the system will keep the signature with the highest revision number (REV) in the systems database. Optimally the system should keep all revisions of a signature so that the user can see what changes have been made in a new version. Again as this is a prototype and because of performance this system only keeps the latest revision of a signature in its database.
- If a signature has been disabled by a user and then that signature is updated to a new revision, it will not be enabled again in this system. This choice is based on the choice above. As the user has no option to see the previous revision of the signature (the reason why it was disabled) this system is not enabling disabled-signatures if a new revision is downloaded.

The code for the update.py script can be found in Appendix B, the code for the distribute.py script is in Appendix C and the code for the web.py script can be found in Appendix D. The code for the configuration file and for one of the templates made can be found in Appendix E and F respectively. The whole system can be viewed and downloaded from this thesis repository found at GitHub [31].

### 4.3.1 Graphical user interface

As the graphical user interface (GUI) of this system is a big part of this thesis, with regard to efficiency, this section will explain the implemented user interface.

The implemented GUI has an always present navigation bar at the top from where the user can choose to navigate to one of four views or by the click of the mouse choose to initiate the distribute or update signature process. Figure 6 shows an example of the type of report the user will be presented with after initiating the update process from the GUI. The four different views are: rule, rule-set, sensor and advanced tuning, and they will be explained in the next sub sections.

#### 4.3.1.1 Rule view

The rule view, as pointed out in the specification, is designed to support the analysis and response phase of intrusion detection work. All actions related to these phases, except for option to modify signature, are therefore available to the user trough this view. Figure 7 shows how this view looks to the user, in use, through the browser. All vital signature information is organized in columns. By default, the information in the table is ordered by newest signature by date first, but the user

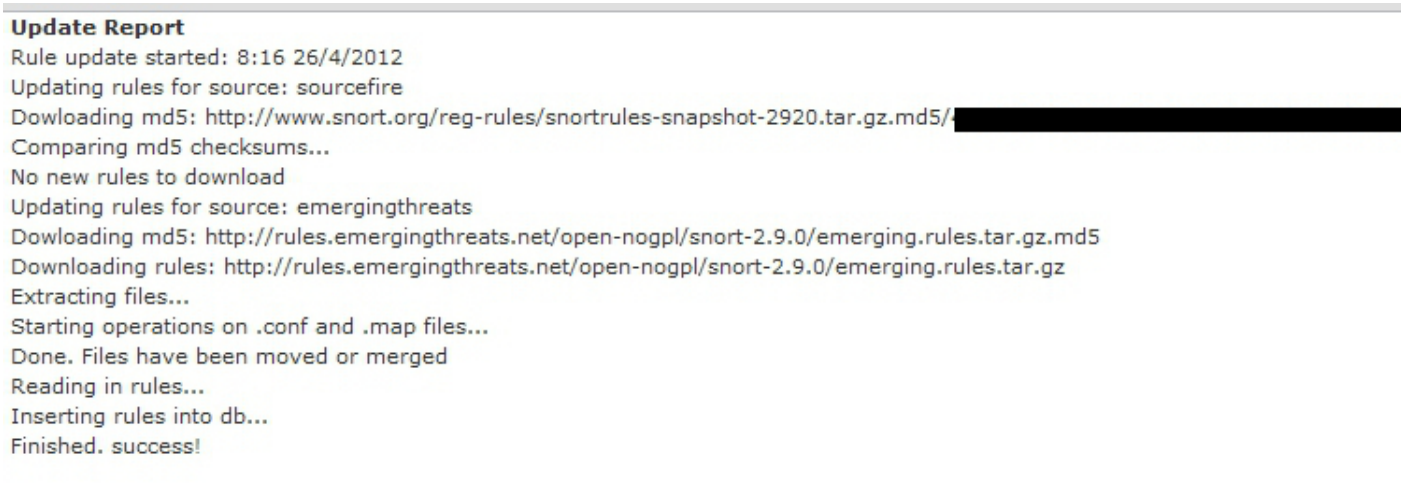


Figure 6: GUI: Update report

has the possibility to order the column of choice either descending or ascending. The user also have the option to perform free text search trough all columns by the use of the input field in the upper right corner. When working with large signature-sets that is the best way to go. Further this view has built-in click-able reference links that will open a reference in a new window/tab if clicked. The plus/minus symbol to the left of each signature lets the user view/hide the whole signature syntax. The reason for hiding the syntax is that the view would be to complex if always displayed. The syntax if also not fetched from the database before the user initiates it, so it has a performance aspect to it too. The status information column displays all registered sensors and the fill colour green or red informs the user that the relevant signature is either enabled (green) or disabled (red) on that sensor. The sensor could also have a yellow exclamation mark next to it which means that on that sensor, a signature has either an active suppression or thresholding rule affecting it. All signatures have a check box. By checking a box and choosing disable/enable the checked signature will be disabled/enabled. Choosing suppression (S...) or thresholding (T...) the user will be presented with a new window with input fields, see Figure 8. This window also presents the user with a help button, which if clicked will guide the user trough this process.

#### 4.3.1.2 Rule-set view

The rule-set view looks very much like the rule view. The big difference between these two views, as the names state, is the displaying of single signatures versus signature-sets. Figure 9 shows this view in action. This view is primarily meant to be used in the pre-processing phase of intrusion detection work. When setting up the IDS for the first time it makes more sense to disable signature-sets instead of single signatures, and that is what this view offers to help the user with. This view could have been integrated with the rule view, but to not complicate and to reduce cognitive load on the user it became a view of its own. In addition to displaying signature-sets, this view has a new column called "number of rules" which displays the total count of signatures in a signature-set. Further, this view also has the status information column, but in

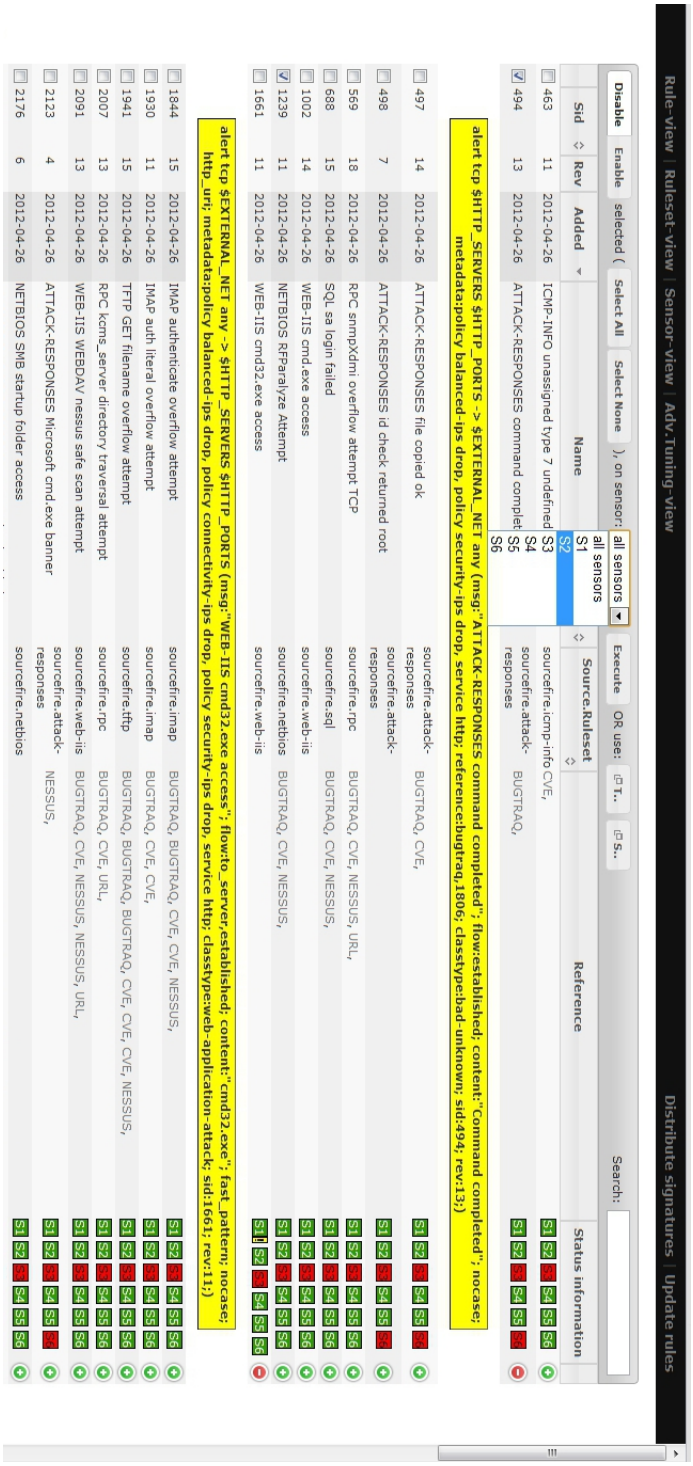


Figure 7: GUI: Rule view

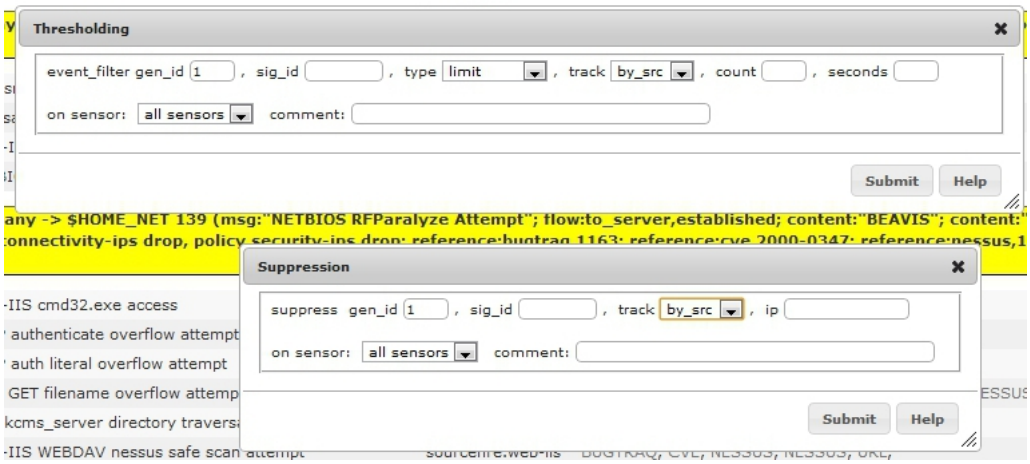


Figure 8: GUI: Suppression and thresholding option in the rule view

this view each sensor will have a disabled signature count in red, if not the whole signature-set is disabled, but only some of the signatures belonging to it.

#### 4.3.1.3 Sensor view

The sensor view is the user's sensor management view. In this view, the user can register, modify and delete sensors. When registering a sensor the user will be asked to specify different options that the system will need in order to distribute the signatures. The sensor name the user chooses is also the name that will appear in the status information column in the rule and rule-set view. This view is displayed in Figure 10.

#### 4.3.1.4 Advanced tuning view

This view displays all active suppression and thresholding rules and gives the user information on what sensors and signatures they affect. When creating a rule the user has the ability to give a reason why that rule was created, this view displays that comment. Further, this view gives the user the option to remove rules listed in this view. This view is displayed in Figure 11.


Figure 9: GUI: Rule-set view

Figure 10: GUI: Sensor view

Figure 11: GUI: Advanced tuning view

# 5 Experiments and results

This chapter will present and discuss the results obtained from the experiments conducted in this thesis. This chapter will also describe the execution of the experiments.

## 5.1 Keystroke-level modelling

By the use of the keystroke-level model (KLM) execution times can be estimated for specific task scenarios. To measure the efficiency of the newly developed system against the already existing methods of doing things, 5 task scenarios were chosen from the signature management process and applied to the KLM (see Figure 12). The chosen task scenarios were: Inspect signature syntax, lookup signature reference, disable signature, distribute signature change to sensor and load signature change on sensor. From Figure 12, one can see that the chosen task scenarios constitute a common series of actions that follow each other in the analysis and response phases of the signature management process. In addition, these are the actions most frequently performed by the IDS-operators. For each scenario, the keystroke-level actions (operators) have been listed and the execution time for all listed actions has been added up and compared. Further, the execution times for all scenarios have been added up and compared.

| Operator | Description | Execution time |
|---|---|---|
| K | Keystroke | 0.2 sec |
| P | Point with mouse to a target on the display | 1.1 sec |
| B | Press or release mouse button | 0.1 sec |
| BB | Click mouse button | 0.2 sec |
| H | Move hands to keyboard or mouse | 0.4 sec |

Table 1: Operators and execution times used

All operators and execution times used in this experiment are listed in Table 1. The execution time for K is based on the assumption that IDS-operators are above average typists and therefore type an average of 55 words per minute.

| Operator | Description | Execution time |
|---|---|---|
| M | Mental act of routine thinking or perception | 1.2 sec |
| W(t) | Waiting for the system to respond (time t must be determined) | - |

Table 2: Operators and execution times not used

Table 2 lists the operators available in the KLM but not being used in this experiment. Choosing how many mental acts of thinking and where they appear is a really difficult task, and the fact

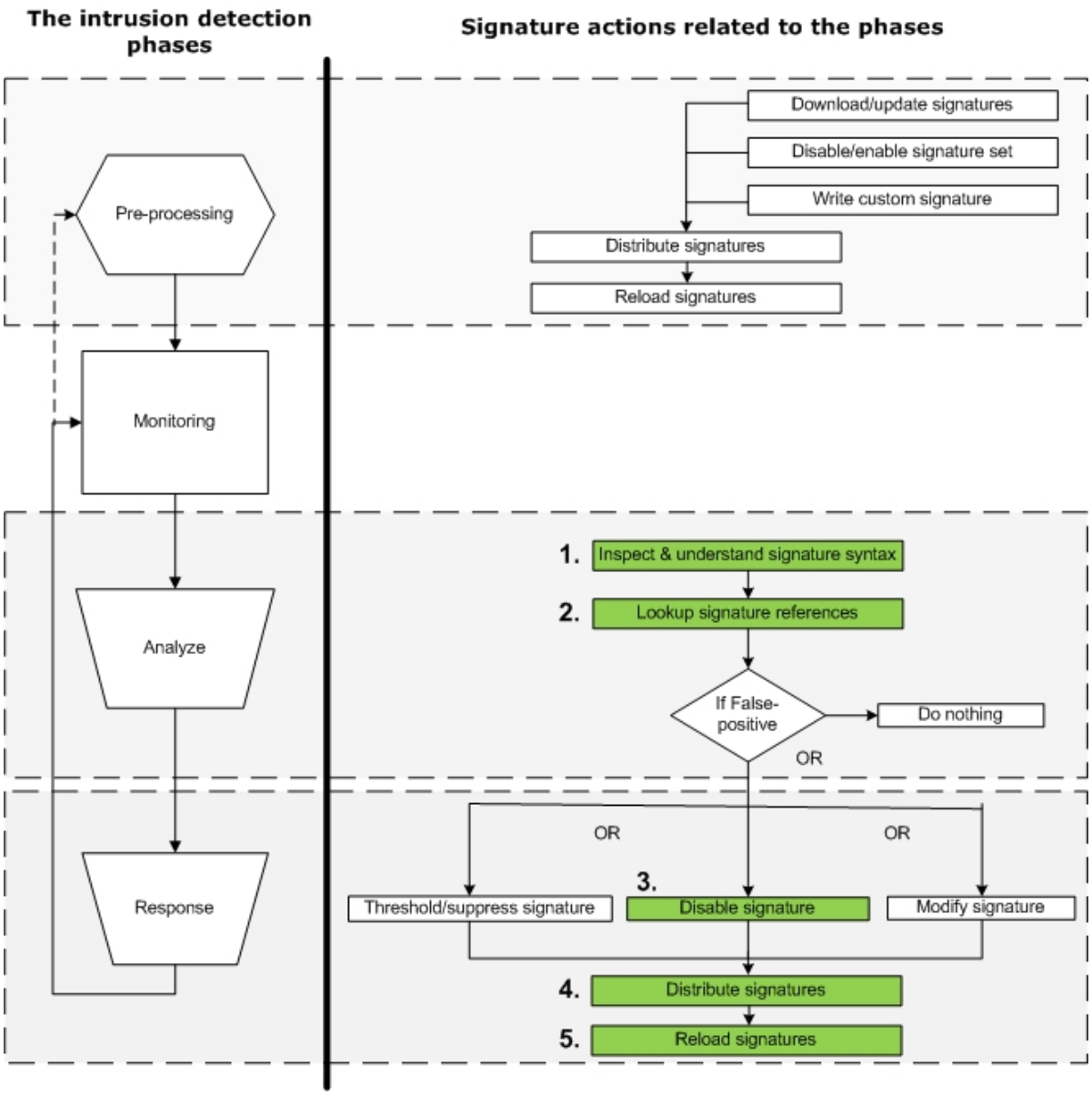


Figure 12: Task scenarios chosen for the KLM experiment

that the cognitive load is presumably lower when using the prototype was the reason why M was not used in this experiment. In other words, not taking M into account in this experiment will not affect the results of the prototype in any positive way with regard to the other methods, rather on the contrary. For the chosen task scenarios there are no waiting times of any significance and that is the reason why W(t) is not being used in this experiment.

The equation used in this experiment therefore looks like this:

$$T = K(n) + P(n) + B(n) + BB(n) + H(n)$$

where:

T is the total execution time
$n$ is the number of occurrences
K, P, B, BB and H is the operators execution time corresponding to Table 1

The difficult part when executing this experiment was to decide on what path and file names to use in some of the keystroke-level actions for the methods involving the use of a shell. To not affect the results in the prototype's favour the paths were not used, assuming that the user always knows where the current directory is, and the file names where kept to a minimum. As a consequence of this, the actual results from the experiment favour the shell based methods. In other words, the results obtained with regard to the shell based methods are better in terms of time than what they would generally be in a production environment. For each scenario, the scenario and the choices made that affect the result are described.

### 5.1.1 Scenario 1: Inspect signature syntax

In this scenario the user has received an IDS alert that was triggered by a signature with ID: 12193. The user wants to inspect the signature syntax to understand why the alert triggered.

Scenario assumptions:

- The keystroke-level action number 1 in Table 4 assumes that the shell is already standing in the signature directory.

Scenario keystroke-level actions and calculations:

| Descriptive action | Keystroke-level action |
|---|---|
| Search with SID in GUI: | 1. Point with mouse to a target on the display<br>2. Click mouse button<br>3. Hand to keyboard<br>4. Type: 12193<enter> |
| Expand signature to see syntax: | 5. Hand to mouse<br>6. Point with mouse to a target on the display<br>7. Click mouse button |

Table 3: Scenario 1 - Prototype

Table 3 calculations:

$$0.2sec * 6 + 1.1sec * 2 + 0.1sec * 0 + 0.2sec * 2 + 0.4sec * 2 = 4.6sec$$

| Descriptive action | Keystroke-level action |
|---|---|
| Search with SID in shell: | 1. Type: grep<space>–F<space>‘12193’<space>*<enter> |

Table 4: Scenario 1 - Manual method

Table 4 calculations:

$$0.2sec * 18 + 1.1sec * 0 + 0.1sec * 0 + 0.2sec * 0 + 0.4sec * 0 = 3.6sec$$

### 5.1.2 Scenario 2: Lookup signature reference

This scenario continues from the previous scenario in the way that the signature syntax is already displayed to the user. In this scenario the user wants to lookup one of the signature references that he found within the signature syntax.

Scenario assumptions:

- The keystroke-level actions 6-10 in Table 6 assume that the user already has an Internet browser window open in a shared view with the shell.
- The keystroke-level actions for the manual method assume that the reference to be looked-up is a complete URL. The Websites being used as reference the most often have a compressed URL, rendering copy-paste directly into the browser useless.

Scenario keystroke-level actions and calculations:

| **Descriptive action** | **Keystroke-level action** |
|---|---|
| Click on hyperlink in GUI: | 1. Point with mouse to a target on the display<br>2. Click mouse button |

Table 5: Scenario 2 - Prototype

Table 5 calculations:

$$0.2sec * 0 + 1.1sec * 1 + 0.1sec * 0 + 0.2sec * 1 + 0.4sec * 0 = 1.3sec$$

| **Descriptive action** | **Keystroke-level action** |
|---|---|
| Copy link from shell window: | 1. Point with mouse to a target on the display<br>2. Press mouse button<br>3. Point with mouse to a target on the display<br>4. Release mouse button<br>5. Type: <ctrl>c |
| Paste link into browser window: | 6. Point with mouse to a target on the display<br>7. Click mouse button<br>8. Type: <ctrl>v<br>9. Hand to keyboard<br>10. Type: <enter> |

Table 6: Scenario 2 - Manual method

Table 6 calculations:

$$0.2sec * 5 + 1.1sec * 3 + 0.1sec * 2 + 0.2sec * 1 + 0.4sec * 1 = 5.1sec$$

### 5.1.3 Scenario 3: Disable signature

In this scenario the user wants to disable the signature with ID: 12193. The manual method for doing this, described in Table 8, is a bad way to do this as the changes will be overwritten when the signature file is updated. Nonetheless it is the only way to do it. Suppress is often used instead, but the big difference between suppress and disabling is that when a signature is suppressed it will still be used by the IDS, thus still using resources, but when disabled it will not be used. As Pulled Pork has disabling functionality and preserves changes, it was also tested in this experiment. When testing Pulled Pork for this scenario the default file names were used.

Scenario assumptions:

- Scenario 3 for the manual method assume that the user already knows which rule-set (name) the SID is found within.
- The keystroke-level action number 1 in Table 8 assume that the shell is standing in the signature directory.
- The keystroke-level action number 1 in Table 8 assume that the signature-set file name containing the SID only consists of four letters (All signature-set files end with ".rules").

- The keystroke-level actions 1 and 7 in Table 9 assume that the disablesid.conf and the pulledpork.conf have been moved to this application's root directory (they are by default found in "app-root-dir/etc").
- The keystroke-level action 1 in Table 9 assume that the shell is standing in this application's root directory.

Scenario keystroke-level actions and calculations:

| **Descriptive action** | **Keystroke-level action** |
|---|---|
| Check box next to signature in GUI: | 1. Point with mouse to a target on the display<br>2. Click mouse button |
| Select sensor: | 3. Point with mouse to a target on the display<br>4. Click mouse button<br>5. Point with mouse to a target on the display<br>6. Click mouse button |
| Click execute button: | 7. Point with mouse to a target on the display<br>8. Click mouse button |

Table 7: Scenario 3 - Prototype

Table 7 calculations:

$$0.2sec * 0 + 1.1sec * 4 + 0.1sec * 0 + 0.2sec * 4 + 0.4sec * 0 = 5.2sec$$

| **Descriptive action** | **Keystroke-level action** |
|---|---|
| Open signature file for editing: | 1. Type: vim<space>name.rules<enter> |
| Search in file: | 2. Type: <shift>/12193<enter> |
| Activate insert mode: | 3. Type: i |
| Go to start of line: | 4. Type: <home> |
| Disable signature in file: | 5. Type: # |
| Exit and save: | 6. Type: <esc>:wq<enter> |

Table 8: Scenario 3 - Manual method

Table 8 calculations:

$$0.2sec * 31 + 1.1sec * 0 + 0.1sec * 0 + 0.2sec * 0 + 0.4sec * 0 = 6.2sec$$

| **Descriptive action** | **Keystroke-level action** |
|---|---|
| Open file for editing: | 1. Type: vim<space>disablesid.conf<enter> |
| Go to end of file: | 2. Type: <shift>g |
| Activate insert mode: | 3. Type: i |
| Go to start of new line: | 4. Type: <end><enter> |
| Disable signature: | 5. Type: 1:12193 |
| Exit and save: | 6. Type: <esc>:wq<enter> |
| Run changes: | 7. Type: pulledpork.pl<space>-c<space>pulledpork.conf <space>–<shift+t><enter> |

Table 9: Scenario 3 - Pulled Pork

Table 9 calculations:

$$0.2sec * 74 + 1.1sec * 0 + 0.1sec * 0 + 0.2sec * 0 + 0.4sec * 0 = 14.8sec$$

### 5.1.4 Scenario 4: Distribute signature change to a sensor

In this scenario the user wants to distribute all signatures from the current server to a sensor. The user only needs to update the signatures on the sensor that have been changed or are new. As the prototypes scales with regard to distribution, the prototype will distribute the signatures to all registered sensors when applying the keystroke-level actions in Table 10.

Scenario assumptions:

- The keystroke-level action 1 in Table 11 assume that the shell is standing in the signature directory.
- The keystroke-level action 1 in Table 11 assume that the signatures are found in the "/snort" directory on the sensor.
- The keystroke-level action 1 in Table 11 assume that the sensor name is five characters long.

Scenario keystroke-level actions and calculations:

| Descriptive action | Keystroke-level action |
|---|---|
| Click distribute button in GUI: | 1. Point with mouse to a target on the display<br>2. Click mouse button |

Table 10: Scenario 4 - Prototype

Table 10 calculations:

$$0.2sec * 0 + 1.1sec * 1 + 0.1sec * 0 + 0.2sec * 1 + 0.4sec * 0 = 1.3sec$$

| Descriptive action | Keystroke-level action |
|---|---|
| Copy files over the network: | 1. Type: rsync<space>-avz<space>*<space>sname:/snort/<enter> |

Table 11: Scenario 4 - Manual method

Table 11 calculations:

$$0.2sec * 27 + 1.1sec * 0 + 0.1sec * 0 + 0.2sec * 0 + 0.4sec * 0 = 5.4sec$$

### 5.1.5 Scenario 5: Reload signature change on a sensor

In this scenario the user wants to "restart" the IDS so that it starts using new or updated signatures. The prototype is not tested as a part of this scenario, because the prototype already performs this action when applying the keystroke-level action described in Table 10.

Scenario assumptions:

- The keystroke-level action number 1 in Table 12 assume that the sensor name is 5 characters long.

Scenario keystroke-level actions and calculations:

| Descriptive action | Keystroke-level action |
|---|---|
| Log-into sensor:<br>Load changes: | 1. Type: ssh<space>sname<enter><br>2. Type: killall<space>-<shift+HUP><space>snort<enter> |

Table 12: Scenario 5 - Manual method

Table 12 calculations:

$$0.2sec * 26 + 1.1sec * 0 + 0.1sec * 0 + 0.2sec * 0 + 0.4sec * 0 = 5.2sec$$

### 5.1.6 Results

The results from this experiment show that by using the prototype in 4 of the 5 scenarios chosen from the signature management process, the user would achieve increased efficiency compared to using the manual method. In Table 13 the results obtained from all the experiments can be

found. This Table also contains a calculation of the total time used for completing scenario 1 trough 5 for each method. From these calculations one can see that the prototype's total time is less than half of what the manual method's total time is. Using Pulled Pork in scenario 3 instead of the manual method, which is very common, will increase the total time and emphasize the difference in efficiency even more.

| | **Scenario 1** | **Scenario 2** | **Scenario 3** | **Scenario 4** | **Scenario 5** | **Time total** |
|---|---|---|---|---|---|---|
| Prototype: | 4.6 sec | 1.3 sec | 5.2 sec | 1.3 sec | - | 12.4 sec |
| Manual method: | 3.6 sec | 5.1 sec | 6.2 sec | 5.4 sec | 5.2 sec | 25.5 sec |
| Pulled Pork: | - | - | 14.8 sec | - | - | - |

Table 13: Results obtained from all the scenarios

In the scenarios, recurrences were not considered. For example, for scenario 2 it is very common that a signature contains more than one reference. Figure 13 shows the time used as a function of the number of references to lookup. From this Figure, one can see that the time ratio is constant, but the big difference in time between the manual method and using the prototype were so great from the start, that increasing the number of references would drastically increase the total time used for the manual method. In scenario 4 and 5 in this experiment there was only one sensor involved. Figure 14 shows time used in these scenarios as a function of the number of sensors. As the prototype was developed with sensor scalability in mind, the time used for these scenarios will not increase when increasing the number of sensors. From Figure 14, one can see that the pattern is horizontal. For the manual method this is not the case. From the Figure, one can see that the pattern increases uniformly, which means that the time when using the manual method is multiplied with the number of sensors.

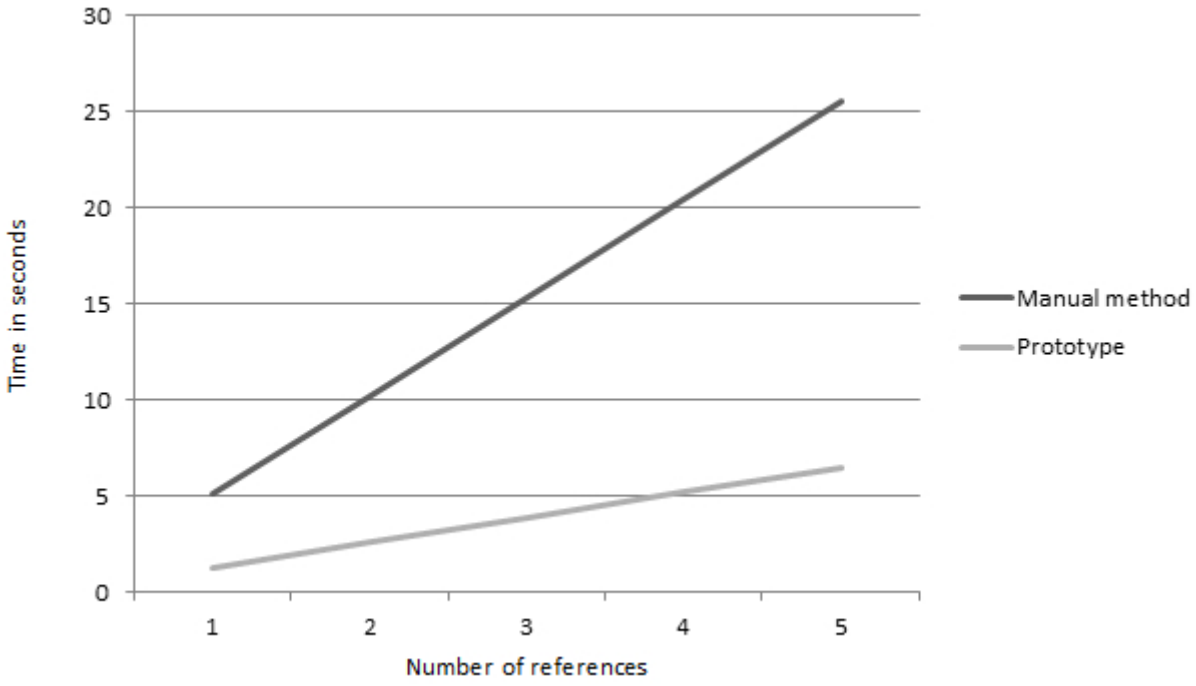


Figure 13: Time used for scenario 2 as a function of the number of references to lookup

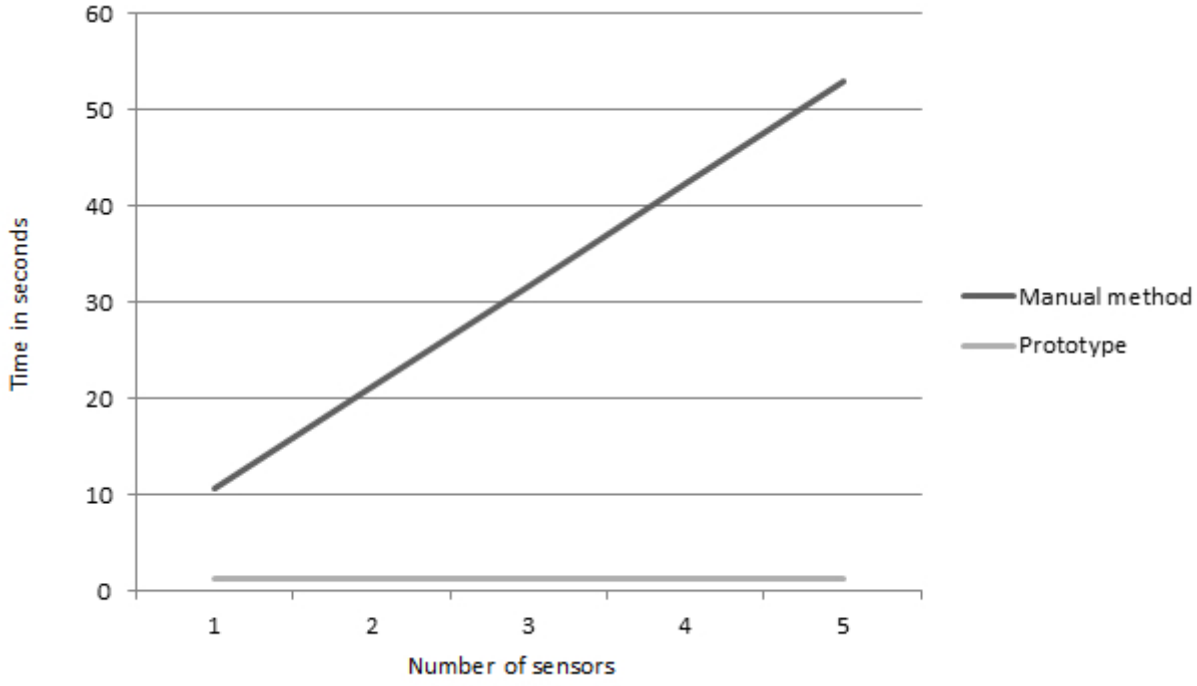


Figure 14: Time used for scenario 4 and 5 as a function of the number of sensors

## 5.2 Questionnaire

As mentioned in the methods chapter a USE-questionnaire has been used to supplement the results from the KLM experiment, but also to get feedback on the other aspects of usability.

### 5.2.1 Test subjects

The test subject in this experiment were information security practitioners employed at the Norwegian Armed Forces Critical Infrastructure Protection Centre (CIPC) working actively with intrusion detection. The majority of theses operators have a minimum of two years experience working with IDS, and they are in the main target group of users that this system is intended for.

### 5.2.2 Testing

The prototype was installed at a test server at the CIPC and the IDS-operators present were given a short introduction to its features and limitations. Further the operators were given the opportunity to ask questions. Beforehand the Web-based questionnaire had been created and the operators were at the end of the sessions provided with the link to it before they were left alone with the prototype. From what we understand, the prototype was a week later installed on one of the CIPC servers belonging to their production/live environment. The answers to the questionnaire could therefore be from experiences with the prototype in a testing or a production environment.

The questionnaire that was used can be found in Appendix A. Questions 1 to 3 measure perceived usefulness/efficiency, questions 4 to 5 measure perceived ease of use, question 6 measure perceived ease of learning and the last two questions measure satisfaction. In addition, there was a non-mandatory question giving the test subjects the opportunity to give written feedback on any improvements or new features that they feel would make the system more useful. The answers to the questionnaire were given on a scale from 1 to 7, where 1 appeared to the test subject as strongly disagree and 7 as strongly agree.

### 5.2.3 Results

After approximately three weeks the questionnaire was closed and the results gathered. The results showed that 6 operators had taken the time to test the prototype and answer the questionnaire. This is a very small sample size, but considering the total number of people at CIPC currently working with IDS, the sample size is most likely representative for that group (see Section 5.2.4 for discussion about validity).
Table 14 shows the results, by the use of mean and median calculations, obtained from the questionnaire. Overall the prototype is rated very high, which indicates that the prototype is usable, something the fact that CIPC introduced the system into their production environment also does. The highest group results are found in the usefulness/efficiency group, which does not come as a surprise as this prototype was primarily developed to improve this aspect. These results support the results obtained in the KLM experiment. The lowest result, obtained from question 7, can be explained by interpretation of the results from the non-mandatory feedback question. 3 operators chose to answer this question. Half of the feedback contained within the replies were directed at improving the current design, while the other half were directed at adding new features to the system. From the suggestions on improvements to current design there were none that would

| Question | Mean | Median |
|---|---|---|
| 1. This system would help me be more effective | 6.3 | 6 |
| 2. This system would be useful in my work with Snort | 6 | 6 |
| 3. This system would save me time if I used it | 6.5 | 6.5 |
| 4. This system is easy to use | 6 | 6 |
| 5. This system requires the fewest steps possible to accomplish what I want to do with it | 5.5 | 5 |
| 6. This system is easy to learn to use | 5.5 | 6 |
| 7. This system works the way I want it to work | 5 | 5 |
| 8. I prefer this system over alternative tools available for Snort today | 6.3 | 6 |

Table 14: Results obtained from the questionnaire

improve directly upon the efficiency of the system. Suggestions regarding new features do not come as a surprise, as in the design phase of this prototype some features were left out on purpose.

Some comments on improvements to current design:

- The system should show more clearly that it is working.
- The views should show 100 entries by default.
- When disabling a rule-set, the already done tuning on single rules disappears. I'd like the tuning to stay, in case I decide to temporarily disable a rule-set.

Some comments on new features:

- Add/remove rule-set sources from the web GUI.
- The system should support commenting.
- When tuning, select why I want to tune it. What means, if I want to tune it because I do not want to see this alarm/alert/event triggering (hence disabling the sig and all future revisions) or if I want to tune it because it is simply not good enough (meaning future revisions should be enabled by default).

### 5.2.4 Reliability and validity

The sample size for this questionnaire is too small for this experiment to be scientifically valid. Although the current sample size is too small for the results to be representative, the size needed for the results to be, can be discussed. People having a job where they work daily with IDS are not part of a large group compared to the people not belonging in this group. The people belonging to this special group also share many other properties like similar education. Bearing that in mind, supplementing the results obtained in this thesis with results obtained from 6 to 9 other test subjects, working for a different company, should be sufficient to produce a valid result. The results obtained in this thesis could therefore be used as an indication to what the valid results

would look like.

As mentioned earlier, this questionnaire was not intended to be used as measurement of improved efficiency compared to existing solutions, but rather as a supplement to the already obtained results. In addition, the questionnaire was intended to bring feedback on the rest of the usability aspect, which it has done.

Looking at the answers given, the overall score is very high. As the test subjects where not observed through the testing and response phase of this experiment, it is possible that one of the test subjects could have expressed his thoughts loudly and in that way affected another test subject, which in turn would affect the results. This is not an unlikely scenario, but since there are no answers to any of the questions that significantly differentiate themselves from the others, it should be safe to say that this has not been the case as it is unlikely that one person would actively affect five other persons in that way.

This questionnaire would most likely produce the same results if given to the same test subject group, but given to another test subject group the results would not necessarily be exact, but they should not be too different considering the homogeneity of the test subjects.

# 6 Conclusion

This thesis has defined the intrusion detection system (IDS) signature management process and proposed a method for creating an efficient system that supports the human work with regard to the identified actions belonging to that process. Further this thesis has implemented a usable system based on that method. The research question will now be addressed.

- To what extent would a dedicated graphical tool improve human efficiency in the IDS signature management process, in terms of time, compared to the existing manual method?

Considering the overall test results, it can be concluded that a graphical tool, such as the one created in this thesis, has the potential to reduce the time actually spent performing actions in the IDS signature management process by half compared to time spent when using the manual method. Or in other words, the graphical tool has potential to increase the work done actually performing actions in this process by twice as much compared to the work done when using the manual method. Considering an environment with more sensors, the potentially gained reduction in time spent would be reduced even more as discussed in Section 5.1.6 and showed in Figure 14. The results from the questionnaire, which can be found in Table 14 in Section 5.2, shows that the perceived efficiency of the system is concurrent with the test results. The results from the questionnaire also indicated that the overall usability of such a system, designed according to the method described in this thesis, is very high. This in turn indicates that such a system will be preferred instead of the manual method if such a system is made available.

The results from this thesis show that a system as the one developed in this thesis can be a great contribution to solving the false-positive problem, as this system would help IDS-operators be more efficient in the work of filtering signatures, which in turn will reduce the false-positive rate and thereby make the IDS more useful. In addition, by using less time in the IDS signature management process, IDS-operators would have more time to do other tasks.

## 6.1 Future work

This thesis has presented a system that if taken into use would increase human efficiency in IDS work. The system developed is usable, but it is a prototype and is therefore not considered stable enough by the author to be used in a production environment. By using the work done in this thesis as guidance for creating a new tool or by improving on the work done in this thesis, a stable tool with proved efficiency improvement could be obtained. Further, the configuration of IDS network variables and settings, which today is also a manual process, could be considered to be implemented as a part of this tool as well. From the results obtained through the questionnaire, the other aspects of usability of this system showed promise, but as the sampling size

was too small, the other aspects of this system's usability were not satisfactorily demonstrated. As previously mentioned, the whole aspect of usability is taken into account when a person/-company decides whether to take a tool into use or not. The usability aspect should therefore be tested more accurately before being able to say that such system could replace the manual method.

# Bibliography

[1] Werlinger, R., Hawkey, K., Muldner, K., Jaferian, P., & Beznosov, K. 2008. The challenges of using an intrusion detection system: is it worth the effort? In *Proceedings of the 4th symposium on Usable privacy and security*, SOUPS '08, 107–118, New York, NY, USA. ACM.

[2] Zomlot, L., Sundaramurthy, S. C., Luo, K., Ou, X., & Rajagopalan, S. R. 2011. Prioritizing intrusion analysis using dempster-shafer theory. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, AISec '11, 59–70, New York, NY, USA. ACM.

[3] Hooper, E. june 2006. An intelligent detection and response strategy to false positives and network attacks: operation of network quarantine channels and feedback methods to ids. In *Security, Privacy and Trust in Pervasive and Ubiquitous Computing, 2006. SecPerU 2006. Second International Workshop on*, 6 pp. –21.

[4] Shimamura, M. & Kono, K. june 2006. Using attack information to reduce false positives in network ids. In *Computers and Communications, 2006. ISCC '06. Proceedings. 11th IEEE Symposium on*, 386 – 393.

[5] Snort, a open source network intrusion prevention and detection system. `http://www.snort.org/`. [Online; accessed 10-April-2012].

[6] Koike, H. & Ohno, K. 2004. Snortview: visualization system of snort logs. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, VizSEC/DMSEC '04, 143–147, New York, NY, USA. ACM.

[7] Stakhanova, N. & Ghorbani, A. A. 2010. Managing intrusion detection rule sets. In *Proceedings of the Third European Workshop on System Security*, EUROSEC '10, 29–35, New York, NY, USA. ACM.

[8] Symantec. April 2012. Internet security threat report: The 2011 threat landscape. 17.

[9] The open information security foundation - suricata. `http://www.openinfosecfoundation.org/`. [Online; accessed 10-April-2012].

[10] The bro network security monitor. `http://bro-ids.org/`. [Online; accessed 10-April-2012].

[11] J. Goodall, W. L. & Komlodi, A. 2004. The work of intrusion detection: Rethinking the role of security analysts. *In Proc of the Americas Conference on Information Systems (AMCIS)*, 1421–1427.

[12] R. S. Thompson, E. R. & Yurcik, W. 2006. Network intrusion detection cognitive task analysis: Textual and visual tool usage and recommendations. *In Proceedings of the Human Factors and Ergonomics Society Annual Meeting (HFES)*, 669–673.

[13] Forum thread: Managing snort signature tuning. `https://www.alienvault.com/forum/index.php?t=msg&goto=7951&S=219d2b75fab903353d269a7f2803d83`. [Online; accessed 17-May-2012].

[14] Forum thread: Centrally manage snort. `https://www.alienvault.com/forum/index.php?t=msg&goto=2657&S=dbcdcf2e6b355cde653308c34825233`. [Online; accessed 17-May-2012].

[15] Forum thread: Manage snort rules. `https://www.alienvault.com/forum/index.php?t=msg&th=3154&goto=10105&S=189f264ef18b1608005e488aee1352`. [Online; accessed 17-May-2012].

[16] Thompson, R. S., Rantanen, E. M., Yurcik, W., & Bailey, B. P. 2007. Command line or pretty lines?: comparing textual and visual interfaces for intrusion detection. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, 1205–, New York, NY, USA. ACM.

[17] Oinkmaster. `http://oinkmaster.sourceforge.net/`. [Online; accessed 17-April-2012].

[18] Pulled pork. `http://code.google.com/p/pulledpork/`. [Online; accessed 17-April-2012].

[19] Spenneberg, R. April 2011. Snort helpers, article published in linux pro magazine, available online. `http://www.linuxpromagazine.com/content/download/61729/482620/file/032-038_snort.pdf`. [Online; accessed 06-May-2012].

[20] Polman. `http://www.gamelinux.org/?p=240`. [Online; accessed 17-April-2012].

[21] Snorby. `http://snorby.org/`. [Online; accessed 17-April-2012].

[22] Kieras, D. 2001. Using the keystroke-level model to estimate execution times. the university of michigan, unpublished report. `http://www.pitt.edu/~cmlewis/KSM.pdf`. [Online; accessed 12-June-2012].

[23] Lund, A. M. October 2001. Measuring usability with the use questionnaire, usability interface society for technical communication usability sig publication. 8(2).

[24] Datatables, a plug-in for the jquery javascript library. `http://datatables.net`. [Online; accessed 28-April-2012].

[25] jquery, a fast and concise javascript library. `http://jquery.com`. [Online; accessed 28-April-2012].

[26] Tornado, a open source web server. `http://www.tornadoweb.org`. [Online; accessed 28-April-2012].

[27] Taylor, B. September 2009. The technology behind tornado, friendfeed's web server. http://backchannel.org/blog/tornado. [Online; accessed 28-April-2012].

[28] Sqlite, the most widley deployed sql database engine in the world. http://www.sqlite.org/. [Online; accessed 28-April-2012].

[29] Rsync. http://www.samba.org/ftp/rsync/rsync.html. [Online; accessed 29-April-2012].

[30] Openssh. http://www.openssh.com/. [Online; accessed 29-April-2012].

[31] Github, repository: bringhomethebacon. https://github.com/davhenriksen/bringhomethebacon. [Online; accessed 7-May-2012].

# Appendices

# A Web-based questionnaire

**This system would help me be more effective**

- Strongly disagree
- Disagree
- Slightly disagree
- Neither agree nor disagree
- Slightly agree
- Agree
- Strongly agree

**This system would be useful in my work with Snort**

- Strongly disagree
- Disagree
- Slightly disagree
- Neither agree nor disagree
- Slightly agree
- Agree
- Strongly agree

**This system would save me time if I used it**

- Strongly disagree
- Disagree
- Slightly disagree
- Neither agree nor disagree
- Slightly agree
- Agree
- Strongly agree

**This system is easy to use**

- Strongly disagree
- Disagree
- Slightly disagree
- Neither agree nor disagree
- Slightly agree
- Agree
- Strongly agree

**This system requires the fewest steps possible to accomplish what I want to do with it**

- Strongly disagree
- Disagree
- Slightly disagree
- Neither agree nor disagree
- Slightly agree
- Agree
- Strongly agree

**This system is easy to learn to use**

- Strongly disagree
- Disagree
- Slightly disagree
- Neither agree nor disagree
- Slightly agree
- Agree
- Strongly agree

**This system works the way I want it to work**

- Strongly disagree
- Disagree
- Slightly disagree
- Neither agree nor disagree
- Slightly agree
- Agree
- Strongly agree

**I prefer this system over alternative tools available for Snort today**

- Strongly disagree
- Disagree
- Slightly disagree
- Neither agree nor disagree
- Slightly agree
- Agree
- Strongly agree

**Any changes or improvements that could have made this system more useful?**

# B update.py

The script that handles updating of signatures and the insertion of them into the database.

```
#!/usr/bin/env python

# update.py

#    bring home the bacon Copyright (C) 2012 David Ormbakken
    Henriksen (davidohenriksen@gmail.com)
#
#    This program is free software: you can redistribute it and/or
    modify
#    it under the terms of the GNU General Public License as
    published by
#    the Free Software Foundation, either version 3 of the License,
    or
#    (at your option) any later version.
#
#    This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.


import os
import sys
import urllib2
import tarfile
import sqlite3
import datetime
import re
import glob

from collections import defaultdict

from config import *

def merge_or_move_file(path,file_name):
        distr_file_path = C_distribute_dir+file_name
        tmp_file_path = path+file_name

        if not os.path.exists(tmp_file_path):
```

```
                txt = 'Error: '+file_name+' could not be found at
                    path: '+tmp_file_path
                print txt
                write_logfile(txt)

        else:
                tmp_FILE = open(tmp_file_path, "r")

                if not (os.path.exists(distr_file_path)):
                        distr_FILE = open(distr_file_path, "w")
                        for line in tmp_FILE:
                                if line.strip():
                                        if not line.startswith('#'):
                                                distr_FILE.write(line
                                                    )

                        tmp_FILE.close()
                        distr_FILE.close()

                else:
                        distr_FILE = open(distr_file_path, "r")
                        tmp = distr_FILE.readlines()
                        distr_FILE.close()
                        distr_FILE = open(distr_file_path, "a")
                        for line in tmp_FILE:
                                if line.strip():
                                        if not line.startswith('#'):
                                                if line not in tmp:
                                                        distr_FILE.
                                                            write(line
                                                            )

                        tmp_FILE.close()
                        distr_FILE.close()

def write_logfile(txt):
        FILE = open(C_updatelog,'a')
        FILE.write(str(txt)+'\n')
        FILE.close()

def check_md5(md5,file_name):
        if os.path.exists(file_name):
                txt = "Comparing md5 checksums..."
                print txt
                write_logfile(txt)
                FILE = open(file_name,"r")
                if not (cmp(FILE.readline(),md5) == 0):
                        FILE.close()
                        FILE = open(file_name,"w")
```

```
                        FILE.write(md5)
                        FILE.close()
                        return True
                else:
                        FILE.close()
                        return False
        else:
                FILE = open(file_name,"w")
                FILE.write(md5)
                FILE.close()
                return True

def download_md5(url):
        try:
                txt = "Dowloading md5: "+url
                print txt
                write_logfile(txt)
                md5 = urllib2.urlopen(url).read().replace('"','')
                return md5
        except urllib2.HTTPError, e:
                txt = 'HTTP Error: '+str(e)
                print txt
                write_logfile(txt)
        except urllib2.URLError, e:
                txt = 'URL Error: '+str(e)
                print txt
                write_logfile(txt)

def download_rules(url,filename):
        try:
                txt = "Downloading rules: "+url
                print txt
                write_logfile(txt)
                tmp = urllib2.urlopen(url)
                FILE = open((C_tmp_dir+filename),"w")
                FILE.write(tmp.read())
                FILE.close()
                return True
        except urllib2.HTTPError, e:
                txt = 'HTTP Error: '+str(e)
                print txt
                write_logfile(txt)
                return False
        except urllib2.URLError, e:
                txt = 'URL Error: '+str(e)
                print txt
                write_logfile(txt)
                return False
```

```
def extract_file(filename,name):
        try:
                txt = 'Extracting␣files...'
                print txt
                write_logfile(txt)
                FILE = tarfile.open((C_tmp_dir+filename),'r:gz')
                FILE.extractall(C_tmp_dir+name)
                FILE.close()
                return True
        except StandardError, e:
                txt = "TAR␣Error:",e
                print txt
                write_logfile(txt)
                return False

def find(regex, string):
        res = re.search(regex, string)
        if not (res is None):
                res = ((res.group()).strip())
        else:
                res = 'none'
        return  res

def find_ref(string):
        ref_tmp = re.findall(r'(?<=reference:)(.*?)(?=;)', string)
        ref = ''

        if not os.path.exists(C_distribute_dir+'reference.config'):
                for hit in ref_tmp:
                        ref = ref+'␣␣'

        else:
                for hit in ref_tmp:
                        FILE = open((C_distribute_dir+'reference.
                            config'), 'r')
                        hit = hit.strip('http')
                        hit = hit.strip('https')
                        key = re.match(r'(.*?)(?=,)',hit).group()
                        url_lastpart = re.search(r'(?<=\,).*',hit).
                            group()
                        url_firstpart = ''
                        regex = r'(?<=%s).*' % key
                        for line in FILE:
                                tmp = re.search(regex,line,re.
                                    IGNORECASE)
                                if tmp is not None:
                                        url_firstpart = tmp.group()
```

```
                        ref = ref+('<a␣rel="nofollow"␣target="_blank"
                            ␣href="'+url_firstpart.strip()+
                            url_lastpart.strip()+'"␣target="_blank">'+
                            key.upper()+'</a>,␣')
                        FILE.close()
        return ref

def read_rule_files(path,source):
        txt = "Reading␣in␣rules..."
        print txt
        write_logfile(txt)
        rules = []
        filepaths = glob.glob(path+'*.rules')
        for filepath in filepaths:
                ruleset = os.path.basename(filepath).replace('.rules'
                    , '').replace('emerging-', '')
                FILE = open(filepath,"r")
                for line in FILE:
                        if line.strip():
                                if not line.startswith('#'):
                                        sid = find(r'(?<=[^l]sid:)
                                            (.*?)(?=;)',line)
                                        rev = find(r'(?<=rev:)(.*?)
                                            (?=;)',line)
                                        name = find(r'(?<=msg:")(.*?)
                                            (?=";)',line)
                                        ref = find_ref(line.lower())
                                        rules.append([sid,rev,source,
                                            ruleset,name,ref,date,line
                                            ])
                FILE.close()
        return rules

def insert_into_db(rules):
        txt = "Inserting␣rules␣into␣db..."
        print txt
        write_logfile(txt)

        db = sqlite3.connect(C_db_path)
        cursor = db.cursor()

        for line in rules:
                sid,rev,source,ruleset,name,ref,date,rule = line

                try:
                        cursor.execute("SELECT␣sidnr,␣revnr␣FROM␣
                            rules␣WHERE␣sidnr␣=␣(?)␣", [sid])
                        res = cursor.fetchone()
                except StandardError, e:
```

```
                        txt = "Exiting␣with␣error:␣"+str(e)
                        print txt
                        write_logfile(txt)
                        sys.exit()

                if res is None:  #if true = new sid
                        try:
                                cursor.execute('''INSERT INTO rules (
                                    sidnr,revnr,source_name,
                                    ruleset_name,rule_name,ref,date,
                                    rule_syntax)
                                VALUES(?,?,?,?,?,?,?,?)''',[sid,rev,
                                    source,ruleset,name,ref,rule_date,
                                    rule])
                        except StandardError, e:
                                txt = "Exiting␣with␣error:␣"+str(e)
                                print txt
                                write_logfile(txt)
                                sys.exit()

                elif (res[1] is not None):
                        if not res[1] < rev:  #if true = old sid, but
                            new/higher rev
                                try:
                                        cursor.execute('''UPDATE
                                            rules SET revnr = (?),
                                            source_name = (?),
                                            ruleset_name = (?),
                                        rule_name = (?),  ref = (?),
                                            date = (?), rule_syntax
                                            =(?) WHERE sidnr = (?)''',
                                        [rev,source,ruleset,name,ref,
                                            rule_date,rule,sid])
                                except StandardError, e:
                                        txt = "Exiting␣with␣error:␣"+
                                            str(e)
                                        print txt
                                        write_logfile(txt)
                                        sys.exit()

        db.commit()
        cursor.close()
        db.close()
        txt = 'Finished.␣success!'
        print txt
        write_logfile(txt)


#### FUNCTIONS END ####
```

```
#### GLOBAL VARIABLES START ####
now = datetime.datetime.now()
date = '%d:%d %d/%d/%d' % (now.hour,now.minute,now.day,now.month,now.
    year)
rule_date = now.strftime("%Y-%m-%d")

#### GLOBAL VARIABLES END ####


#### MAIN START ####
txt = "Rule update started: "+date
print txt
write_logfile(txt)

#operations on local rule source
if not (C_locale_rule_path == ''):
        txt = 'Updating rules for source: local'
        print txt
        write_logfile(txt)
        insert_into_db(read_rule_files(C_locale_rule_path,'local'))

#operations on external rule sources
for source in C_rule_sources:
        source_name, md5_url, rule_url, rules_path, files_path =
            source
        source_name = source_name.lower().replace(' ', '')
        txt = 'Updating rules for source: '+source_name
        print txt
        write_logfile(txt)

        if not (md5_url == ''):
                if check_md5((download_md5(md5_url)),(C_tmp_dir+
                    source_name+'.md5')):
                        if not (download_rules(rule_url,(source_name+
                            '.tar.gz')) is False):
                                if not (extract_file((source_name+'.
                                    tar.gz'),source_name) is False):
                                        if not (files_path == 'none')
                                            :
                                                txt = 'Starting
                                                    operations on .
                                                    conf and .map
                                                    files...'
                                                print txt
                                                write_logfile(txt)
                                                for f in C_files:
```

```
                                                    merge_or_move_file
                                                        ((
                                                        C_tmp_dir+
                                                        source_name
                                                        +'/'+
                                                        files_path
                                                        ),f[0])
                                                txt = 'Done. Files
                                                    have been moved or
                                                    merged'
                                                print txt
                                                write_logfile(txt)

                                            insert_into_db(
                                                read_rule_files((C_tmp_dir
                                                +source_name+'/'+
                                                rules_path),source_name))
                    else:
                            txt = "No new rules to download"
                            print txt
                            write_logfile(txt)
            else:
                    txt = 'Skipping md5 check'
                    print txt
                    write_logfile(txt)

                    if not download_rules(rule_url,(source_name+'.tar.gz'
                        )) is False:
                            if not (extract_file((source_name+'.tar.gz'),
                                source_name) is False):
                                    if not (files_path == 'none'):
                                            txt = 'Starting operations on
                                                . conf and .map files ...'
                                            print txt
                                            write_logfile(txt)
                                            for f in C_files:
                                                    merge_or_move_file((
                                                        C_tmp_dir+
                                                        source_name+'/'+
                                                        files_path),f[0])
                                            txt = 'Done. Files have been
                                                moved or merged'
                                            print txt
                                            write_logfile(txt)

                                    insert_into_db(read_rule_files((
                                        C_tmp_dir+source_name+'/'+
                                        rules_path),source_name))
                    else:
```

```
                txt = "No new rules to download"
                print txt
                write_logfile(txt)

#### MAIN END ####
```

# C distribute.py

The script that extracts signatures from the database and then distributes them to the sensors.

```
#!/usr/bin/env python

# distribute.py

#    bring home the bacon Copyright (C) 2012 David Ormbakken
    Henriksen (davidohenriksen@gmail.com)
#
#    This program is free software: you can redistribute it and/or
    modify
#    it under the terms of the GNU General Public License as
    published by
#    the Free Software Foundation, either version 3 of the License,
    or
#    (at your option) any later version.
#
#    This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.

import os
import sys
import urllib2
import sqlite3
import datetime
import subprocess

from config import *

#### FUNCTIONS START ####
def write_logfile(txt):
        FILE = open(C_distrblog,'a')
        FILE.write(str(txt)+'\n')
        FILE.close()

def get_sensors():
        db = sqlite3.connect(C_db_path)
        cursor = db.cursor()

        try:
```

```
                txt = "Collecting␣sensor␣information..."
                print txt
                write_logfile(txt)
                cursor.execute('SELECT␣*␣FROM␣sensors')
                all_sensors = cursor.fetchall()
                return all_sensors

        except StandardError, e:
                txt = 'Exiting␣with␣error:␣'+str(e)
                print txt
                write_logfile(txt)
                sys.exit()

        cursor.close()
        db.close()

def get_rules(sname):
        db = sqlite3.connect(C_db_path)
        cursor = db.cursor()

        try:
                sql = 'SELECT␣sid␣FROM␣'+sname+'_disabled'
                cursor.execute(sql)
                tmp = cursor.fetchall()
                txt = 'Disabled␣rule␣count:␣'+str(len(tmp))
                print txt
                write_logfile(txt)
                disabled = ",".join(str(x[0]) for x in tmp)
                sql = 'SELECT␣rule_syntax␣FROM␣rules␣WHERE␣sidnr␣NOT␣
                   IN␣('+disabled+')'
                cursor.execute(sql)
                rules = cursor.fetchall()
                txt = 'Enabled␣rule␣count:␣'+str(len(rules))
                print txt
                write_logfile(txt)
                return rules

        except StandardError, e:
                txt = 'Exiting␣with␣error:␣'+str(e)
                print txt
                write_logfile(txt)
                sys.exit()

        cursor.close()
        db.close()

def create_dir(name):
        try:
                if not os.path.isdir(name):
```

```
                        txt = 'Creating␣directory:␣'+name
                        print txt
                        write_logfile(txt)
                        os.mkdir(name)

        except StandardError, e:
                txt = 'Exiting␣with␣error:␣'+str(e)
                print txt
                write_logfile(txt)
                sys.exit()

def write_rules(sname,rules):
        try:
                dir_name = C_distribute_dir+sname+'/'
                create_dir(dir_name)

                txt = 'Writing␣'+sname+'.rules...'
                print txt
                write_logfile(txt)

                FILE = open(dir_name+sname+'.rules', "w")

                for rule in rules:
                        FILE.write(rule[0])

                FILE.close()

        except StandardError, e:
                txt = 'Exiting␣with␣error:␣'+str(e)
                print txt
                write_logfile(txt)
                sys.exit()

def get_threshold(sname):
        db = sqlite3.connect(C_db_path)
        cursor = db.cursor()

        try:
                txt = 'Checking␣for␣threshold␣and␣suppress␣rules...'
                print txt
                write_logfile(txt)

                sql = 'SELECT␣syntax␣FROM␣'+sname+'_threshold'
                cursor.execute(sql)
                threshold = cursor.fetchall()

                txt = 'Threshold/suppress␣rule␣count:␣'+str(len(
                   threshold))
                print txt
```

```
                write_logfile(txt)

                if not str(len(threshold)) == '0':
                        return threshold
                else:
                        return False

        except StandardError, e:
                txt = 'Exiting with error: '+str(e)
                print txt
                write_logfile(txt)
                sys.exit()

        cursor.close()
        db.close()

def write_threshold(sname,threshold):
        try:
                dir_name = C_distribute_dir+sname+'/'

                if len(threshold) is not '0':
                        txt = 'Writing threshold.conf...'
                        print txt
                        write_logfile(txt)

                        FILE = open(dir_name+'threshold.conf', "w")

                        for rule in threshold:
                                FILE.write(rule[0]+'\n')

                        FILE.close()

        except StandardError, e:
                txt = 'Exiting with error: '+str(e)
                print txt
                write_logfile(txt)
                sys.exit()

def transfer_files(dir_name,dest):
        try:
                for file in os.listdir(dir_name):
                        if '.' in file:
                                txt = 'Syncing file: '+file
                                print txt
                                write_logfile(txt)

                                tmp = dir_name+file
```

```
                                p = subprocess.Popen(["rsync","-auvz"
                                    ,"-e","ssh",tmp,dest],stdout=
                                    subprocess.PIPE)
                                for line in p.stdout:
                                        if line.replace('\n','').
                                            strip():
                                                txt = line
                                                print txt
                                                write_logfile(txt)
                                p.stdout.close()

        except StandardError, e:
                txt = 'Error: '+str(e)
                print txt
                write_logfile(txt)

def distribute(sname,ip,path,uname):
        try:
                dest = uname+'@'+ip+':'+path

                txt = 'File transfer started...'
                print txt
                write_logfile(txt)

                transfer_files(C_distribute_dir,dest)
                transfer_files(C_distribute_dir+sname+'/',dest)

        except StandardError, e:
                txt = 'Error: '+str(e)
                print txt
                write_logfile(txt)

def reload_rules(ip,uname,cmd):
        try:
                txt = 'Reloading rules on sensor...'
                print txt
                write_logfile(txt)

                command = cmd+' && exit'

                p = subprocess.Popen(["ssh","-x","-l",uname,ip,
                    command],stdout=subprocess.PIPE)
                for line in p.stdout:
                        txt = line+'\n'
                        print txt
                        write_logfile(txt)
                p.stdout.close()

        except StandardError, e:
```

```
                txt = 'Error: '+str(e)
                print txt
                write_logfile(txt)

#### FUNCTIONS END ####


#### GLOBAL VARIABLES START ####
now = datetime.datetime.now()
date = '%d:%d %d/%d/%d' % (now.hour,now.minute,now.day,now.month,now.
    year)

#### GLOBAL VARIABLES END ####


#### MAIN START ####
txt = "Rule distribution started: "+date
print txt
write_logfile(txt)

all_sensors = get_sensors()
for sensor in all_sensors:
        sname,ip,path,uname,cmd = sensor
        txt = 'Gathering rules for sensor: '+sname
        print txt
        write_logfile(txt)
        write_rules(sname,get_rules(sname))
        threshold = get_threshold(sname)
        if threshold is not False:
                write_threshold(sname,threshold)
        distribute(sname,ip,path,uname)
        reload_rules(ip,uname,cmd)
        txt = 'Done.'
        print txt
        write_logfile(txt)


#### MAIN END ####
```

# D web.py

The web server script. Handles all communication between the server and the client.

```
#!/usr/bin/env python

# web.py

#     bring home the bacon Copyright (C) 2012 David Ormbakken
    Henriksen (davidohenriksen@gmail.com)
#
#     This program is free software: you can redistribute it and/or
    modify
#     it under the terms of the GNU General Public License as
    published by
#     the Free Software Foundation, either version 3 of the License,
    or
#     (at your option) any later version.
#
#     This program is distributed in the hope that it will be useful,
#     but WITHOUT ANY WARRANTY; without even the implied warranty of
#     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#     GNU General Public License for more details.

import os
import sqlite3
import re
import sys
import subprocess

import tornado.httpserver
import tornado.ioloop
import tornado.web
import tornado.options
import tornado.autoreload

import simplejson as json
from tornado.options import define, options


define("port", default=8080, help="run␣on␣the␣given␣port", type=int)

class Application(tornado.web.Application):
        def __init__(self):
```

```
                handlers = [
                (r"/", MainHandler),
                (r"/rulesets", RulesetsHandler),
                (r"/rules", RulesHandler),
                (r"/sensors", SensorsHandler),
                (r"/get_rulesets", GetRulesetsHandler),
                (r"/get_rules", GetRulesHandler),
                (r"/get_sensors", GetSensorsHandler),
                (r"/add_sensor", AddSensorHandler),
                (r"/remove_sensor", RemoveSensorHandler),
                (r"/open_rule", OpenRuleHandler),
                (r"/getsensorname", GetSensorNameHandler),
                (r"/tuning_rules", TuningRulesHandler),
                (r"/tuning_rulesets", TuningRulesetsHandler),
                (r"/update_sensor", UpdateSensorHandler),
                (r"/update", UpdateHandler),
                (r"/atuninghelp", ATuningHelpHandler),
                (r"/suppress", SuppressHandler),
                (r"/threshold", ThresholdHandler),
                (r"/atuning", ATuningHandler),
                (r"/get_atuning", GetATuningHandler),
                (r"/remove_atuning", RemoveATuningHandler),
                (r"/distribute", DistributeHandler),
                ]
                settings = dict(
                #login_url="/auth/login",
                template_path=os.path.join(os.path.dirname(__file__),
                    "templates"),
                static_path=os.path.join(os.path.dirname(__file__), "
                    static"),
                autoescape=None)
                tornado.web.Application.__init__(self, handlers, **
                    settings)

class RemoveATuningHandler(tornado.web.RequestHandler):
        def post(self):
                syntax = self.request.arguments.get("atuningid")

                db = sqlite3.connect('../DB.db')
                cursor = db.cursor()

                try:
                        cursor.execute('SELECT␣sname␣FROM␣sensors')
                        all_sensors = cursor.fetchall()
                        for hit in all_sensors:
                                table = hit[0]+'_threshold'
                                sql = 'DELETE␣FROM␣%s␣WHERE␣syntax="%
                                    s"' % (table,syntax[0])
                                cursor.execute(sql)
```

```
                db.commit()

            except StandardError,e:
                FILE = open('weberrorlog.txt','a')
                FILE.write('RemoveATuningHandler ERROR: '+str
                    (e)+'\n')
                FILE.close()

            cursor.close()
            db.close()

class GetATuningHandler(tornado.web.RequestHandler):
    def get(self):
            db = sqlite3.connect('../DB.db')
            cursor = db.cursor()

            atuning = []

            try:
                cursor.execute('SELECT sname FROM sensors')
                all_sensors = cursor.fetchall()
                for hit in all_sensors:
                    table = hit[0]+'_threshold'
                    sql = 'SELECT * FROM '+table
                    cursor.execute(sql)
                    for row in cursor:
                        idnr,sid,typ,syntax,comment,
                            sensor = row
                        check = "<center><input type
                            ='checkbox' name='
                            atuningid' value='%s'></
                            center>" % (syntax)
                        tmp = (check,sid,typ,syntax,
                            comment,sensor)
                        if tmp not in atuning:
                            atuning.append(tmp)

            except StandardError,e:
                FILE = open('weberrorlog.txt','a')
                FILE.write('GetATuningHandler ERROR: '+str(e)
                    +'\n')
                FILE.close()

            cursor.close()
            db.close()
            self.write(json.dumps({"aaData":atuning},sort_keys=
                True,indent=4))
```

```
class ThresholdHandler(tornado.web.RequestHandler):
        def post(self):
                db = sqlite3.connect('../DB.db')
                cursor = db.cursor()

                if 'sigid' not in self.request.arguments:
                        self.write('Input missing. Try again.')

                elif 'count' not in self.request.arguments:
                        self.write('Input missing. Try again.')

                elif 'sec' not in self.request.arguments:
                        self.write('Input missing. Try again.')

                else:
                        genid = self.request.arguments.get("genid")
                        sigid = self.request.arguments.get("sigid")
                        typ = self.request.arguments.get("type")
                        track = self.request.arguments.get("track")
                        count = self.request.arguments.get("count")
                        sec = self.request.arguments.get("sec")
                        sensor = self.request.arguments.get("select")
                        comment = ''

                        if 'comment' in self.request.arguments:
                                tmp = self.request.arguments.get("
                                    comment")
                                comment = tmp[0]

                        syntax = 'event_filter gen_id '+genid[0]+',
                            sig_id '+sigid[0]+',type '+typ[0]+',track
                            '+track[0]+',count '+count[0]+',seconds '+
                            sec[0]

                        try:
                                def insert_t(table,x):
                                        sql = 'INSERT OR IGNORE INTO
                                            '+table+' (id,sid,type,
                                            syntax,comment,sensor)
                                            VALUES (null,'+sigid[0]+'
                                            ,"threshold","'+syntax+'
                                            ","'+comment+'","'+x+'")'
                                        cursor.execute(sql)

                                if not (sensor[0] == "all"):
                                        table = sensor[0]+'_threshold
                                            '
                                        insert_t(table,sensor[0])
```

```
                                else:
                                        cursor.execute('SELECT␣sname␣
                                            FROM␣sensors')
                                        all_sensors = cursor.fetchall
                                            ()
                                        for hit in all_sensors:
                                                table = hit[0]+'
                                                    _threshold'
                                                insert_t(table,'ALL')

                                db.commit()
                                self.write('threshold␣rule␣for␣sid:␣'
                                    +sigid[0]+'␣has␣been␣added!')

                        except StandardError,e:
                                FILE = open('weberrorlog.txt','a')
                                FILE.write('ThresholdHandler␣ERROR:␣'
                                    +str(e)+'\n')
                                FILE.close()
                                self.write(str(e))

                cursor.close()
                db.close()

class SuppressHandler(tornado.web.RequestHandler):
        def post(self):
                db = sqlite3.connect('../DB.db')
                cursor = db.cursor()

                if 'sigid' not in self.request.arguments:
                        self.write('Input␣missing.␣Try␣again.')

                elif 'ip' not in self.request.arguments:
                        self.write('Input␣missing.␣Try␣again.')

                else:
                        genid = self.request.arguments.get("genid")
                        sigid = self.request.arguments.get("sigid")
                        track = self.request.arguments.get("track")
                        ip = self.request.arguments.get("ip")
                        sensor = self.request.arguments.get("select")
                        comment = ''

                        if 'comment' in self.request.arguments:
                                tmp = self.request.arguments.get("
                                    comment")
                                comment = tmp[0]
```

```
                        syntax = 'suppress␣gen_id␣'+genid[0]+',sig_id
                            ␣'+sigid[0]+',track␣'+track[0]+',ip␣'+ip
                            [0]

                        try:
                                def insert_t(table,x):
                                        sql = 'INSERT␣OR␣IGNORE␣INTO␣
                                            '+table+'␣(id,sid,type,
                                            syntax,comment,sensor)␣
                                            VALUES␣(NULL,'+sigid[0]+'
                                            ,"suppress","'+syntax+'","
                                            '+comment+'","'+x+'")'
                                        cursor.execute(sql)

                                if not (sensor[0] == "all"):
                                        table = sensor[0]+'_threshold
                                            '
                                        insert_t(table,sensor[0])

                                else:
                                        cursor.execute('SELECT␣sname␣
                                            FROM␣sensors')
                                        all_sensors = cursor.fetchall
                                            ()
                                        for hit in all_sensors:
                                                table = hit[0]+'
                                                    _threshold'
                                                insert_t(table,'ALL')

                                db.commit()
                                self.write('suppress␣rule␣for␣sid:␣'+
                                    sigid[0]+'␣has␣been␣added!')

                        except StandardError,e:
                                FILE = open('weberrorlog.txt','a')
                                FILE.write('ThresholdHandler␣ERROR:␣'
                                    +str(e)+'\n')
                                FILE.close()
                                self.write(str(e))

                cursor.close()
                db.close()

class DistributeHandler(tornado.web.RequestHandler):
        def get(self):
                self.write('''<html  xmlns="http://www.w3.org/1999/
                    xhtml">
<head>
<title>Distribute  report</title>
```

```
<link type="text/css" rel="stylesheet" href="../static/css/custom.css
    "/>
<link type="text/css" rel="stylesheet" href="../static/css/demo_page.
    css"/>
</head>
<body>
 <b>Distribute report</b></br>''')

                try:
                        p = subprocess.Popen(["python","../distribute
                            .py"], stdout=subprocess.PIPE)
                        for line in iter(p.stdout.readline, ''):
                                self.write(' ')
                                self.write(line)
                                self.write('</br>')
                        p.stdout.close()

                except StandardError,e:
                        FILE = open('weberrorlog.txt','a')
                        FILE.write('DistributeHandler ERROR: '+str(e)
                            +'\n')
                        FILE.close()

                self.write('''</body>
</html>''')


class UpdateHandler(tornado.web.RequestHandler):
        def get(self):
                self.write('''<html xmlns="http://www.w3.org/1999/
                    xhtml">
<head>
<title>Update report</title>
<link type="text/css" rel="stylesheet" href="../static/css/custom.css
    "/>
<link type="text/css" rel="stylesheet" href="../static/css/demo_page.
    css"/>
</head>
<body>
 <b>Update Report</b></br>''')

                try:
                        p = subprocess.Popen(["python","../update.py"
                            ], stdout=subprocess.PIPE)
                        for line in iter(p.stdout.readline, ''):
                                self.write(' ')
                                self.write(line)
                                self.write('</br>')
                        p.stdout.close()
```

```
                except StandardError,e:
                        FILE = open('weberrorlog.txt','a')
                        FILE.write('UpdateHandler ERROR: '+str(e)+'\n
                            ')
                        FILE.close()

                self.write('''</body>
</html>''')

class UpdateSensorHandler(tornado.web.RequestHandler):
        def post(self):
                db = sqlite3.connect('../DB.db')
                cursor = db.cursor()

                sensor = self.request.arguments.get("select")

                try:
                        if not (sensor[0] != 'all'):
                                cursor.execute('SELECT sname FROM
                                    sensors')
                                all_sensors = cursor.fetchall()

                        def update(f,v,s):
                                sql = 'UPDATE sensors SET '+f+'="'+v+
                                    '" WHERE sname="'+s+'"'
                                cursor.execute(sql)

                        if "ip" in self.request.arguments:
                                ip = self.request.arguments.get("ip")
                                if not (sensor[0] == 'all'):
                                        update("ip",ip[0],sensor[0])
                                else:
                                        for hit in all_sensors:
                                                update("ip",ip[0],hit
                                                    [0])

                        if "path" in self.request.arguments:
                                path = self.request.arguments.get("
                                    path")
                                if not (sensor[0] == 'all'):
                                        update("path",path[0],sensor
                                            [0])
                                else:
                                        for hit in all_sensors:
                                                update("path",path
                                                    [0],hit[0])

                        if "uname" in self.request.arguments:
```

```
                                uname = self.request.arguments.get("
                                    uname")
                                if not (sensor[0] == 'all'):
                                        update("uname",uname[0],
                                            sensor[0])
                                else:
                                        for hit in all_sensors:
                                                update("uname",uname
                                                    [0],hit[0])

                        if "cmd" in self.request.arguments:
                                pw = self.request.arguments.get("cmd"
                                    )
                                if not (sensor[0] == 'all'):
                                        update("cmd",cmd[0],sensor
                                            [0])
                                else:
                                        for hit in all_sensors:
                                                update("cmd",cmd[0],
                                                    hit[0])

                        db.commit()
                        self.write('Sensor updated! Refresh page to
                            see changes.')

                except StandardError,e:
                        FILE = open('weberrorlog.txt','a')
                        FILE.write('UpdateSensorHandler ERROR: '+str(
                            e)+'\n')
                        FILE.close()
                        self.write(str(e))

                cursor.close()
                db.close()

class TuningRulesetsHandler(tornado.web.RequestHandler):
        def post(self):
                source_ruleset = self.request.arguments.get("
                    rulesetid")
                sensor = self.request.arguments.get("sensor")
                action = self.request.arguments.get("action")

                db = sqlite3.connect('../DB.db')
                cursor = db.cursor()

                sids = ''

                try:
                        def disable_sid(table,sid):
```

```
                value = sid.split(',')
                for entry in value:
                        sql = 'INSERT OR IGNORE INTO
                            '+table+' (sid) VALUES ('+
                            entry+')'
                        cursor.execute(sql)

        def enable_sid(table,sid):
                sql = 'DELETE FROM '+table+' WHERE
                    sid IN ('+sid+')'
                cursor.execute(sql)

        length = len(source_ruleset)
        counter = 1
        for hit in source_ruleset:
                split = hit.split('.')
                sql = 'SELECT sidnr from rules WHERE
                    source_name="'+split[0]+'" AND
                    ruleset_name="'+split[1]+'"'
                cursor.execute(sql)
                tmp = cursor.fetchall()
                sids = sids+(",".join(str(x[0]) for x
                     in tmp))
                if not (counter == length):
                        sids = sids+","
                counter += 1

        if not (sensor[0] == 'all'):
                table = sensor[0]+'_disabled'
                if not (action[0] == "enable"):
                        disable_sid(table,sids)
                else:
                        enable_sid(table,sids)

        else:
                cursor.execute('SELECT sname FROM
                    sensors')
                all_sensors = cursor.fetchall()
                for hit in all_sensors:
                        table = hit[0]+'_disabled'
                        if not (action[0] == "enable"
                            ):
                                disable_sid(table,
                                    sids)
                        else:
                                enable_sid(table,sids
                                    )

        db.commit()
```

```
                except StandardError,e:
                        FILE = open('weberrorlog.txt','a')
                        FILE.write('TuningRulesetsHandler ERROR: '+
                            str(e)+'\n')
                        FILE.close()

                cursor.close()
                db.close()

class TuningRulesHandler(tornado.web.RequestHandler):
        def post(self):
                sids = self.request.arguments.get('sidnr')
                sensor = self.request.arguments.get('sensor')
                action = self.request.arguments.get('action')

                db = sqlite3.connect('../DB.db')
                cursor = db.cursor()

                def disable_sid(table,sid):
                        sql = 'INSERT OR IGNORE INTO '+table+' (sid)
                            VALUES ('+sid+')'
                        cursor.execute(sql)

                def enable_sid(table,sid):
                        sql = 'DELETE FROM '+table+' WHERE sid='+sid
                        cursor.execute(sql)

                try:
                        if not (sensor[0] == "all"):
                                table = sensor[0]+'_disabled'
                                for sid in sids:
                                        if not (action[0] == "enable"
                                            ):
                                                disable_sid(table,sid
                                                    )
                                        else:
                                                enable_sid(table,sid)
                        else:
                                cursor.execute('SELECT sname FROM
                                    sensors')
                                all_sensors = cursor.fetchall()
                                for hit in all_sensors:
                                        table = hit[0]+'_disabled'
                                        for sid in sids:
                                                if not (action[0] ==
                                                    "enable"):
                                                        disable_sid(
                                                            table,sid)
```

```
                                                else:
                                                        enable_sid(
                                                            table, sid)

                        db.commit()

                except StandardError,e:
                        FILE = open('weberrorlog.txt','a')
                        FILE.write('TuningRulesHandler ERROR: '+str(e
                            )+'\n')
                        FILE.close()

                cursor.close()
                db.close()

class GetSensorNameHandler(tornado.web.RequestHandler):
        def get(self):
                db = sqlite3.connect('../DB.db')
                cursor = db.cursor()

                try:
                        cursor.execute('SELECT sname FROM sensors')
                        selectbox = '<select name="select" id="select
                            "><option value="all">all sensors</option>
                            '
                        for sensor in cursor:
                                selectbox = selectbox+'<option value
                                    ="'+sensor[0]+'">'+sensor[0]+'</
                                    option>'

                        selectbox = selectbox+'</select>'
                        self.write(selectbox)
                except StandardError,e:
                        FILE = open("weberrorlog.txt","a")
                        FILE.write("GetSensorNameHandler ERROR: "+str
                            (e)+"\n")
                        FILE.close()
                        self.write('<select><option>ERROR</option></
                            select>')

                cursor.close()
                db.close()

class OpenRuleHandler(tornado.web.RequestHandler):
        def get(self):
                sid = self.get_argument("sid")
                db = sqlite3.connect('../DB.db')
                cursor = db.cursor()
```

```
                try:
                        cursor.execute('SELECT␣rule_syntax␣FROM␣rules
                            ␣WHERE␣sidnr␣=␣(?)', [sid])
                        rulesyntax = cursor.fetchone()
                        self.render("open_rules.html",rulesyntax=
                            rulesyntax[0])

                except StandardError,e:
                        FILE = open('weberrorlog.txt','a')
                        FILE.write('OpenRuleHandler␣ERROR:␣'+str(e)+'
                            \n')
                        FILE.close()

                cursor.close()
                db.close()

class RemoveSensorHandler(tornado.web.RequestHandler):
        def post(self):
                snames = self.request.arguments.get("sensorid")

                db = sqlite3.connect('../DB.db')
                cursor = db.cursor()

                try:
                        for sensor in snames:
                                sql = 'DELETE␣␣FROM␣sensors␣WHERE␣
                                    sname="%s"' % (sensor)
                                cursor.execute(sql)
                                sql = 'DROP␣TABLE␣%s_disabled' % (
                                    sensor)
                                cursor.execute(sql)
                                sql = 'DROP␣TABLE␣%s_threshold' % (
                                    sensor)
                                cursor.execute(sql)

                        db.commit()

                except StandardError,e:
                        FILE = open('weberrorlog.txt','a')
                        FILE.write('RemoveSensorHandler␣ERROR:␣'+str(
                            e)+'\n')
                        FILE.close()

                cursor.close()
                db.close()

class AddSensorHandler(tornado.web.RequestHandler):
        def post(self):
                db = sqlite3.connect('../DB.db')
```

```
cursor = db.cursor()

if 'sname' not in self.request.arguments:
        self.write('Sensor NOT added. Input missing.
            Try again.')

elif 'ip' not in self.request.arguments:
        self.write('Sensor NOT added. Input missing.
            Try again.')

elif 'path' not in self.request.arguments:
        self.write('Sensor NOT added. Input missing.
            Try again.')

elif 'uname' not in self.request.arguments:
        self.write('Sensor NOT added. Input missing.
            Try again.')

elif 'cmd' not in self.request.arguments:
        self.write('Sensor NOT added. Input missing.
            Try again.')

else:
        sname = self.request.arguments.get("sname")
        sname = sname[0]
        ip = self.request.arguments.get("ip")
        ip = ip[0]
        path = self.request.arguments.get("path")
        path = path[0]
        uname = self.request.arguments.get("uname")
        uname = uname[0]
        cmd = self.request.arguments.get("cmd")
        cmd = cmd[0]

        try:
                db = sqlite3.connect('../DB.db')
                cursor = db.cursor()
                cursor.execute('''INSERT INTO sensors
                    (sname,ip,path,uname,cmd)
                                VALUES(?,?,?,?,?) '''
                                    ,(sname,ip,path,
                                    uname,cmd))
                sql = 'CREATE TABLE '+sname+'
                    _disabled (sid INTEGER PRIMARY KEY
                    )'
                cursor.execute(sql)
                sql = 'CREATE TABLE '+sname+'
                    _threshold (id INTEGER PRIMARY KEY
                    , sid INTEGER, type TEXT, syntax
```

```
                                    TEXT, comment TEXT, sensor TEXT)'
                                cursor.execute(sql)
                                self.write(sname+' added! Refresh
                                    page to see changes.')
                                db.commit()

                        except StandardError,e:
                                FILE = open('weberrorlog.txt','a')
                                FILE.write('AddSensorHandler ERROR: '
                                    +str(e)+'\n')
                                FILE.close()
                                self.write(str(e))

                cursor.close()
                db.close()

class GetSensorsHandler(tornado.web.RequestHandler):
        def get(self):

                db = sqlite3.connect('../DB.db')
                cursor = db.cursor()

                sensors = []

                try:
                        cursor.execute('SELECT * FROM sensors')
                        for row in cursor:
                                sname,ip,path,uname,cmd = row
                                check = "<center><input type='
                                    checkbox' name='sensorid' value='%
                                    s'></center>" % (sname)
                                sensor = (check,sname,ip,path,uname,
                                    cmd)
                                sensors.append(sensor)

                except StandardError,e:
                        FILE = open('weberrorlog.txt','a')
                        FILE.write('GetSensorsHandler ERROR: '+str(e)
                            +'\n')
                        FILE.close()

                cursor.close()
                db.close()
                self.write(json.dumps({"aaData":sensors},sort_keys=
                    True,indent=4))

class GetRulesHandler(tornado.web.RequestHandler):
        def get(self):
                db = sqlite3.connect('../DB.db')
```

```
cursor = db.cursor()

details = '<img␣class="sig"␣src="static/images/open.
    png">'
sigs = []

try:
        cursor.execute('SELECT␣*␣FROM␣rules')
        all_rules = cursor.fetchall()
        cursor.execute('SELECT␣sname␣FROM␣sensors')
        all_sensors = cursor.fetchall()

        for row in all_rules:
                sidnr,revnr,source,ruleset,name,ref,
                    date,rule = row
                status =''
                for hit in all_sensors:
                        sql = 'SELECT␣sid␣FROM␣'+hit
                            [0]+'_disabled␣WHERE␣sid='
                            +str(sidnr)
                        cursor.execute(sql)
                        res = cursor.fetchone()
                        sql = 'SELECT␣sid␣FROM␣%
                            s_threshold␣WHERE␣sid="%s"
                            ' % (hit[0],sidnr)
                        cursor.execute(sql)
                        tmp2 = cursor.fetchone()
                        if not (res is None):
                                if not (tmp2 is None)
                                    :
                                        status =
                                            status+'<
                                            font␣class
                                            ="red">'+
                                            hit[0]+'</
                                            font><font
                                            ␣class="
                                            yellow"><b
                                            >!</b></
                                            font> 
                                            ;' #red/
                                            yellow
                                else:
                                        status =
                                            status+'<
                                            font␣class
                                            ="red">'+
                                            hit[0]+'</
                                            font> 
```

```
                                                    ;' #red
                                            else:
                                                    if not (tmp2 is None)
                                                        :
                                                            status =
                                                                status+'<
                                                                font class
                                                                ="green">'
                                                                +hit[0]+'
                                                                </font><
                                                                font class
                                                                ="yellow
                                                                "><b>!</b
                                                                ></font>&
                                                                nbsp;' #
                                                                green/
                                                                yellow
                                                    else:
                                                            status =
                                                                status+'<
                                                                font class
                                                                ="green">'
                                                                +hit[0]+'
                                                                </font>&
                                                                nbsp;' #
                                                                green

                                        check = '<input type="checkbox" name
                                            ="sidnr" value="%i">' % (sidnr)
                                        source_ruleset = '%s.%s' % (source,
                                            ruleset)
                                        sig = (check, sidnr, revnr, date,
                                            name, source_ruleset, ref, status,
                                             details)
                                        sigs.append(sig)

                            except StandardError,e:
                                    FILE = open('weberrorlog.txt','a')
                                    FILE.write('GetRulesetsHandler ERROR: '+str(e
                                        )+'\n')
                                    FILE.close()

                            cursor.close()
                            db.close()
                            self.write(json.dumps({"aaData":sigs},sort_keys=True,
                                indent=4))

class GetRulesetsHandler(tornado.web.RequestHandler):
        def get(self):
```

```
db = sqlite3.connect('../DB.db')
cursor = db.cursor()

rulesets = []

try:
        cursor.execute("SELECT␣DISTINCT␣ruleset_name,
            ␣source_name␣FROM␣rules")
        query = cursor.fetchall()

        for row in query:
                ruleset,source = row
                source_ruleset = '%s.%s' % (source,
                    ruleset)
                check = '<center><input␣type="
                    checkbox"␣name="rulesetid"␣value
                    ="%s"></center>' % (source_ruleset
                    )
                sql = 'SELECT␣sidnr␣from␣rules␣WHERE␣
                    source_name="%s"␣AND␣ruleset_name
                    ="%s"' % (source,ruleset)
                cursor.execute(sql)
                tmp = cursor.fetchall()
                count = len(tmp)
                sids = ','.join(str(x[0]) for x in
                    tmp)
                cursor.execute('SELECT␣sname␣FROM␣
                    sensors')
                all_sensors = cursor.fetchall()
                sql = 'SELECT␣MAX(date)␣FROM␣rules␣
                    WHERE␣source_name="%s"␣AND␣
                    ruleset_name="%s"' % (source,
                    ruleset)
                cursor.execute(sql)
                max_date = cursor.fetchone()

                status = ''
                for x in all_sensors:
                        sensor = x[0]
                        sql = 'SELECT␣sid␣FROM␣%
                            s_disabled␣WHERE␣sid␣IN␣(␣
                            %s␣)' % (sensor,sids)
                        cursor.execute(sql)
                        tmp2 = cursor.fetchall()
                        scount = len(tmp2)
                        if not (scount == count):
                                if not (scount == 0):
                                        status =
                                            status+'<
```

```
                                                font␣class
                                                ="green">%
                                                s</font><
                                                font␣class
                                                ="red">%s
                                                </font>&
                                                nbsp;' % (
                                                sensor,
                                                scount)
                                        else:
                                                status =
                                                    status+'<
                                                    font␣class
                                                    ="green">%
                                                    s</font>&
                                                    nbsp;' %
                                                    sensor
                                else:
                                        status = status+'<
                                            font␣class="red">%
                                            s</font> ' %
                                            sensor

                                rset = (check,source_ruleset,max_date
                                    ,count,status)
                                rulesets.append(rset)

                except StandardError,e:
                        FILE = open('weberrorlog.txt','a')
                        FILE.write('GetRulesetsHandler␣ERROR:␣'+str(e
                            )+'\n')
                        FILE.close()

                cursor.close()
                db.close()
                self.write(json.dumps({"aaData":rulesets},sort_keys=
                    True,indent=4))

class ATuningHandler(tornado.web.RequestHandler):
        def get(self):
                self.render("atuning.html")

class ATuningHelpHandler(tornado.web.RequestHandler):
        def get(self):
                self.render("atuninghelp.html")

class SensorsHandler(tornado.web.RequestHandler):
        def get(self):
                self.render("sensors.html")
```

```
class RulesHandler(tornado.web.RequestHandler):
        def get(self):
                self.render("rules.html")

class RulesetsHandler(tornado.web.RequestHandler):
        def get(self):
                self.render("rulesets.html")

class MainHandler(tornado.web.RequestHandler):
        def get(self):
                self.render("index.html")

def main():
        tornado.options.parse_command_line()
        http_server = tornado.httpserver.HTTPServer(Application())
        http_server.listen(options.port)
        tornado.autoreload.start()
        tornado.ioloop.IOLoop.instance().start()

if __name__ == "__main__":
        main()
```

# E config.py

The configuration file.

```
#!/usr/bin/env python
# CONFIG FILE

C_rule_sources = []
C_files = []
C_db_name = 'DB.db'

########## DO NOT MODIFY VARIABLES ABOVE THIS LINE  ##########


# Specify path to tmp. dir. (must be manually created). Must be full
    path and path must end with a trailing slash.
# Example: '/home/user/bringhomethebacon/tmp/'
C_tmp_dir = ''

# Specify path to ditribute dir - dir that contains all files to be
    ditributed (must be manually created). Must be full path and path
    must end with a trailing slash.
# Example: '/home/user/bringhomethebacon/files_to distribute/'
C_distribute_dir = ''

# Specify path to sms folder. Must be full path and path and path
    must end with a trailing slash.
# Example: '/home/user/bringhomethebacon/'
C_db_path = ''+C_db_name

# Specify path to local rules dir. Must be full path and path must
    end with a trailing slash.
# Example: '/home/user/rules/'
# Leave blank if not using local rules or choosing to use web-GUI to
    add rules.
C_locale_rule_path = ''

# Specify path and name for update.py logfile. Must be full path.
# Example: '/home/user/bringhomethebacon/updatelog.txt'
C_updatelog = ''

# Specify path and name for distribute.py logfile. Must be full path.
# Example: '/home/user/bringhomethebacon/distributelog.txt'
C_distrblog = ''
```

```
# NB: Remember to set correct permissions on folders and files!


# Select files to be moved to the distribute dir. If file already
    exists in that dir, it will be updated if any changes.
# The file-folder option must be specified in external sources for
    this to work. Use default values if not sure.
C_files.append(['reference.config'])
C_files.append(['sid-msg.map'])
C_files.append(['gen-msg.map'])
C_files.append(['classification.config'])
C_files.append(['unicode.map'])


######################################## ADD EXTERNAL RULE SOURCES
    BELOW ##################################################
# INFO:
#
# Use this syntax to add more rule sources: rule_sources.append([
    rule-source-name, md5-url*, rule-url**, rule-folder***, file-
    folder****])
#
# * Rule-url must point to a tar.gz file
# ** Md5-url is optional. if no md5-url, keep it blank (''), else md5
    -url must point to a file that only contains md5 sum
# *** Rule-folder is the path (within the tar.gz) to where the rule
    files reside. If no folder, keep it blank: ''
# **** File-folder is the path (within the tar.gz) to where gen-msg.
    map, reference.config etc. reside.
# If no folder, keep it blank: ''. Else if the tar.gz dont have these
     files, use none: 'none'.
#
# See VRT and ET example below for more information.
#
    ##########################################################################


## VRT-rules

# Specify snort version. Example: '2920'.  Use snort -V to get snort
    version
C_vrt_snort_version = ''

C_oinkcode = ''

C_rule_sources.append([
        'Sourcefire',
```

```
        'http://www.snort.org/reg-rules/snortrules-snapshot-'+
            C_vrt_snort_version+'.tar.gz.md5/'+C_oinkcode,
        'http://www.snort.org/reg-rules/snortrules-snapshot-'+
            C_vrt_snort_version+'.tar.gz/'+C_oinkcode,
        'rules/',
        'etc/'
        ])

## ET-rules

# Specify snort version. Example: '2.9.0'.  Use snort -V to get snort
    version
C_et_snort_version = ''

C_rule_sources.append([
        'Emerging Threats',
        'http://rules.emergingthreats.net/open-nogpl/snort-'+
            C_et_snort_version+'/emerging.rules.tar.gz.md5',
        'http://rules.emergingthreats.net/open-nogpl/snort-'+
            C_et_snort_version+'/emerging.rules.tar.gz',
        'rules/',
        'rules/',
        ])
```

# F templates.html

One of the many templates that makes up the graphical user interface.

```
### Rule-view template ###

{% extends "base.html" %}
{% block head %}

<script type="text/javascript" charset="utf-8">
        $(document).ready(function(){
                var oTable = $('#dataTable').dataTable( {
                        "bProcessing": true,
                        "sAjaxSource": '/get_rules',
                        "bDeferRender": true,
                        "bJQueryUI": true,
                        "bPaginate": true,
                        "sPaginationType": "full_numbers",
                        "bStateSave": false,
                        "bAutoWidth": false,
                        "iDisplayLength": 50,
                        "sDom": '<"header ui-toolbar ui-widget-header
                            ui-corner-tl ui-corner-tr ui-helper-
                           clearfix"fr>t<"footer ui-toolbar ui-widget
                          -header ui-corner-bl ui-corner-br ui-
                           helper-clearfix"ipl>',
                        "aaSorting": [[3,'desc']],
                        "aoColumns":[
                                { "bSearchable": false, "bSortable":
                                    false, "sWidth": "15px"},
                                {"sWidth": "60px"},
                                { "bSearchable": false, "bSortable":
                                    false, "sWidth": "30px"},
                                { "sWidth": "80px"},
                                null,
                                null,
                                { "bSearchable": false, "bSortable":
                                    false},
                                { "bSearchable": false, "bSortable":
                                    false},
                                { "bSearchable": false, "bSortable":
                                    false, "sWidth": "15px"}
                        ]
                }).fnSetFilteringDelay(1000);
```

```
new FixedHeader( oTable );
$('td img').live('click', function () {
        var nTr = this.parentNode.parentNode;
        if (this.src.match('close')){
                /* This row is already open - close
                    it */
                this.src = "static/images/open.png";
                oTable.fnClose( nTr );
        }
        else {
                /* Open this row */
                row = oTable.fnGetData(nTr);
                this.src = "/static/images/close.png
                    ";
                $.get('/open_rule',{'sid':row[1]},
                    function(data) {
                         oTable.fnOpen(nTr, data);
                });
        }
});
$.get("/getsensorname",function(data) {
        $("div.header").append('<span id="radio"><
            input type="radio" id="radio1" name="
            choice" value="disable" checked="checked
            "/><label for="radio1">Disable</label><
            input type="radio" id="radio2" name="
            choice" value="enable" /><label for="
            radio2">Enable</label></span><span id="
            select1">selected ( <span id="options"></
            span> ), on sensor: </span><span id="
            select2> ',data,' <a class="
            submitbutton">Execute</a></span> OR use: <
            span id="atuning"></span>');
        $("#thr-sensor").append(data);
        $("#sup-sensor").append(data);

        $( "#radio" ).buttonset();

        $(".submitbutton").button().click(function()
            {
                var sData = $('input', oTable.
                    fnGetNodes()).serialize();
                var sensor = $('[name=select]').val()
                    ;
                var action = $('input[name=choice]:
                    checked').val();
                $.post("/tuning_rules", "sensor="+
                    sensor+"&action="+action+"&"+sData
                    ,
```

```
                function() {oTable.
                    fnReloadAjax('/get_rules')
                    ;});
});

$("#options").append('<a class="selectall">
    Select All</a>');

$(".selectall").button().click(function() {
        $('input', oTable.fnGetDisplayNodes()
            ).attr('checked','checked');
});

$("#options").append('<a class="deselectall">
    Select None</a>');

$(".deselectall").button().click(function() {
        $('input', oTable.fnGetDisplayNodes()
            ).attr('checked',false);
});

$("#atuning").append('<a class="thr">T..</a>&
    nbsp');
$("#atuning").append('<a class="sup">S..</a
    >');

$(".thr").button({icons: {primary: "ui-icon-
    newwin"}}).click(function() {
        $('#thr-dialog').dialog('open');
});
$(".sup").button({icons: {primary: "ui-icon-
    newwin"}}).click(function() {
        $('#sup-dialog').dialog('open');
});

$("#thr-dialog").dialog({
        autoOpen: false,
        width: 750,
        buttons: {
                'Submit': function() {
                        fData = $('#thrform')
                            .serialize();
                        $.post("/threshold",
                            fData,function(
                            data) {
                                alert(data);
                        });
                        $(this).dialog('close
                            ');
```

```
                                        },
                                        'Help': function() {
                                                window.open("/
                                                    atuninghelp");
                                        }
                                }
                        });
                        $("#sup-dialog").dialog({
                                autoOpen: false,
                                width: 550,
                                buttons: {
                                        'Submit': function() {
                                                fData = $('#supform')
                                                    .serialize();
                                                $.post("/suppress",
                                                    fData,function(
                                                    data) {
                                                        alert(data);
                                                });
                                                $(this).dialog('close
                                                    ');
                                        },
                                        'Help': function() {
                                                window.open("/
                                                    atuninghelp");
                                        }
                                }
                        });

                });

        });
</script>
{% end %}

{% block body %}

<div id="container">
        <div id="header">
                <div id="toolbar">
                </div>
                <table width="100%" cellspacing="0" border="0" id="
                    dataTable">
                        <thead>
                                <tr>
                                <th></th>
                                <th>Sid</th>
                                <th>Rev</th>
                                <th>Added</th>
```

```
                                <th>Name</th>
                                <th>Source.Ruleset</th>
                                <th>Reference</th>
                                <th>Status information</th>
                                <th></th>
                                </tr>
                        </thead>
                        <tbody>
                        </tbody>
                </table>
        </div>
</div>
<div id="thr-dialog" title="Thresholding" style="display:none;">
        <form id="thrform" method="post">
                <fieldset>
                        event_filter <label for="genid">gen_id</
                            label>
                        <input type="text" size="3" value="1" name="
                            genid" id="genid" class="text ui-widget-
                            content ui-corner-all" />
                        , 
                        <label for="sigid">sig_id</label>
                        <input type="text" size="8" name="sigid" id="
                            sigid" class="text ui-widget-content ui-
                            corner-all" />
                        , 
                        <label for="type">type</label>
                        <select name="type" id="type">
                                <option value="limit">limit</option>
                                <option value="threshold">threshold</
                                    option>
                                <option value="both">both</option>
                        </select>
                        , 
                        <label for="track">track</label>
                        <select name="track" id="track">
                                <option value="by_src">by_src</option
                                    >
                                <option value="by_dst">by_dst</option
                                    >
                        </select>
                        , 
                        <label for="count">count</label>
                        <input type="text" size="3"  name="count" id
                            ="count" class="text ui-widget-content ui-
                            corner-all" />
                        , 
                        <label for="sec">seconds</label>
```

```
                        <input type="text" size="3"  name="sec" id="
                            sec" class="text ui-widget-content ui-
                            corner-all" />
                        </br>
                        <label for="select"></label>
                        </br>
                        on sensor: 
                        <span id="thr-sensor"></span>
                         
                        <label for="comment">comment:</label>
                        <input type="text" size="50" name="comment"
                            id="comment" class="text ui-widget-content
                             ui-corner-all" />
                </fieldset>
        </form>
</div>
<div id="sup-dialog" title="Suppression" style="display:none;">
        <form id="supform" method="post">
                <fieldset>
                        suppress 
                        <label for="genid">gen_id</label>
                        <input type="text" size="3" value="1"  name="
                            genid" id="genid" class="text ui-widget-
                            content ui-corner-all" />
                        , 
                        <label for="sigid">sig_id</label>
                        <input type="text" size="8" name="sigid" id="
                            sigid" class="text ui-widget-content ui-
                            corner-all" />
                        , 
                        <label for="track">track</label>
                        <select name="track" id="track">
                                <option value="by_src">by_src</option
                                    >
                                <option value="by_dst">by_dst</option
                                    >
                        </select>
                        , 
                        <label for="ip">ip</label>
                        <input type="text" size="15" name="ip" id="ip
                            " class="text ui-widget-content ui-corner-
                            all" />
                        </br>
                        <label for="select"></label>
                        </br>
                        on sensor: 
                        <span id="sup-sensor"></span>
                         
                        <label for="comment">comment:</label>
```

```
                        <input type="text" size="50" name="comment"
                            id="comment" class="text ui-widget-content
                             ui-corner-all" />
                </fieldset>
        </form>
</div>
{% end %}
```