*Disclaimer: My focus until June is mainly on my courses. I am still very excited about the thesis, but I have given myself until January for this, while I want to finish my courses for this semester (hopefully) in June. This would mean no retake exams in August and that my focus from July onwards is fully towards the thesis. Due to the amount of deadlines and tasks for the courses my planning might not seem very ambitious until they are finished.*

# PLANNING

|  |  |
| --- | --- |
| Monday 17/05 | Creating my detailed planning and improving my research problem |
| Monday 01/05 | Running the event monitor on my cluster, being able to generate the Kano matrix on my cluster and seeing the Kano matrix reflect changes I manually make. |
| Monday 15/05 | Piping the events to the algorithm and generating the Delta Kano matrix. |
| Monday 29/05 | Getting the information from the VMs and generating the required VM matrices. |
| Monday 12/06 | Using the newly generated VM matrices and using them in the algorithm |
| Monday 26/06 | Testing, finalizing and updating the algorithm to make sure everything is in order |
| Monday 10/07 | No new activities. This gives some extra space due to vacation and/or to catch up with missed previous activities due to deadlines, exams or unforeseen circumstances. The algorithm should not have to be touched after this phase. |
| Monday 24/07 | Writing Chapter 1: Introduction and Problem statement. This does not yet include the abstract, |
| Monday 07/08 | Writing Chapter 2: More in-depth about the problem |
| Monday 21/08 | Writing Chapter 3: Describing the solution/algorithm |
| Monday 04/09 | Setting up testing benchmarks, gathering results and trying to derive conclusions |
| Monday 18/09 | Writing chapter 4: The testing setup, results and conclusions |
| Monday 02/10 | Writing chapter 5 and abstract: Final chapter rounding everything up including possibly open research questions. |
| Monday 16/10 | TO BE FILLED/REVISION/EXTENSION |
| Monday 30/10 | TO BE FILLED/REVISION/EXTENSION |
| Monday 13/11 | TO BE FILLED/REVISION/EXTENSION |
| Monday 27/11 | TO BE FILLED/REVISION/EXTENSION |
| Monday 11/12 | TO BE FILLED/REVISION/EXTENSION |
| Monday 25/12 | TO BE FILLED/REVISION/EXTENSION |
| Monday 08/01 | TO BE FILLED/REVISION/EXTENSION |
| Monday 22/01 | TO BE FILLED/REVISION/EXTENSION |

# 

# Research Problem:

Verifying the consistency of Kubernetes security policies between security layers

*Disclaimer: technical background information is left behind and will be more in-depth in the final thesis. This is just a short problem statement and thesis goals.*

Within Kubernetes we might have multiple nodes (Cluster layer) running on different Virtual Machines (VMs) (Cloud layer). These VMs might have rules defined between them in terms of security policies that indicate where communication is allowed or disallowed. The problem arises if we take into account the security policies defined within the Cluster layer of the Kubernetes stack. Here more specific rules might be specified that define whether or not containers and/or pods are allowed to communicate. Seeing how pods can be deployed on different nodes within the cluster, which in turn might be deployed on different VMs within the cloud layer it is not hard to see how we might have conflicts between security policies.  
  
Let me give an example. Imagine a Pod *Pa* being deployed on Node *Na*, and a Pod *Pb* being deployed on Node *Nb*. *Pa* and *Pb* have their security policies set up in such a way that they should be able to communicate. This is necessary because *Pa*pulls its data from *Pb*. However, nodes *Na* and *Nb* are deployed on different VM’s at the cloud provider: VM *Va* and *Vb* respectively. The cloud provider has set up rules that all of it’s VMs are not allowed to communicate with each other in any way, meaning that the pods deployed on them also can not communicate.

Clearly these different security rules are conflicting with each other, but are never checked. With the constant possibility of nodes, pods and containers being deleted, moved and added this needs to be checked regularly as well. Due to the enormous amount of these entities being deployed, each with possibly multiple security rules in place makes manually verifying these rules against each other unviable. So summarized: Verification needs to be done quickly, automatically and only taking into account the changes that might impact the security rules.

My thesis goal is to achieve this automatic verification. To do this I start with the Kano container matrix and by generating matrices that show (1) the relations between containers and VMs and (2) the communication relation between these VMs.   
Every event that changes one or multiple security rules must trigger an algorithm that only takes into account the affected existing rules. Conflicts, where (1) rules are not restricting enough, (2) rules are too restricting or (3) the newly added security rules creates a conflict must be outputted. Solving conflicts that I have detected fall outside the scope.