|  |  |
| --- | --- |
| Jimmy Holdö  890130-6319  gusholji@student.gu.se | Henrik Möller  800419-4810  d00hema@student.chalmers.se |

Lab 3 – Open project

Working with the MAC library for increasing the security of using untrusted code.

# Why we choose to work with the MAC library

Many times, in the work life and in other occurrences there exists a deadline for when a task should be done. Therefore, it is often that we reuse code that we have not written ourselves. This is a good approach to save time because we do not have to solve an already solved problem again but it can be hard to know if the code developed by someone else is secure.

There exist many different approaches to solve this problem but we found that the approach the MAC library uses to address this problem to be interesting. By forcing all code to use the MAC library all written code is guided to an implementation that make it possible to guard sensitive data passed to untrusted code from leaking to unintentional destinations.

# Related learning outcomes

It is possible that we will cover more of the learning outcomes that we have not thought about yet but we think the main ones we are going to cover is within the learning outcomes about types.

We are going to try to explain and show some examples how implemented type classes and types from the MAC library is used to securing sensitive data.

We are also going to touch the learning outcomes about monads. This point is perhaps obvious because the MAC library depends on the monad family MAC to encapsulate IO-actions in a secure manner for execution.

# Preliminary description of what we intend to do

We intend to do a tutorial of the library where we are going to pick out parts of the library and discussing the implications of these parts. Why they work and how they work. Where it is appropriate we are also going to implement small examples of a discussed part from the MAC library to show how it is possible to use that part.