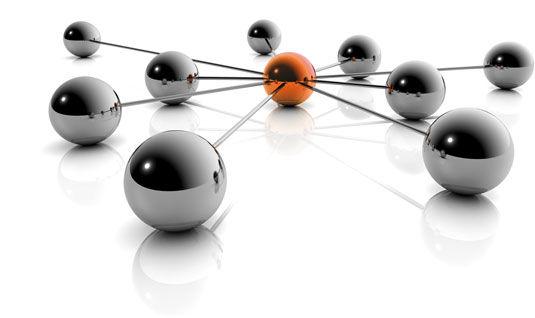
|  |  |
| --- | --- |
|  | Documentation |
|  | Class: 5AHITT  Last modification: 9/22/2014  Document Version: 1.0 |

Software engineering

2014/2015

Bergler, Bobek, Janeczek, Mair, Özsoy

Rock the net



List of contents

[Statement of task 1](#_Toc399184258)

[Trained competencies 1](#_Toc399184259)

[Basic tasks 1](#_Toc399184260)

[Additional information 1](#_Toc399184261)

[Advanced tasks (obligatory for grades better than C) 2](#_Toc399184262)

[Teams 2](#_Toc399184263)

[Grading 2](#_Toc399184264)

[Submission 2](#_Toc399184265)

[Interviews 3](#_Toc399184266)

[Apportionment of work with effort estimation 4](#_Toc399184267)

[Final time apportionment 5](#_Toc399184268)

[Design consideration 6](#_Toc399184269)

[Technology description 7](#_Toc399184270)

[Task execution 8](#_Toc399184271)

[Test report 9](#_Toc399184272)

[Bibliography 10](#_Toc399184273)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Version* | *Author/s* | *Date* | *Status* | *Comment* |
| 1.0 | Özsoy | 9/22/2014 | draft | Created the first version of the documentation |
| 1.1 |  |  |  |  |

# Statement of task

## Trained competencies

* Using APIs, Network programming
* Application programming: GUI-programming, parallel programming
* software engineering: buildsystems, testing with mock-objects, design patterns

## Basic tasks

Implement a simple-to-use application to monitor and configure a hardware firewall appliance “Juniper NetScreen 5GT “. The firewall allows read access over the SNMP-protocol (your app should be able to test if SNMPv3 is available and if not fallback on SNMPv2c) and write access over Telnet.

Your app should accomplish following tasks:

* List all configured firewall rules (policies) on the device, add the details of the mentioned services and zones as well.
* Allow refreshing of the list by clicking a button and by a configurable time-intervall. Your GUI should remain responsive even with short refresh-intervals!
* Visualize the thru-put for a highlighted firewall-rule (nice2have: multiple rows) in a line-chart (configurable refresh-interval, unit bytes/sec)
* Encapsulate the data retrieval for further reuse and easy expansion. An UML-model of your design will help you defend it at the review!
* Build a visual appealing and easy to use interface (there is more than Swing out there).

### Additional information

* Since there is only one firewall-appliance available, the time each team can test with the hardware will be strictly limited. Therefore it is essentially to use mock-objects to allow testing the app during times where the hardware is not available.
* An additional benefit of using mock-objects will be, that a CI-Server can use them for automated building and testing.
* You only need to consider firewall-rules for TCP and UDP connections in IPv4.
* You can find Information about the SNMP-Mibs special for the manufacturer of the used appliance here (maybe not all of the Mibs work with the used model):   
  <http://www.oidview.com/mibs/3224/md-3224-1.html>
* For exploring the SNMP-Data coming from the appliance you can use tools like this:  
  <http://ireasoning.com/mibbrowser.shtml>

## Advanced tasks (obligatory for grades better than C)

Additionally to the basic tasks your app should accomplish the following:

* Alarm the user visually and per email if the config of the firewall-rules changes. To avoid polling use the SNMP-trap mechanism.
* Allow managing of firewall-rules (CRUD). To accomplish this, you will have to send configuration commands via telnet or ssh. An admin-account is available per request.
* Use multicast-groups to build a simple transaction system to serialize administrative tasks on the firewall (for example pass an “admin token” to recognize the collaborator who is allowed to write to the firewall). This should also work in a heterogenous environment (different implementations, different OSes), so you have to coordinate with other teams.
* Make sure, that your interface to the firewall allows an easy change of the firewall-model (new releases, manufacturer, ...). It is not necessary to make this configurable in the GUI but must (explicitly) be considered in your software-design!

## Teams

Build teams with 3 to 5 participants (5 only if two or more members choose advanced level and at least one member chooses basic level). Each individual team-member has to implement, test and document code and is allowed to choose the level of difficulty he/she wants to achieve. For example: if you have a group of four students and two of them want to achieve advanced level, they can focus their implementation work on the advanced tasks. The other two team-members focus on the basic functionality. In any case there must be a working product, advanced tasks cannot stand for themselves.

## Grading

A team can apply for submission with a (mostly) functional product.

Each team-member will be graded separately, based on the documentation (and git-logs) which name him/her as author in all three main competencies as listed.

Advanced tasks will only be considered if the basic tasks are fulfilled for the most part in this team.

### Submission

* Every group must have its own design/solution! Meta-group solutions will end in massive loss of points!
* As for group work usual, a protocol with the UML-Design, the work-sharing, the timetable and test documentation is mandatory!
* Upload your solution as a ZIP file. Please submit only the sources of your solution and a build file (build.xml, pom.xml, Makefile etc.) not the compiled class files and only approved third-party libraries. Your submission must compile and run!
* Before the submission deadline, you can upload your solution as often as you like. Note that any existing submission will be replaced by uploading a new one.

### Interviews

* During the implementation there will be review interviews with the teams. Please be aware that the continous implementation will be overseen and evaluated!
* After the submission deadline, there will be a mandatory interview.
* The interview will take place in the lesson. During the interview, every group member will be asked about the solution that everyone has uploaded (i.e., changes after the deadline will not be taken into account! There will be only extrapoints for nice and stable solutions!). In the interview you need to explain the code, design and architecture in detail.

# Apportionment of work with effort estimation

|  |  |  |  |
| --- | --- | --- | --- |
| Competent  person(s) | Task | Description | Estimated time in h |
| Özsoy | Creating the first version of the documentation | -- | 1 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Estimated total time exposure**

|  |  |
| --- | --- |
| Person | Time exposure in h |
| Bergler |  |
| Bobek |  |
| Janeczek |  |
| Mair |  |
| Özsoy | 1 |
| **Sum:** | **1** |

# Final time apportionment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Competent  person(s) | Task | Estimated time in h | Actual time in h | Comment |
| Özsoy | Creating the first version of the documentation | 1 | 1,5 | - |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Actual total time exposure**

|  |  |
| --- | --- |
| Person | Time exposure in h |
| Bergler |  |
| Bobek |  |
| Janeczek |  |
| Mair |  |
| Özsoy | 1,5 |
| **Sum:** | **1,5** |

# Design consideration

# Technology description

# Task execution

# Test report

# Bibliography

|  |  |
| --- | --- |
| [1] | Title:  Author:  Online-/Resource:  last modified:  abstracted: |
| [2] | Title:  Author:  Online-/Resource:  last modified:  abstracted: |