SNORT Rule Manager

A complete, effective and secure rule management system for SNORT®

Project Planning Document

Thomas Nyheim

Eirik Skogstad

Eigil Obrestad

Bachelor's Thesis

Department of Computer Science and Media Technology

Gjøvik University College, Spring 2014

CONTENTS

CONTENTS 3

1. OBJECTIVES AND CONSTRAINS 4

2. SCOPE 8

3. PROJECT ORGANIZATION 11

4. DEVELOPMENT MODEL 13

5. QUALITY ASSURANCE 15

6. PROJECT TIMELINE 16

APPENDIX A - CITATIONS 17

1. OBJECTIVES AND CONSTRAINS

1.1. Background

The Norwegian Armed Forces Cyber Defense (CYFOR) unit for Computer Network Defense (CND), Avdeling BKI, is responsible for detecting and stopping cyber-attacks against the critical infrastructure and command and control systems utilized by the Norwegian military. One of the tools used to carry out this task is the open source Intrusion Detection System (IDS) SNORT® (1).

SNORT® is a leading open source IDS that is used to detect potentially unwanted or malicious network traffic based on predefined patterns described in signatures. There are both free and commercial signatures available for SNORT® and as SNORT® has been highly adopted by the security community, these signatures are continually updated and maintained.

Using SNORT® has been plagued for years by the fact that it does not have a good system for handling, updating or maintaining the signatures (henceforth called rules and rule sets) used for detection, thus making it cumbersome to utilize SNORT® for security analysts.

Research has shown that a good rule management system for SNORT® will improve the workflow and efficiency of the IDS significantly and a prototype for such a system was built in 2012 called Bring Home The Bacon (BHTB) (2). This prototype has been in use at Avdeling BKI since 2012, but is suffering from lacks in functionality and efficiency due to being a proof-of-concept prototype. The prototype did, however, prove that there is a significant gain from having an intuitive Graphical User Interface (GUI) by making the operator workflow much more efficient, as you go from having to do complex operations in a Command Line Interface (CLI) to just push buttons to make the system do things automagically (2).

There was another similar project conducted at Gjøvik University College (GUC) in 2012 called Snortmanager (3) that also deals with managing SNORT® rules. However, that project is not properly released as open source software, the project files are available but there is no documentation or instructions publically available that describes usage and installation. The project has also not been maintained since its initial release, is designed for a specific company in mind and is heavily designed around the user interface. Enabling, disabling and configuring rules is also cumbersome in this system as this is based on plaintext arguments in "policies" that users must construct themselves. The rule distribution is also done manually from the files generated. This makes Snortmanager not very usable for Avdeling BKI and their routines.

Avdeling BKI has therefore tasked a group of students from GUC to design and develop a more complete, effective and secure rule management system for SNORT® that can continue to live on beyond this project.

1.2. Project Objectives

Results

Research, design and develop a system that as a minimum;

- Is programmatically more efficient than BHTB.
- Is developed as a standalone engine that responds to API calls from any arbitrary user interface.
- Is modular in design and allows future changes and additions to not affect existing functionality.
- Is scalable enough to handle both small and large sets of IDS sensors and rules in a production environment.
- Is able to download rules and rule sets from multiple customizable sources.
- Is able to distribute different rules to different IDS sensors.
- Can download and distribute automatically at set intervals.
- Is capable of exporting rules for sharing.
- Has the capability to activate, deactivate, threshold and suppress rules on a per sensor basis, in a way that is user friendly.
- Supports multi-user interaction.
- Can efficiently and safely work with different installations of SNORT® and various other third party tools, such as SNORBY.
- Is easy to install.
- Has an internal logging system for all activity.
- Is well documented and can be easily adopted by anyone as open source software.

In addition, the project will attempt to add the following functionality;

- A separate graphical user interface.

- Said user interface must be user friendly and intuitive, and require only limited knowledge of SNORT® rules from the user.
- The system can also distribute rules to Suricata, another SNORT®-like IDS, which can operate with the exact same rules as SNORT®.
- Implement security features that ensure confidentiality of rules.
- The project will investigate and potentially implement multi-threading technology to exploit modern hardware capacities.
- The system is capable of validating and checking rules for errors before they are distributed to production sensors.
- The system allows commenting of the rules.
- The system supports writing custom rules and editing existing rules.

Once the system has been developed, it will be released as an Open Source Software to the public in a well-documented format.

Effects

Forming baselines from the current prototype BHTB and from Snortmanager, the project has a goal to further increase the efficiency of rule management by investigating and implementing programming techniques such as multi-threading and a change-based system to reduce latency and delays in the system by as much as 50% compared to the other projects. The project will therefore be measuring for this effect.

Furthermore, the project will research and implement safeguards against unintentional failures in SNORT® as a result of user error and malformed or misconfigured rules, thus reducing the potential downtime of the IDS sensors, as a SNORT® sensor will simply not restart if a single rule is malformed or misconfigured. Efficient ways of solving this problem will have to be researched.

1.3. Constraints

Time

The project will commence on Febuary 3rd 2014 and will end on May 19th 2014, elapsing 15 weeks. It will comprise of an initial planning and design phase lasting 2 weeks, followed by a development period of 8 weeks, and then ending with production of a report for the last 5 weeks. During the project, to ensure health and motivation, no work shall be conducted on Sundays.

Personnel

The project group consists of three students. Additionally, the group disposes one mentor from GUC and one mentor/representative from Avdeling BKI.

Technology

During development and testing, the project will utilize a virtualized server to host a testbed of multiple sensors.

Economy

Economic constrains are established in the Project Agreement. There is no budget for this project and any costs will be compensated as per the Project Agreement on an ad hoc basis.

Legal

The project group may during development see a need for testing the system and SNORT® in a setting with live network traffic. This must be done in a controlled fashion and consider the need for confidentiality in personal data.

The project will be influenced by the two previous projects surrounding the same type of software and must be wary of copyright infringements. If anything is reused, credit must be given where credit is due.

SNORT® is also a registered trademark of Sourcefire, Inc, and the project must treat it as such. (4)

2. SCOPE

2.1. Subject

Software Engineering, Information Security.

2.2. Limitations

As SNORT® is primarily used in conjunction with the Linux operating system, this project will limit its development of software to this operating system only. More specifically the project will contain its development to the Ubuntu distribution as this is the platform utilized by Avdeling BKI. The project will, however, strive to make the software portable.

The project will also have its main focus on the systems functionality and efficiency from a programmatically standpoint and developing a graphical user interface (GUI) is considered a secondary task and will prioritized as such. A GUI for demonstration purposes will be included in the project and it will primarily take inspiration from BHTB.

While the system itself will not handle any data that may comprise of personal data, it may contain rules and rule sets that are considered classified or confidential. The system must therefore be built with this in mind and ensure confidentiality within the boundaries of the system. The project will not guarantee the confidentiality of the system outside these boundaries.

These boundaries include the program and its processes and threads, and the network link established between the system and remote sensors. The project assumes that the system owner has already established database security.

2.3. Project description

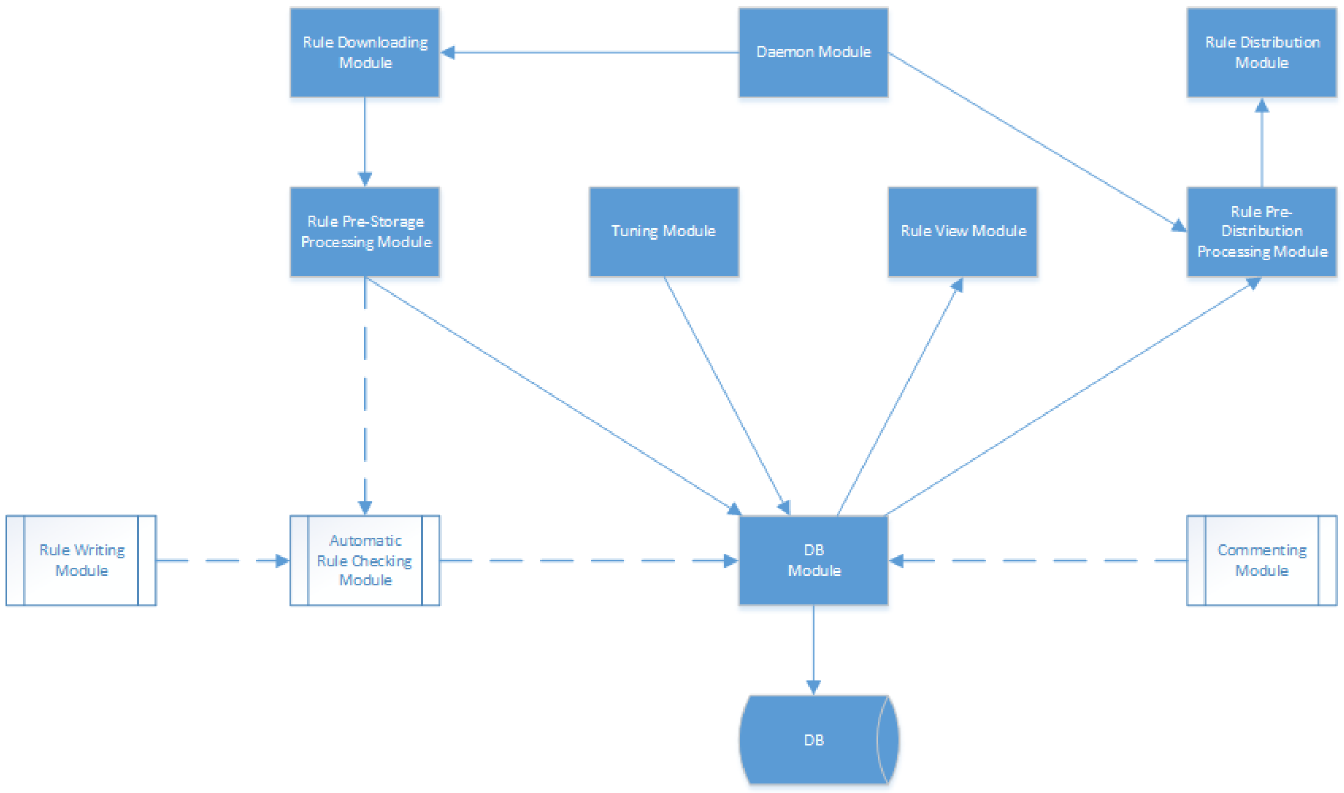
In broad strokes, this project is about designing a system that is a highly modular API-based engine, which is independent of any GUI. The project will be the spiritual successor to BHTB, but also take some inspiration from Snortmanager. Essentially taking good ideas from both projects, add a few of our own, and create a more adoptable open source rule management system for SNORT®.

We will utilize the latest stable version of Python as the main programming language for the project. C++ was also considered, but due to the limited time at our disposal, we feel we can create more functionality with Python, due to it requiring less memory management and general “programmer friendly” ways of doing things.

The system will require a database to function efficiently and the project will utilize MariaDB, as this is the de facto standard for the Linux platform. MariaDB is a separate branch developed from MySQL by the original developer of MySQL, which is now owned and maintained by Oracle.

Furthermore, the project will also consist of creating a GUI that is user friendly and responsive. We will base this heavily on the GUI developed for BHTB in terms of displaying rules in lists and with buttons and checkboxes to manipulate them. We may also borrow some design ideas from Snorby (5), another SNORT® tool. The main technologies we will utilize for the GUI will consist of standard web-oriented languages such as HTML, CSS, PHP and/or Javascript.

Project modules



The rule “engine” can essentially be boiled down into 11 modules, which we will base our development around. The blue modules must be in place to consider the system “functionally complete”.

Rule Downloading Module

This module will handle all functionality related to downloading rules.

Rule Pre-Storage Processing Module

This module will essentially take downloaded rules and process them so that they are ready to be fed into the database.

DB Module

This module will handle all input and output to/from the database.

Rule Pre-Distribution Processing Module

This module will essentially take rules stored in the database and prepare them for distribution.

Rule Distribution Module

This module will be in charge of distributing rules out to the SNORT® sensors.

Tuning Module

This module will handle all functionality related to turning rules on or off, thresholding them or suppressing them.

Rule View Module

This module will be similar to the Rule Pre-Distribution Processing Module, but instead of preparing the rules for distribution to SNORT® sensors, it will prepare them for a human readable format or GUI format.

Daemon Module

This module will run continually on the system to ensure that the API is responsive and to carry out any interval based tasks, such as automatic updates or distributions.

Commenting Module

This module will allow users to attach comments to rules.

Rule Writing Module

This module will handle functionality for writing custom rules and editing existing rules.

Automatic Rule Checking Module

This module will be responsible for verifying rules to ensure that there are no malformed or misconfigured rules that may potentially break a SNORT® sensor.

3. PROJECT ORGANIZATION

3.1. Roles

Project Manager

Thomas Nyheim

Development Team

Thomas Nyheim

Eirik Skogstad

Eigil Obrestad

Project Mentor

Slobodan Petrovic

Client Representative

Jarle Kittilsen

3.2. Regular meetings

The project will conduct regular status meetings every Friday. These meetings will be alternating every other week between who is participating among the project mentor and the client representative. Meaning the group will meet with the project mentor every two weeks and the client representative every two weeks.

These meetings will be timed so that the meeting with the project mentor happens in the middle of a development cycle and the meeting with the client representative will take place at the end of a development cycle. This ensures that the project group can get feedback from the mentor during the development and get feedback from the client on our progress and direction.

3.3. Rules

The rules exist to ensure an effective execution of the project and to ease the handling and resolution of potential conflict situations.

1. Group leader. The project group shall have one group leader who is elected for the entire duration of the project. The group leader is responsible for hosting weekly group meetings and keeping an overview of the project's work progress.

2. Work. Each group member is obliged to complete their work as best they can in accordance to what is agreed upon in the work-planning meetings. It is expected that each group member devotes a significant amount of time to the project and that the work load is, as far as possible, equally divided between all group members. Lack of participation from a group member will be discussed with the project mentor and repeated deliberate failures to produce the expected amount of work might ultimately lead to an exclusion from the group. Also, no group member is permitted to perform any work related to the project during Sundays unless this is approved by all group members.

3. Conflicts. Any disagreement regarding a decision shall be resolved by voting by the majority rule. Should the majority for some reason be unclear, the project leader will have the final say. If the decision in question has an important consequence for the end-product of the project, the project mentor or client representative should normally be consulted before making a decision.

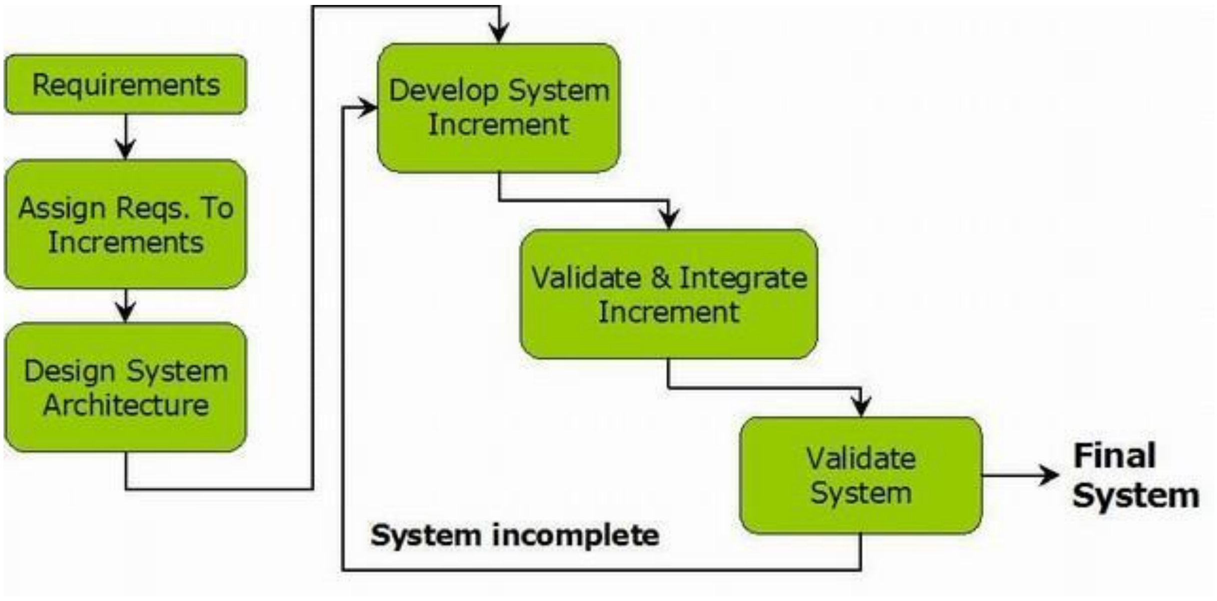
4. Expenses. Any expenses borne by a group member in direct relation to the project shall be reimbursed fully or partially by the other group members if this is decided by the majority of the project group. If the car of a group member is utilized for project-related transportation, the fuel cost shall be equally split between participating group members.

5. Absence. Group members are obliged to report to the other members any previsioned absence that directly interferes with the project work. Absence from the project lasting one consecutive week or more, or multiple unauthorized absence periods, is not permitted as this can severely endanger the work progress.

4. DEVELOPMENT MODEL

As this project is time-restricted, the development should be as efficient as possible. We therefore need a model that utilizes the manpower we have present in the best possible way. The size of the group is also quite small, so big and complex development models would add too much bureaucracy to coordinate the project and is thus overkill.

As we want to have a lightweight development-model, some form of an agile development model with well-defined development-cycles is preferred. This way, each group-member can take an active part in the project and all members would feel a sense of ownership.


Requirements
Assign Reqs. To Increments
Design System Architecture
Develop System Increment
Validate & Integrate Increment
Validate System
Final System
System incomplete

Developing iterative and incrementally, we are getting the best from both the unstructured world, and from the very well defined waterfall approach. We are able to get feedback during the project, and have a structured and reasonable way of actually doing changes in the plan while the project is running. At the same time we would be able to maintain the overall plan within cycles, so that it is possible to work individually, while still being well coordinated.

To be able to develop all the modules of the project within the timeframe, each development cycle will consist of multiple modules developed in parallel. This allows us to maximize our utilization of time and personnel.

The start of each cycle will consist of a meeting where we define the goals for that cycle, and identification of the individual tasks that we need to get done. This is similar to the start of a sprint in the development model Scrum.

In the development part of the cycle, we will individually work on the tasks that were defined in the planning-part. Regular meetings and cooperative programming within the group would ensure that we all pull the project in the same direction.

At the end of each cycle we should have a working set of software, which ideally should comply with all the goals set in the start of each cycle. We are planning to have a meeting with Avdeling BKI between each cycle to present the progress, and to let them influence the direction of the project.

After all development cycles have been complete, we will then make final assurances that everything works as intended and that all modules coexist in harmony. This part of the development will also consist of hands-on testing by the client and various performance testing.

5. QUALITY ASSURANCE

5.1. Documentation

The source code will be documented as per the standards for Python programming. The project will also use a wiki-based documentation that will explain all functionality and API calls in a user friendly way. This wiki will be used as the project website.

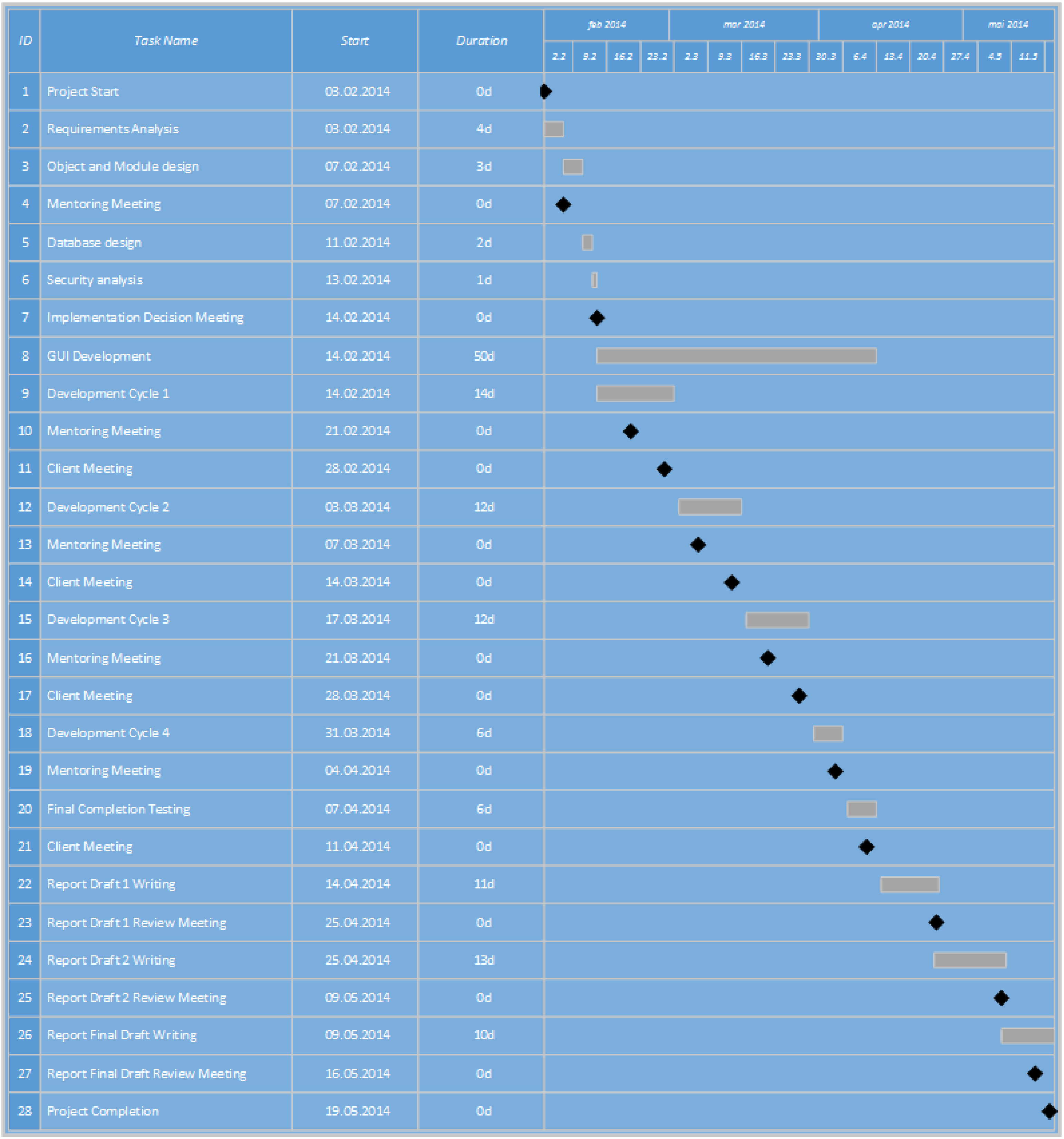
5.2. Configuration control

The project will utilize Subversion in a standard directory structure to ensure configuration control. The software will be versioned from 0.1 to 1.0.

5.3. Risk analysis

Threat	Likelihood	Consequence	Mitigation strategy
Project fails to complete in time	5	8	Ensure that proper planning is conducted before project start and to not graps over too much in the allotted time.
Project files or data is lost	2	10	Operate with backups routines and version control.
Sabotage of the project from an external threat agent in the form of either damage or planting of malware/backdoors	1	9	Ensure configuration control and internal project security at all times.
Project fails to meet customer expectations	3	8	Facilitate constant feedback and discussions with client to assure that the project is heading in the right directions at every step.
Loss of, or lack of participation from, project personell	3	8	Ensure availability and motivation of project personell before project start.
Illness among project personell	4	6	Maintain regulatory health and safety standards. Ensure time for breaks, food consumption and health-inducing actitivies.

6. PROJECT TIMELINE



ID	Task Name	Start	Duration
1	Project Start	03.02.2014	0d
2	Requirements Analysis	03.02.2014	4d
3	Object and Module design	07.02.2014	3d
4	Mentoring Meeting	07.02.2014	0d
5	Database design	11.02.2014	2d
6	Security analysis	13.02.2014	1d
7	Implementation Decision Meeting	14.02.2014	0d
8	GUI Development	14.02.2014	50d
9	Development Cycle 1	14.02.2014	14d
10	Mentoring Meeting	21.02.2014	0d
11	Client Meeting	28.02.2014	0d
12	Development Cycle 2	03.03.2014	12d
13	Mentoring Meeting	07.03.2014	0d
14	Client Meeting	14.03.2014	0d
15	Development Cycle 3	17.03.2014	12d
16	Mentoring Meeting	21.03.2014	0d
17	Client Meeting	28.03.2014	0d
18	Development Cycle 4	31.03.2014	6d
19	Mentoring Meeting	04.04.2014	0d
20	Final Completion Testing	07.04.2014	6d
21	Client Meeting	11.04.2014	0d
22	Report Draft 1 Writing	14.04.2014	11d
23	Report Draft 1 Review Meeting	25.04.2014	0d
24	Report Draft 2 Writing	25.04.2014	13d
25	Report Draft 2 Review Meeting	09.05.2014	0d
26	Report Final Draft Writing	09.05.2014	10d
27	Report Final Draft Review Meeting	16.05.2014	0d
28	Project Completion	19.05.2014	0d

APPENDIX A - CITATIONS

1. SourceFire Inc. About SNORT. [Online]. Available from: http://www.snort.org/snort.

2. Henriksen DO. Managing signatures for IDS in a distributed environment - A study of a signature management system. Gjøvik University College; 2012.

3. Vaskinn C, Wikestad K. Snortmanager. Gjøvik University College; 2012.

4. SourceFire Inc. Snort Trademark Guidelines. [Online]. [cited 2014 January. Available from: http://www.snort.org/legal/snort-licensing/snort-trademark-guidelines.

5. threat stack, inc. Snorby Website. [Online]. Available from: https://snorby.org/.