Haaga-Helia
University of Applied Sciences

Diary Thesis: Security Solutions DevOps Intern

Marija Kostadinov

Bachelor's Thesis
Degree Programme in BITE
2022

Haaga-Helia
University of Applied Sciences

Abstract

19.05.2022

Author
Marija Kostadinov

Degree programme
Business Information Technology

Thesis title	**Number of pages**
Diary Thesis: Security Solutions DevOps Intern	72

The diary-form thesis demonstrates the author's ability to apply her knowledge and skills in a practical role as a DevOps Engineer Inter in a multinational telecommunications company. The work presented in the thesis has been following the author's progress, gained knowledge, as well as professional and personal development.

The daily entries in the thesis include eight weeks of observation, where for each week the work is described, and then the summary of insights and new findings is provided in the weekly analysis. Tasks during the observation period were mainly related to modifications and testing of Ansible scripts for security automation, and testing and verification of the company's product releases.

In terms of research approach, a diary-form thesis represents autoethnographic research. Various sources are used to analyse and improve the author's working methods on the thesis. Engaging in self-reflection has improved the way how an intern presents her own experience in a diverse team.

The development process can be categorized into three phases. The first phase was at the very beginning when it was more about learning new technologies and tools used in daily work. The second phase documents the work on the more complex tasks that required much more collaboration with others. The last phase refers to soft skills developed over time.

Keywords
DevOps, ESM, Policy Catalog, Ansible, YAML, SSH, Program Increment

Table of contents

Abbreviations

API	Application Programming Interface
CLI	Command Line Interface
CPI	Customer Product Information
CI / CD	Continuous Integration / Continuous Delivery
ESM	Ericsson Security Manager
ISO	International Standards Organization
LCM	Life Cycle Management
MOTD	Message of the day
NAT	Network address translation
NETCONF	Network Configuration
NIST	National Institute of Standards and Technology
PAM	Pluggable Authentication Modules
PC	Policy Catalog
PI	Program Increment
PO	Product Owner
SNMP	Simple Network Management Protocol
SSH	Secure Shell
TCP	Transmission Control Protocol
VM	Virtual Machine
VNF	Virtual network functions
XML	Extensible Markup Language
YAML	Yet Another Markup Language

1 Introduction

This is a Thesis Diary report in which I recorded my daily activities for eight weeks, where I am describing my work and experience as an intern at Ericsson. Since my internship started four months before reporting period, I will also have a short overview of the work done during that period. The starting date of the diary entry is 14.02.2022 and the scheduled end date is 08.04.2022. The content of the thesis is done by following guidelines for writing diary theses. Therefore, recured parts of the thesis in addition to the diary entries are also the introduction part where are the company and my working environment presented. This is followed by an explanation of the framework needed for the work duties and an analysis of my current situation at work.

The main reason for choosing a Diary thesis is the possibility to make better connections between work and studies which leads to better organization of time and my thoughts, a better understanding of the certain aspect of my daily work, and the ability to summarize gained knowledge.

The thesis follows my progress, knowledge, as well my professional development. My position in Ericsson is DevOps Engineer Intern, and my team which is responsible for Policy Catalog (PC) development is a part of the bigger team – The Security Solution Team. My tasks have mainly included modification, development, and test of Ansible security automation scripts, testing and verifying product releases, and fixing bugs in the code.

Since I am a beginner in this area of work, my tasks mostly require a lot of researching and learning on a daily level (code logic, principle of testing, and scripting). Most of the diary entries are not covering the workday in all detail. The information comes from internal pages and my peers, especially PO, but I use the different web forums, knowledge sharing sites, and official documentation for example Ansible and Red Hat.

The following text (1.2 – 1.4) provides information about ESM, Security Solution Team, and the main responsibilities within the team as well as PC descriptions. The provided information is gathered from Ericsson’s internal documentation - Wiki pages written by employees but also reflect my observations.

1.1 Ericsson

As one of the well-known name in the telecommunication world, Ericsson is a Swedish multinational company headquartered in Stockholm, founded 145 years ago. Ericsson’s

comprehensive portfolio ranges across Networks, Digital Services, and Managed Services powered by 5G and IoT platforms.

Ericsson is present in Finland since 1928. There are nearly 800 employees, and they are part of the R&D team, focusing mostly on product and network security, IoT, cloud, and 5G.

1.2 Ericsson Security Manager – ESM

By joining Ericsson, I have the opportunity to work on a product named Ericsson Security Manager (ESM). Ericsson Security Manager provides industry standards-based security, policy-driven security automation, and compliance monitoring. It helps to automate the enforcement of consistent security controls and maintains them at the desired level during the operations. (Ericsson, n.d.)

ESM provides the following solutions for customers:

- One unified, end-to-end view for security automation and identity management.
- Ericsson PC aligned with industry-standard security frameworks including pre-defined policy families, policies, and controls.
- Automated execution of security policies i.e., configuration enforcement for security controls and functions per defined policy sets.
- Automated compliance monitoring for the enforced policy sets.
- Rule-based analytics for known threats based on logs and events collected from the network.
- Vulnerability management through Ericsson PSIRT (Product Security Incident Response Team) vulnerability feed and support for external vulnerability scanners
- machine-learning-based analytics capabilities detect network anomalies and unknown threats for different use cases.
- Support for Ericsson Threat Intelligence services sharing indicators of compromise with other parties.
- Support for workflows providing support for a different types of action flows based on the analytics results that may require security expert involvement or approval e.g., to instantiate temporary security controls.

1.3 Security Solution DevOps

Security Solutions DevOps organization is responsible for:

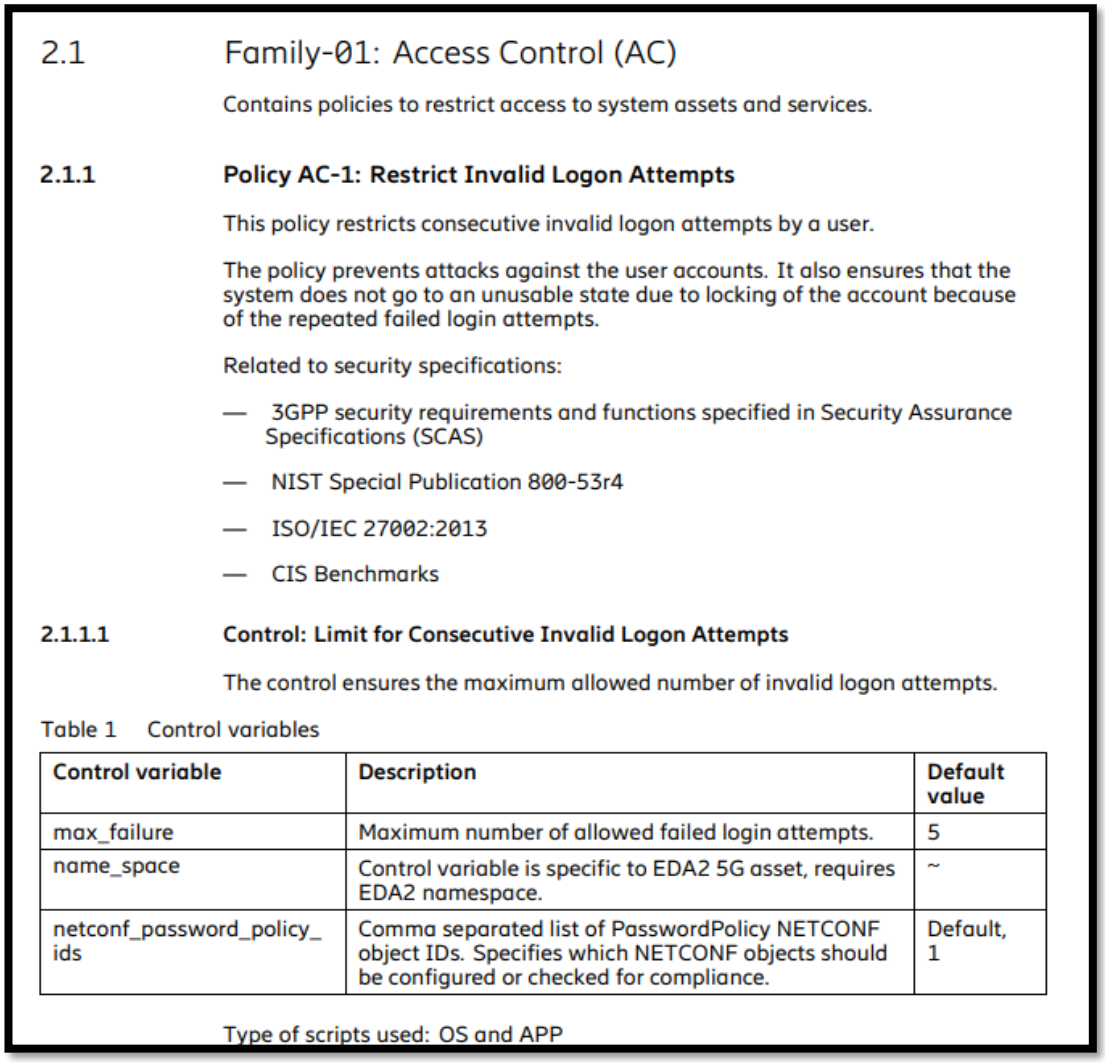
- Automation of telecommunication network security policies and controls by analyzing, scripting, and verifying through test automation

- Analyze customer requirements, technologies, and networks, solving complex technical problems
- Perform deployments, troubleshooting, and incident handling
- Support development and services by scripting and providing technical expertise
- Drive continuous improvements in Product, Processes, and Documentation
- Log and track customer reported incidents and corresponding emergency patches
- Security Monitoring, Analysis, and Perform Mitigation

The responsibility of the Security Solutions organization is shared among many teams that are working together toward a common goal. Shared responsibility involves the commitment of a wide range of participants who combine different skills, experiences, and knowledge to successfully deliver a great product to the customer.

1.4 Policy Catalog

Policy Catalog (PC) development has been separated from the main ESM. PC is pre-selected security policy families, policies, and controls based on well-known security standards (ISO, NIST, etc) and security policy frameworks tailored to telecom deployments. An information security framework represents a set of guidelines and defined adopted policies, procedures, and processes that an organization can adopt to secure its infrastructure and assets (TBG Security, 2018). The most known security frameworks are the International Standards Organization (ISO) and the US National Institute of Standards and Technology (NIST).



2.1 Family-01: Access Control (AC)

Contains policies to restrict access to system assets and services.

2.1.1 Policy AC-1: Restrict Invalid Logon Attempts

This policy restricts consecutive invalid logon attempts by a user.

The policy prevents attacks against the user accounts. It also ensures that the system does not go to an unusable state due to locking of the account because of the repeated failed login attempts.

Related to security specifications:

— 3GPP security requirements and functions specified in Security Assurance Specifications (SCAS)

— NIST Special Publication 800-53r4

— ISO/IEC 27002:2013

— CIS Benchmarks

2.1.1.1 Control: Limit for Consecutive Invalid Logon Attempts

The control ensures the maximum allowed number of invalid logon attempts.

Table 1 Control variables

Control variable	Description	Default value
max_failure	Maximum number of allowed failed login attempts.	5
name_space	Control variable is specific to EDA2 5G asset, requires EDA2 namespace.	~
netconf_password_policy_ids	Comma separated list of PasswordPolicy NETCONF object IDs. Specifies which NETCONF objects should be configured or checked for compliance.	Default, 1

Type of scripts used: OS and APP

Figure 1. Policy Catalog

The figure above (Figure 1) is a screenshot from a pdf document that contains a description of each policy, family, and control. Therefore, policies are organized into policy families. Each family contains security policies and controls related to the security function represented by the policy family. A control can have control variables that specify the control behavior. All controls are implemented using Ansible scripts providing a mechanism to automate the policy enforcement and compliance monitoring. As can be seen from the figure (Figure 1), Ericsson's Policy Catalog implements commonly accepted information security controls of ISO/IEC 27002:2013 and other internationally recognized standards as a reference for implementing and managing information security controls.

The development process of the PC includes the Ansible scripts development. The policy controls are implemented using Ansible scripts providing a mechanism to automate the policy enforcement and compliance monitoring. Every new script should be stored under the staging directory if it is not part of the PC by creating a subdirectory for the network node (asset). A study must be done to identify the applicability and possible scenarios for each

relevant Security Control in the existing ESM Policy Catalog per component to be implemented and monitored for compliance on the asset. For each policy control, one Ansible script is created.

1.4.1 Team Zeus

I am a member of the team Zeus, which is one of the teams responsible for PC Development. At this moment, my team consists of 7 members – (Figure 2).

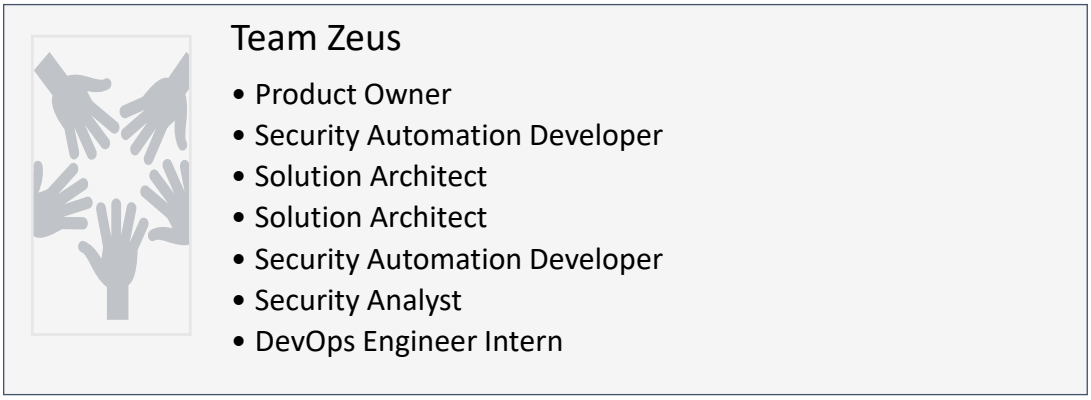


Figure 2. Team Zeus members

The main responsibilities of the team are scripting, asset studies, continuous improvements of product, processes, and documentation, and testing (manual, functional and automated testing).

1.5 Ways of Working

The whole ESM unit works in agile ways but there is no strict following of agile methods. From the agile framework the closest to the team is scrum, but there no scrum master.

Scrum is founded on empiricism and lean thinking. Empiricism asserts that knowledge comes from experience and making decisions based on what is observed. Lean thinking reduces waste and focuses on the essentials. Scrum employs an iterative, incremental approach to optimize predictability and control risk. Scrum engages groups of people who collectively have all the skills and expertise to do the work and share or acquire such skills as needed. (Schwaber & Sutherland, 2020)

So, we utilize agile methodology by planning the release cycle. A release is a set of working software that is made available to customers or for a production environment.

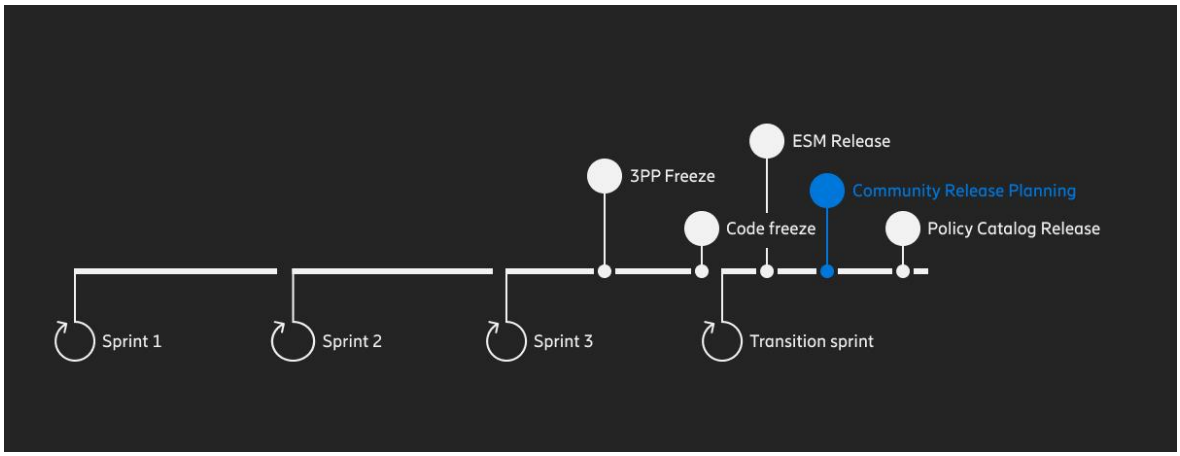


Figure 3. Eight weeks release cycle

Before each release cycle, which is eight weeks, the planning of the next release content already starts during the ongoing cycle. The release plan needs to have a list of requirements coming from customers, an estimation of the availability for each team member, and an evaluation of the execution of all required items. Planning is done in a way that the Product Line together with Product Owners defines a roadmap based on priorities from different stakeholders. Development items are added to the product backlog and pre-analysis for items is done.

As can be seen from the figure above (Figure 3) there are four sprints within the release cycle and one sprint lasts for two weeks, from Wednesday to Wednesday. Basically, three sprints include work done on topics that have been chosen for the specific release, and then there is one transition sprint for discussion on things that need to be improved, upgraded, and included in the next release cycle. Sprint planning is kept on each Wednesday when Sprint starts.

Each working team decides how often they want to have daily meetings. This can be once a day or a couple of times per week. In my team, Zeus, the daily stand-up meeting is three times per week, where each member of the team explains briefly what they are working on and if they have issues that would need others' help. In this way, all team members are informed of every member’s current status.

1.5.1 Jira

Jira is used as a collaboration tool to manage work. Jira is “a powerful work management tool for all kinds of use cases, from requirements and test case management to agile software development” (Atlassian, n.d. a).

All work can be planned and followed in Jira. But not long ago, based on my understanding it was decided by Ericsson management that Jira migrate to eTeamProject. The main features are the same and now eTeamProject space is mainly used for the following tasks:

- running daily activities such as creating and editing tasks, starting, and ending sprints, using the Kanban board(s)
- running reports
- keeping documentation (Wiki pages) for everything related to the project in one place, for example, different studies, instructions, and other documentation.

The team's Kanban board is managed by Jira. Here is displayed the most relevant information for each sprint. Issues assigned to each team member (Jira tickets) are present in columns that represent the process of work: what is needed to be done, work in progress, and the work that is completed. The rule is that a ticket should have its code, in that way, progress can be followed, and references can be reviewed at every moment.

2 Framework

2.1 Analysis of your current work

My internship started at the end of September 2021, and at the time of writing my thesis, I am still working as an intern. Applying for an internship at Ericsson was more based on what I can learn and achieve than on my experience in Security Domain and industry standards in Telecom Networks. I can say that participation in different projects, especially Business IT Project helped me to develop my agile working techniques which were crucial to getting work placement and at this point to support my work on the thesis.

The first month was the most challenging time for me. I had mostly spent time learning about the company working styles, about the product (ESM), gathering information about security management and security assurance, and preparing for my development environment. During the next month, I had the opportunity to get familiar with code, git repository, and ESM UI.

I can say that most of the time for me was a self-learning period. I learned the Ansible basics from Udemy and followed different suggestions given by team members about online guides for learning Linux commands and bash scripting. Knowledge about the command-line interface (CLI or shell) is needed for daily work and sometimes just to find information faster by using commands like **find** (for searching and locating files), **grep** (for locating a file that contains a particular string), and **sed** (can perform functions on file like searching, find and replace, insert, or delete). Although I have used Git before in some courses, now I understand why it is so important to use version control software, especially in a big team where everybody can work on any script at any time without warning about conflicts. I can say I'm becoming more confident in Git and using it on daily basis.

The first task was assigned to me when I felt ready to ask for one. It was a bug in the code and my task was to find a place in the code where the control variable was defined incorrectly. That was the first time I run a compliance check in ESM. The main goal was to get to know the code and to understand code organization inside the project. The main resources for finding information are Wiki pages on eTeamSpace and the help of co-workers.

So far, my tasks have mainly included modification, development, and test of Ansible security automation scripts, testing and verifying product releases, and fixing bugs in the code. Testing scripts are done locally or manually on ESM GUI (Graphical User Interface).

2.1.1 Setup Dev Environment on Windows

To be able to work on PC development, I needed first to set up the virtual workspace. Since I am using a Windows machine, and the development environment must be Linux I needed the option for running Linux software on Windows. One of the options was to install VirtualBox, download an ISO file for Ubuntu as one of the Linux distributions, and install Ubuntu inside the virtual machine. But after spending some time researching and based on the advice received from a colleague, I have decided to go with WSL (Windows Subsystem for Linux). Windows Subsystem for Linux (WSL) lets developers run a GNU/Linux environment including most command-line tools, utilities, and applications directly on Windows, unmodified, without the overhead of a traditional virtual machine (Microsoft).

All necessary steps are done by following instructions for installation on the official Microsoft site – WSL Documentation. The next step was to install the Remote -WSL extension on VS Code. So, to open a project from my WSL distribution in VS Code I need to open the command line, to make sure that I am in the right directory and just enter: `code .`

2.1.2 Git and Gerrit

Gerrit is a self-hosted software application that provides code review functionality. Gerrit functions as an intermediate between developers and git repositories since it is integrated with Git.

Gerrit is the ESM project's repository and code collaboration site. To clone the git repository for the first-time next steps were needed:

- Gerrit's account needs to be activated.
- Configuration of ssh identities (ssh public key upload to Gerrit).
- Allowed access to catalog repository.

How Gerrit works

On the top level of the PC git repository, there is a script that helps with installing the required local git hooks (post-commit and comment hooks) and doing other necessary git setups and git config. By default, it will set up docker to run the post-commit hooks. So, the script should run after the repository has been cloned.

To create a new Gerrit review Gerrit hooks need to be installed, commit changes locally with a proper commit message (git commit) and push code for review (git push). Each change in Gerrit is identified by a Change-Id, so in case there is a need to update a change

based on received feedback through the code review process, a Change-Id allows that by addressing the change in an amended commit, without worrying about conflicts. Then pushing again for review as before and it will create a new patch set to the already existing Gerrit review.

ESM uses the standard Gerrit workflow with a couple of added quirks.

- Verify +1 is only given by Jenkins if verification tests are executed successfully.
- Code-Review +2 is required before the patch set can be submitted to the remote master branch.
- Submit button will not be shown until the above is done.
- Ready-To-Submit +1 vote is required to trigger a Jenkins job that adds the review to the submit queue. The review will eventually get cherry-picked and tested, possibly in a batch with other reviews, and merged if the test is successful.

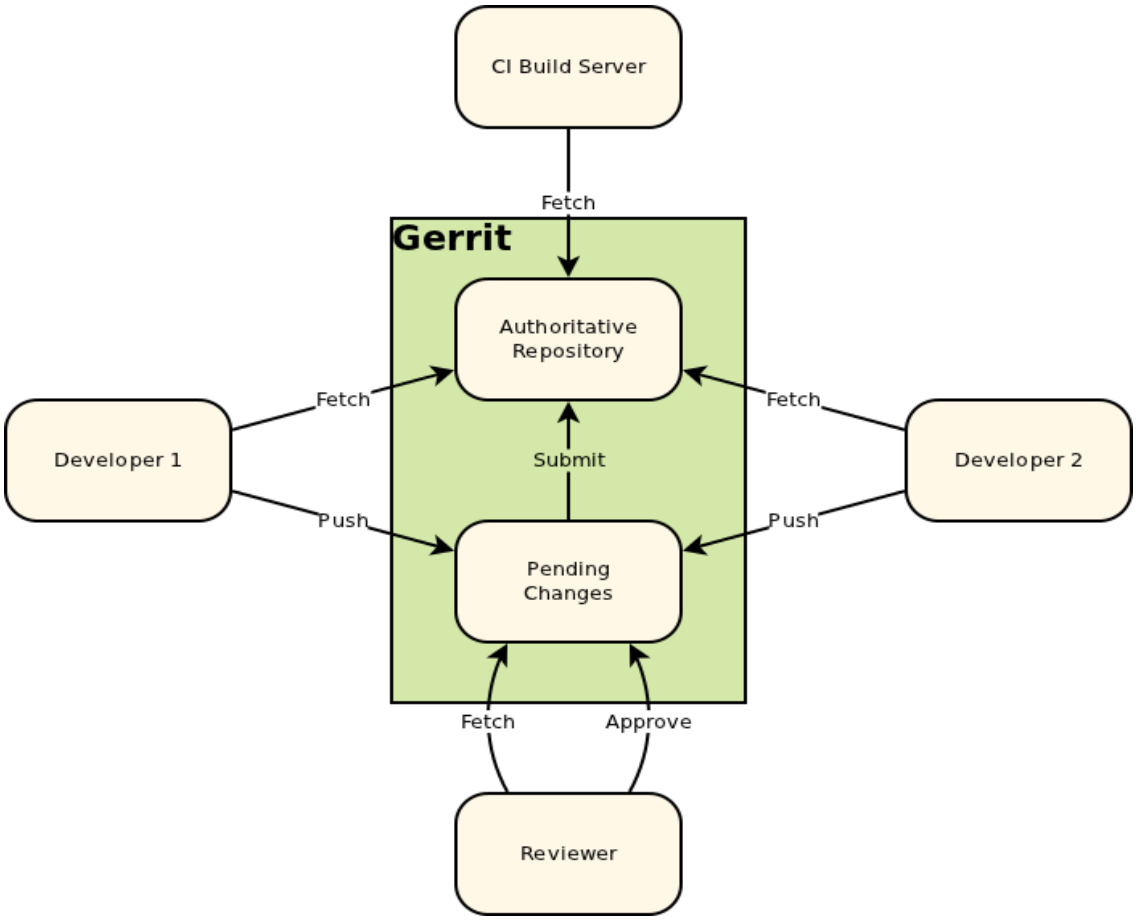


Figure 4. Gerrit as the Central Repository (gerrit-review.googlesource)

The figure above (Figure 4) depicts Gerrit as the central source repository. Each developer pushes code changes with Git to the Pending Changes for others to review and fetches

changes made by others which are sent to Central Repository. The code changes can be fetched to be tested.

PC Development has been separated from the main ESM git repository and CI workflow.

2.1.3 Asset

An asset is a network element, for example, a node (virtual machine or server) that must be checked for compliance with security policies. So, an asset can be a server, router, or telecom equipment in the network.

2.2 Evaluation

At the time I started the internship, I didn't know anything about DevOps practices and principles of DevOps because it is not being part of my studies directly. This is my first IT-related job. My experience was limited and based on completed courses in Haaga Helia. I can say that I had only experience with tools used on daily basis such as version control (Git) and Visual Studio Code.

At this point, all tasks are assigned to me by the Product Owner since he is aware of my situation and my level of knowledge. For me, learning is an ongoing process, and I need more time to connect pieces, especially a better understanding of the whole CI/CD Pipeline.

I mostly work alone, independently, and spend time learning the techniques and skills required to achieve the tasks. This suits me because I am still adopting and need more time to complete tasks. I am always trying to find information on my own before asking for help even if it can be time-consuming. I still can't estimate how much time is needed for each task, but I think I did well until now, even without having previous experience. I start to feel like a valid part of the team, I managed to learn basic terminologies and the workflow.

Based on my experience the most important qualities for the job are to be enthusiastic with a can-do attitude and the rest comes over time. So, lack of experience can be compensated by hard work and determination to get work done. What I like about this job is that I don't feel pressure from colleagues. With each new task, I learn new things and update my knowledge and technical skills. Can be challenging that the area where I am working has specific requirements and information needed for performing a task can be hard to get. It is important to know who to turn to for advice and guidance.

2.3 Development

I am exploring a new field of interest. I got the opportunity to learn new skills needed in this area. So, now I am at the beginning of my journey, and I am aware that many job-related things require my attention, especially scripting (Shell, Ansible). Also, one of the goals is to get familiar with the whole concept of the CI / CD pipeline.

The necessary technical skills still required daily upgrades, and I am sure that more work and experience will lead me to a satisfactory level for this job.

2.4 Interest groups at work

A stakeholder is a party that has an interest in a company and can either affect or be affected by the business. Stakeholders can be internal or external to an organization. External stakeholders are customers, vendors, and creditors, while internal are those who directly have a relationship with an organization, like employees, owners, and investors. (Fernando, 2021)

I am being part of one international company, and it is hard to determine a more detailed classification of stakeholders, therefore I will concentrate only on part of the company that is responsible for ESM product development. I have used mind mapping (Figure 5) to classify the interest groups.

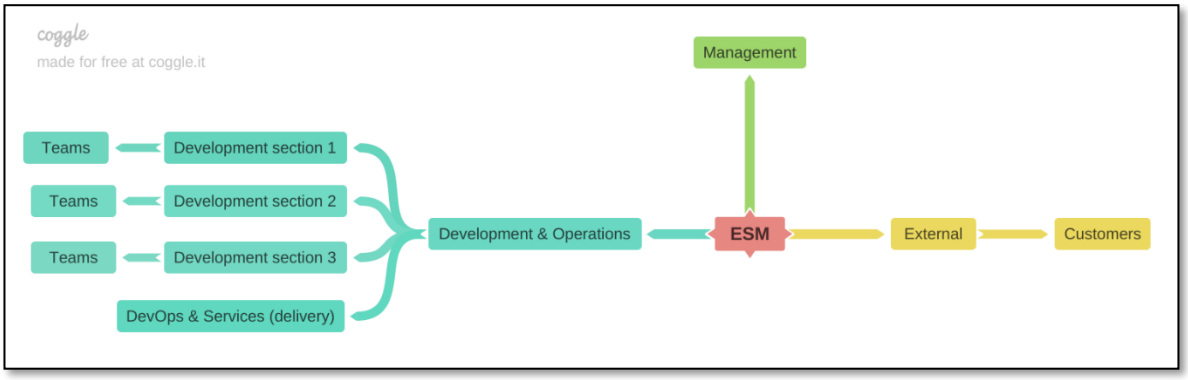


Figure 5. Interest groups

The biggest part is Development and Operations. As can be seen from the figure above (Figure 5), the ESM development has more development sections consisting of more than one team responsible for different security manager components. For example, a development team that is responsible for UX/frontend has various work areas, like user

research, UX design and prototyping, UI design, etc. PC section consists of several teams which are working on PC development and automation.

Management is the central element of each organization. Management activities are about working within the business to produce value for different stakeholders, deliver the product, and assist the business to sell the product. The management activities are spread across various departments, like the technology department, product management, research, and development, etc.

Customers are the most important company's external stakeholder, whose needs the company aims to satisfy by building and updating the current product.

2.5 Interaction skills at work

Due to the current situation with COVID-19, the work is organized to be remote. Therefore, the most important communication channel between development teams and inside of each team is Microsoft Teams. Various channels for communication are hosted under the common team "Security Solution" and access is allowed by a manager. Outlook is used for meeting invites, information sharing, and collaboration. All ESM developers can be reached by email. Ericsson uses various mailing lists for sharing information and other material with those who matter.

From the first day, my work was remote which causes challenges in interaction with others, within the team and outside of my team. Most conversations occurred through daily stand-up meetings; therefore, communication is within a small group with my peers. All interactions until now were task related. Sometimes I miss small talks since there is no "coffee break" where I can have a short, friendly conversation with co-workers, to get to know them better. My line manager is aware of this situation, and he arranges weekly team meetings as a time to end the week and reflects a bit – so time for everyone to share news, experiences, concerns, and not only work-related.

For all needed task-related things I contact the Product Owner first, who tries to be always available and willing to help, but in some cases, he can refer me to someone else.

At this point, I do not have any interactions with other teams or customers, and I am not involved in any discussions about specific internal requirements or ones coming from customers. However, in the future, I could be part of the cooperation between the teams.

3 Diary entries (14.02.2022 – 08.04.2022)

3.1 Observation week 1

Monday

The week started as usual with the daily stand-up meeting. Since last week was Program Increment (PI) planning week, and on Friday, all team members were introduced to items for the ongoing sprint, there wasn't a lot to discuss today.

My task is to transform Charging System node pam scripts to use include files. Currently, in the PC repo, there are a lot of scripts that have some parts with almost identical functionality. This code is hard to maintain as a minor change in functionality will require changes in a lot of files. Additionally, when a new script is added to the project, it will require some copy and paste code from other scripts, and this very often introduces some error (copy-paste error) as usual, some small part has to be changed, specialized for a new script. To solve this shared code functionality, Ansible supports as many other languages the including mechanism which allows one playbook to use other playbooks or tasks directly – there is direct relation from one playbook file to another. Module Include_tasks includes a file with a list of tasks to be executed in the current playbook.

First thing is to understand the task, what should be done, what documentation I should read to be able to fully understand the task, to understand the scripts I have to change, implement the task, and then test.

I will be working with pam scripts, so the first target for me today is to get basic information about pam files in Linux, their purpose, and how they are written. Implementation will be easier, as I have already done a similar task for the same asset but ssh config files.

Linux Pluggable Authentication Modules (PAM) configuration files are located under directory /etc/pam.d and they allow Linux system administrators to easily configure many things related to Linux account (user) authentication, for example, password expiration date, password checking, rules for password changing, action which system will do when a user is log-in and on log-out.

Tomorrow I will continue work by looking into existing code (scripts) and how we currently modifying pam files and validating them.

Tuesday

I will continue with the task today, and I think that the work on this ticket will take a whole week, if not longer.

This morning PO gave me guidance on where to see some examples in existing code as one asset, part of charging system nodes, has already scripts that are using shared files. So, at least have a good starting point. Shared functionality is already separated into new files under the folder ansible_shared_files. My job becoming clearer now. It will mostly come down to making a symbolic link to the ansible_shared_files folder in every asset folder, then replacing existing code in existing scripts to use shared files, and adding all necessary variables in modified scripts that are used in shared files that script will be run correctly. Then of course testing one by one modified script to prove they work correctly.

During the day I have identified two groups of the controls:

- Controls from Restrict Invalid Logon Attempts policy, such as Lockout Duration, No Dictionary Words in Password, etc.
- Controls from Password Management policy - controls that use pwquality (PAM module to perform password quality checking), such as New Password Differs from Previous Password, Password Complexity Strict, etc.

The structure of the playbook must be the same and functionality should stay the same. Shortly about how scripts are organized by order of functionality. All tests are done in three steps:

- pre-check
- configuration
- compliance

Pre-check is done by copying the file which needs to be checked from the pam folder to the temporary folder and then running checks against the file. If the file is not correct, it will some error messages and set some variables. Therefore, no changes will be made to the original file. The configuration will do all necessary changes to the original file if some error messages were generated in the pre-check. The compliance check will report errors generated in pre-check or configuration. Scripts can be run in configuration mode, compliance mode, or both.

So, I started with precheck files. I have noticed that some tasks in pam shared files use some variables that are not defined in the script, so the first step was to define those variables as local.

Later during the day, I attend a meeting that is more like an open forum that occurs only if there is some proposed topic for discussion. The purpose of the meeting is to share, solve and decide on community-wide PC Development related issues. Today's topic was Asset Lifecycle Management. For presentation, a Wiki page was prepared with a use case and the main questions were: How to deal with an updated version of assets from the PC point of view and what are steps required to make sure that PC supports the updated version.

Wednesday

This day started with a daily meeting. We had a brief discussion on the current progress of every team member.

Today I need to test modified scripts. I can do so-called local testing, making ssh tunnel from my machine to a real asset. Final testing is made by making a policy build, uploading this build into the existing ESM web application, and testing it again against the same assets.

Running ansible-playbook locally from CLI for local testing I have never done before. From my machine, I need to make ssh connection to the so-called jump host machine, and from there to make ssh connection to the final node (machine) under the test. The command for tunnelling was given to me by PO.

```
ssh -fNM <runo_username>@<runo_ip> -L <local_port_number>:<asset_ip>:22 -
S ./<tunnel_name>
```

More about the so-called "runo" machine. It is the jump host. It is an intermediary host to a remote network. We use a jump host to connect from laptop to inside of the company's internal network. The options (linuxCommand.org, 2020): -f Send SSH to the background, -N do not execute remote commands just do port forwarding, -M places the ssh client into "master" mode for connection sharing, -S ctl_path Specifies the location of a control socket for connection sharing, -L [bind_address:]port:host:hostport.

So, with this command, every TCP connection which is going towards local_port_number will be forwarded to the IP address of the asset on its port 22, which is ssh port, so ssh deamon (program running on that machine) will accept secured connection from our local machine. This is called tunneling.

Now I can run a script against localhost (my machine) on the port local_port_number from the previous command, and Ansible will execute them on the remote asset. But in order to do that I need the inventory file where I will define IP as localhost. Example of the Inventory file:

```
hostxxx ansible_port=22127 ansible_host=localhost ansible_user=esmuser
ansible_ssh_pass="PasswordForAsset" ansible_become_user="root"
ansible_become_password="PasswordForRootUser"
```

Here ansible_port has the same value as local_port_number from the command to make ssh tunnel. The tunnel must be closed once you don't need any more with the command:

```
ssh -S ./<path_to_the_tunnel> -O exit <runo_username>@<runo_ip>
```

I spent almost half of a day trying to figure out how ssh tunnel is working.

Thursday

The plan for today is to continue modifying the script.

I was able to modify a few precheck files, a few configuration files, and a few compliance files. Local testing in development time accelerates the process of development quite a lot and is much more comfortable to use. No need to build a full catalog, upload in ESM and run. There were two errors in running the configuration and compliance check. I had to understand deeply logic inside the files to be able to solve the errors. Full day left to solve it. The problem was in the usage of the ansible lineinfile module. This module will search for some text (given by regular expression) inside the file and will replace the text with the provided one. But here regular expression was made to search not from the beginning of the file, so it was replacing text which was commented.

Friday

All-day, except for two meetings, I was doing manual testing, and hopefully, there were no new errors compared to local testing.

Manual Testing is done with the ESM application using the Web UI. For testing PC changes, I need to build a catalog and test the changes in ESM itself. To build the catalog means that I need to create my catalog for testing purposes, and I am building it with the build command in the command line in Linux. The result is a zip file which then I import from ESM UI from

my local machine. All instructions are available on the Wiki page and the steps were easy to follow.

After uploading the new PC into the ESM, using the ESM menu, it is tested against the device under test (DUT), so assets that we have in the test laboratory. Those assets are different computers in the network. Their IP address is well known and can be found in the internal wiki, and then I need to start different policy checks against some or all the assets. The ideal results are that all our scripts are passing, which will mean all assets are good configured and there are no security issues. If some of them failed, I need to check the reason for the failing, to manually ssh to asset and to check why some check is failing, is it correct, so we need to do a new configuration of the asset, or there is some error in our script, so it needs to be changed.

Weekly analysis

The team is supportive but sometimes everyone has some urgent tasks and many things I have to resolve and learn alone. Internal documentation does not contain all details needed to finalize a task, therefore many times useful information has to be gotten by contacting some more senior developer.

The main outcome of this week's learning can be divided into general knowledge and project-specific knowledge.
General knowledge:

- Linux PAM files
- pwquality file
- SSH tunneling
- Ansible modules lineinfile, shell, copy, pamd and debug, and a few ansible keywords as set_fact, loop, when, register, check and debug mode.

Project-specific knowledge:

- local testing
- many ansible files are used inside a project for configuring and compliance checks related to pam files.

Linux PAM system is an acronym for Pluggable Authentication Modules. It is a group of libraries that allow the system administrator to do the configuration of the system for user authentication, and a flexible system for relatively easy configuration, because they are using configuration files, read from the disc. (Red Hat, n.d. a; InformIT, 2001; Weidner, 2004) It is divided into four independent modules:

- **account modules** check that the user (account) is valid at the moment the user wants to use some services. In this module, it is set time of expiration for user password, user right to access the service, etc.
- **authentication module** check that the password is correct when the user wants to access the service
- **password module** is responsible for updating passwords
- **session module** defines what will happen after the user is login into the current session and at the end of the session. The session starts after the user is successfully authenticated.

Every service in the Linux system which wants to use password protection has a file under the folder /etc/pam.d and that file have the same name as a service. This file will be used by PAM modules to do adequate action. For example, the *login* program defines its service name as *login* and installs the /etc/pam.d/login PAM configuration file.

pwquality.conf file is file in folder /etc/security. Its purpose is to give the way to configure the default password quality requirements for the system passwords. This configuration file will be read by the libpwquality library and utilities that use this library for checking and generating passwords. (Mraz, n.d.)

SSH tunnelling, or local forwarding, is a method of transferring data over an encrypted SSH connection using a simple method when any data sent to some local port on our machine will be automatically encrypted and send to some other machine port. This is useful for some organizations as they have one extra secured machine (hardened machine), called jump host, in the front of the secured network, so the connection to any resource will need to go via jump host.

Different Ansible modules I learned this week mainly using official ansible documentation (Ansible, n.d.):

shell – module for executing some shell command on the remote host. It can be any command on the remote host with many space-delimited parameters. One of the parameters for shell command can be executable which defines shell used to execute the command.
copy – module is useful when we want to copy some file from the local or remote machine to remote machine
pamd – module is useful for changing pam file on the remote host, has many parameters which usage is giving the possibility to change an exact parameter in remote pam file

debug – module is one which I was using the most and in many places. It can print any message, static or dynamic message with some parameters used in other modules, which helped me to find a few errors.

lineinfile – module ensures the line is in a file or replace an existing line using search given by regular expression. This is a powerful module for making files correct, like pam files. It does similar tasks like grep or sed or aws command on the local machine.

set_fact – action is used to set some variable to the current host (as it can be many hosts when we run a playbook, and this variable will be available during hole playbook. It can be *cacheable* so it can be visible even to another playbook in execution time.

loop – is a keyword that is used to execute some tasks in ansible multiple times.

when – is a keyword used to execute some tasks only when some condition is valid (true).

register – is a keyword to assign a result of some task to a variable, so latter we can do some other tasks depending on result which is stored in variable

check and **diff** mode – are used to run to just validate some tasks. Not all tasks support these modes of running. If they do, when the task is executed, nothing is changed on the remote machine, but the result of the task execution will be like things are changed on the remote host. Report of the task will say what would be changed. Diff mode gives detailed results before execution and after the execution of the command. These modes can be used separately or together in tasks where they are supported.

3.2 Observation week 2

This week’s objectives are not fully defined for now. The tasks that are assigned to me for this sprint have task dependency. However, PO told me that he will ping me during the week, maybe Monday already about a study that needs to be done.

Monday

As every Monday, the workday starts with a daily meeting. I informed my team about work that I had done last week and that I will push code in Gerrit for review.

I always follow my colleague's advice on using branches in Git. Over time, I understood a few advantages of branching the code. It allows me to separate work by task. So, when I have a few tasks in parallel, I can divide my work by tasks and can fast switch between them. Even when I do some experiments on the current branch, I make a branch on top of another working branch, test fast code, and go back to the previous branch. Therefore, my practice is to create a new branch for each task.

To create a new Gerrit review first is to add all changes which I want to the staging area (index) in git and then I commit changes locally with a proper commit message and then pushed changes to the remote (Gerrit) branch for review. Verify +1 is given by Jenkins since verification tests are executed successfully. Now I need to wait for code review, so other people can look into the code and make some suggestions or correct errors and do requested changes if any. This sometimes takes days, depending on how urgent changes need to be in the master and how much other colleagues have time. Review is done usually by senior developers, in my case PO or someone else from the team.

Later in the day, PO contacted me and gave me more information about the study I had to make. The main aspect of the study is to check if there are any configuration interfaces for the MINI-LINKs to use for PC integration. The source for finding relevant information is on CPI Store. CPI Store is a library explorer (ELEX) and contains only released product information – Customer Product Information (CPI). I have never done any study, but based on the experience of my peers, these studies are not something that is always going smoothly. Sometimes there is information, sometimes there is nothing.

My conclusion after doing some reading is that usually, MINI-LINKs are having a command-line interface (CLI). But some proprietary applications sometimes must be used, and they should be documented. I will continue work tomorrow with a clearer plan of what needs to be done and how to approach this task.

Tuesday

An approach is needed to conduct any study, including this one. Yesterday I was reviewing the documentation for half a day, and I noticed that there are a lot of sections in CPI Store for different interest parties. A lot of information is related to product technical description, installation, operation, and maintenance. Therefore, my plan for today is to find out what kind of information we need, and where to find them because browsing through CPI can be challenging. In the end, as a result, a Wiki page should be created where I will document my findings.

As a start, I have a list of MINI-LINKs for which I need to find relevant information. MINI-LINKs are a family of Ericsson products used in mobile telecommunication for transferring the data in the microwave spectrum, from 4 up to 80Ghz. We should investigate is it possible somehow to rich them via the internet and check their security configuration. If possible, then we can include them as an asset in our PC. Any of these assets has its CPI and relevant information for me are included in the Interface section (User Interface). Once I figure out what I need to do it was easy to find information for all assets from the list.

At the end of the day, I made a preliminary document in the form of a table. The table contains information about possible configuration tools. In addition, one column holds information with some relevant comments, which I considered important.

Wednesday

On the daily stand-up meeting, today as any other time each of us gave a short report on the work from the previous two days. I ask for a review of the first preliminary document about MINI-LINKs that I have made. I need feedback to make sure I am on the right track with this task.

During the day I got feedback from PO. Overall, it was ok, with only one comment that it will be useful to have information in the table about how we can connect to the asset by using the configuration tool, so which protocol is used. So, once again I was browsing through CPI and spent the rest of the working hours on that.

The goal for today was to complete the study, and I think I reached the planned goal. The study is present in form of a table. As an overview of my work, the table below (Table 1) can be used as an example.

Table 1. Preliminary study on the MINI-LINKs

Node Type	Supported configuration tools	Protocols	Comments	Source
MINI-LINK xxx	CLI CLI MINI-LINK tool MINI-LINK Node GUI	SSH:22 Telnet:23 – not recommended NETCONF:830	For scripting commands, CLI MINI-LINK tool must be installed (only Windows supported)	CPI Store

Usually, MINI-LINK are having CLI Interface. This means that once we are able to connect via some protocol (ssh, telnet, Netconf, SNMP) to MINI-LINK, commands we type on the command line in the terminal on our computer will be actually transferred to MINI-LINK and executed there, and the result will be returned to us and printed in the terminal. So, through the CLI users interact with a remote system. In this way, MINI-LINK can be configured, or just tested the current configuration from the security point of view. A list of configuration commands is provided in the documentation. For a secure connection to a remote asset, SSH is used (the default port is 22) and is always recommended. Also, there is a possibility to use Telnet but not recommended because is less secure.

Because many times commands can be written in script and script transferred to the device, and then executed there as one command, we are interested to know how scripting is working for any of MINI-LINK and does we need some additional (propriety or not) tool to make and transfer the scripts. It turns out that some of the scripts can be transferred and executed with secure file transfer protocol (SFTP), but other propriety applications for writing and executing scripts must be used.

There are also GUIs (Graphical User Interface) for each MINI-LINK but as a configuration interface, CLI is mostly used.

Thursday

There is no strict agenda for today. I am still waiting for code review, so keeping myself available in case there is a need to do the requested changes if any. Therefore, I decided to double-check the CPI documentation just to be sure that I have all the relevant information. Furthermore, looking into some scripts trying to deeper my code understanding. As the code base if big, with hundreds of files, it is good to sometimes use free time for this.

Last week the team was informed that there will be some changes in the team setup. In the afternoon, my team has a coffee break meeting with the new PO. It was an informal session to get to know each other more, not related to work.

Friday

On a daily, I was informed that tasks for this sprint had changed since there were some problems in a lab involving infrastructure. PO provide me with a link to a Wiki page where I can find information needed for the epic. Epic holds many Jira issues (tickets), and in that way, work is broken down from one large body of work into more manageable tasks. Some of those tasks will be assigned to me, and maybe someone else. Also, I received very good feedback for my study, and the ticket will be closed.

After a weekly meeting with the line manager and a few colleagues where we discuss casually week, I start looking into a problem that will be my task next week. The big picture is that applying policy to an asset is called asset configuration. In practice, it will mean that some configuration file on the target asset is changed. Nice to have as an option, and sometimes is mandatory to return the file to the beginning (start) stage, so before a configuration was done, for whatever reason the user has. This feature is called rollback. Rollback functionality is even implemented on some parts of the code. The main problem is that there are several scripts modifying the same file or an interface in one configuration run and all of them take a backup of the file and then do some changes, so the last running script is backing up an already modified file.

There is one asset where is rollback already fully implemented, and I started to investigate the code. It will require a few days to understand full functionality and work on this will be continued next week.

Weekly analysis

This week was mostly about doing the study for the first time and what is the best way to approach an unfamiliar task. This may seem like a relatively simple task, but reading this type of documentation is not easy, especially without a general idea. So, beginning from scratch.

I first did some studying about the asset myself to understand what kind of network elements MINI-LINKs are. As it is a hardware component, it was not needed to understand the details, just some big picture. This component is used in a mobile network for transferring data from

one mobile base station to another and finally to a telecommunication center. In the figure below (Figure 6), this transfer between base stations (black vertical lines) is marked with green lines. Simple, whenever a user is using a mobile phone, for example, data is sent to the nearest base station, and then using MINI-LINK the data are sent further until they come to the telecom center (BSC), which has a knowledge where is currently located receiver.

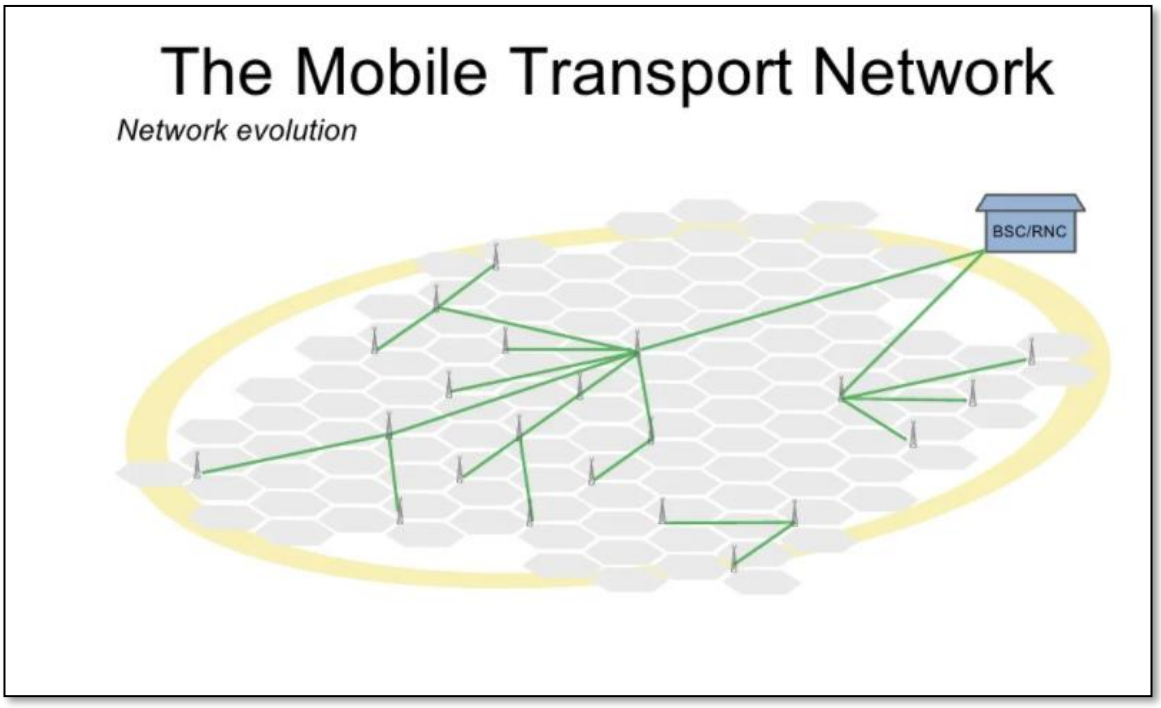


Figure 6. Mobile Transport Network (Mahmood, 2011)

As it has its security configuration, like any other hardware device, we want to integrate them into ESM to be able to check it. It is possible if they support the so-called CLI (command line interface) and it can be reached via standard protocols we are using, such as SSH, Telnet, NETCONF, and SNMP. CLI in this context means that after a connection via some of these protocols is made to the device and we are successfully logged into it using username and password, some predefined commands can be sent to the device which will return the current security configuration or set a new one. This kind of information exists in MINI-LINK's documentation, usually in Interface parts. Some of them can execute a group of commands, written in so-called scripts, which is similar to the bash script in Linux. For some of them, it can be just a manually edited file, but some specific applications exist which will generate those files.

Reading this kind of documentation is not easy. It is about technical information and instructions in the form of manuals that hold specific knowledge about the installation, configuration, maintenance, safety, and usage of a specific network element. The CPI

content is complex in a way that they are organized in a tree structure (hierarchical structure) and each root value has a group of subgroups.

The main outcome from this week is learning good practices for analyzing the information.

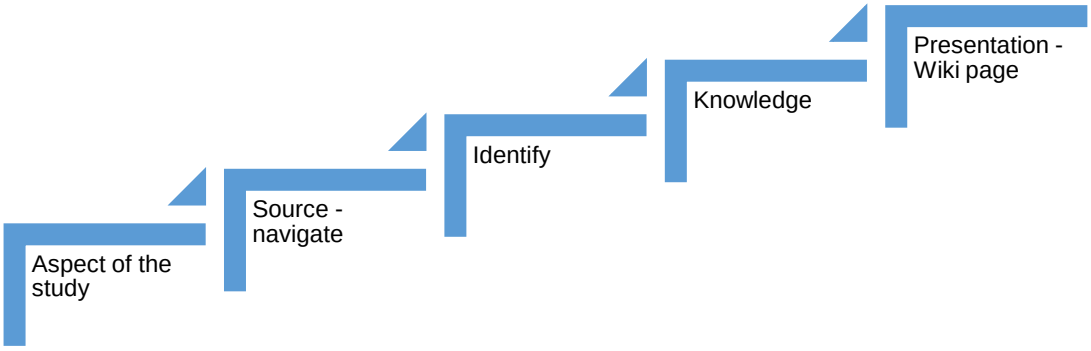


Figure 7. The main steps in the study

The figure above illustrates (Figure 7) steps in conducting the study. The initial step is the objective and reason of the study. The aspect of the study is to check if there are any configuration interfaces for the MINI-LINKs to use for PC integration. This step involves finding out who the target audience is and how detailed information should be. The source was known but the exact place in the data hierarchy was still unknown, so I spent some time browsing the CPI Store to find the right documents that contain relevant information. Understanding the data to gain general insights and knowledge is perhaps the main part of the study. When I had all the necessary information, I needed to find the best way to present it. As I mentioned, the result of the study was in the form of a table that holds only key information that can be noticed at a glance.

3.3 Observation week 3

This week's objectives are defined already on Friday last week. User stories are ready and listed in the backlog. One epic is assigned to me with 12 issues. For the ongoing sprint, which lasts one week more, there are two tickets in the "to do" stage in Jira. In short, I will work with one asset (let's call it asset X) that should have full coverage for rollback implementation, by storing data on ESM itself instead of the local file system.

Monday

The first working weekday. The first thing for me today is to check the documentation about the new rollback functionality that also provides the information about the current rollback and proposal for shared rollback data. After a few hours spent I have a clearer picture of what needs to be done.

The problem with the current rollback is that when we backup and modify the same files in multiple scripts, rollback won't restore the original state. The problem solution is to use shared rollback data, where rollback data would be shared between scripts at the same component level. This is required for scripts that are modifying the same configuration file on the same asset. For example, PAM configuration scripts.

So, what is the impact on scripts? For a better understanding of the control and function of each baseline automation script the following details are defined as the script's variables:

```
script_attributes:
  platform:
    - rhel6
  layer:
    - os
  target:
    - pamd
  compliance:
    applicability: full
    details: This field will be needed, if need to explain why
partial or none
  configuration:
    applicability: full
  rollback:
    applicability: full
    details: Backup file-based rollback.
```

The platform defines an operating system or other common platforms. The layer is where the control operates and can be an operating system (OS) or application. The target function/module/application for the script, such as pamd, sshd, etc. Compliance is used to

define applicability level and details for it. For example, applicability can be full, partial, or none. Full applicability means that the control can do the compliance check as the is expected to be implemented, and partial applicability is when the control implements the intent of the control only partially. If the applicability is full details will not be needed, but partial, and none require details. Configuration is used to define applicability level and details for it and the same logic is used as with the compliance.

Since the scripts' attributes are already defined in the scripts for the asset I am working on, but the script object must contain some new information about the rollback tag for the asset, my job, in this case, is to add this information under a new field script attributes as it is shown below:

```
shared: True
tag: scripts_sharing_same_tag_share_same_rollback_data
scope: asset
```

The scripts with the same rollback tag are sharing the same rollback data. These details used for shared rollback data need to be placed under the rollback field. Shared is defined as a Boolean value, so true or false. True is defined when this script is sharing data with other scripts and false is defined when the current rollback mechanism is used (non-shared data is returned in rollback). The tag is an identifier tag and scope can be an asset and it means that the data is shared at the asset level.

The current task is to implement rollback for pam scripts, therefore the work for the rest of the day was adding this information to each relevant script. The work on implementation will continue tomorrow.

Tuesday

Yesterday I finalized adding the new details used for shared rollback data. The plan for today is to work with backup data and import new ansible shared files into the script to implement new rollback functionality. Today's task is not very demanding since there was an example to follow (not strict). Include rollback and backup yml files hold tasks for performing those functions.

I will not go into details, with this part because similar work is explained in the section for the first reporting week. Only a few things about a backup into scripts. So, some scripts

require the creation of configuration files or another type of file backup. In the old script, there was a defined backup path but now we need a backup file list.

Wednesday

The agenda for today is performing the test for modified scripts. The question is: does after the configuration, rollback bring back the system to the state it was before the configuration?

I have some idea how to test does rollback works correctly and I wrote to PO about that, and he said that's exactly how should be tested. So, idea is to ssh to asset and save a copy of current pam files, to have a startup backup of pam files, for any case. Then I will manually write an incorrect value for some parameter in a few pam files. Running compliance check should report the incorrect value. Then running configuration should correct the value and running compliance check should show now everything is correct. Running rollback should now return the pam file in an incorrect state which I made at the beginning on purpose. In this way, I will prove rollback functionality is made correctly. Once I finalize all my testing on all pam files, I will return pam files to their original state from the backup.

But to perform testing on ESM, first I need to build Custom Catalog. It is custom because is used for testing one specific asset. There is a tool (shell script) under tools directory in the catalog repo, and the command and parameters for calling are:

```
./create_custom_catalog.sh EXXXXX asset_name
```

The first param of the script will be included in the name of the catalog and it is our signum. The second param of the scripts specifies the asset to build the catalog for. The catalog file is created as a zip file under the packaging directory, and it is ready to be used in ESM. For testing policy catalog changes it is advised to install an ESM. For the installation part, there is a guide that covers the procedure. But now there was no need for that since PO just ping me that I can use an environment that is already installed and can be used by others.

I perform some test as I explained previously, and it seems that rollback work as should.

Thursday

Starting a day by submitting code changes in Gerrit for review and verification. After making a commit, on the local machine, a few scripts start to execute. They are part of so-called post-commit hooks execution. They do a code verification, and there are 8 of them. One of them, which is checking yaml file verification, the so-called yamllint-check, is failing. I was following the error and it is an indentation error, which is happening in the script, which is automatically generated whenever some of the specific scripts, in this case, pam files scripts, are changed. The error was happening in many places. Trying to solve the error for a few hours, but as it was not in my code, it was hard to find it. Contacted PO, but he was busy with work, so waiting for a resolution. In the middle time, I was advised to commit the code in Gerrit as the same verification, and even more, are executed there. I have done it, but the same error is happening there. So, I decided to spend some time for the study, and on the research report for the Research processes course in Haaga-Helia that I have ongoing.

Friday

I spent the whole day solving the error from the commit which I made the previous day. I was working in parallel with PO, who is in the project for many years and is a very experienced one. To have the same code as him, I rebased my code on the top of the current master. One thing which I learn is how to check out the code from Gerrit. It means that even if someone else commits a code in Gerrit for review, we can take this code on our machine and do some additional work. As result after a few hours of checking, it turns out that there was a bug in the code that automatically generate the script, and the second error was that I had on my machine a different version of two utilities that the script is using. They are **yq** and **jq**. They are used for processing the yaml, json, and xml files, so you can select some parts of these files which you need. After solving those errors, I made a clear new commit with the same code as before and generated a new config file which has an indentation error yesterday. Abandoned the previous commit in Gerrit and push a new one for review

Weekly analysis

This week I had not-so-demand tasks. There was an example to follow (not strict) and an available Wiki page with a very good explanation of the current rollback functionality and with the implementation proposal. But on the other hand, the task was in some way time-consuming because it required full manual testing just to be sure that implemented rollback

functionality with new information worked as it should. However, I can say that there were still new things, like creating a custom catalog, better understanding backup and rollback functionality, hooks in git and **yq**, and **jq** tools in Linux.

To better understand the rollback functionality, I also needed a better understanding of the three execution modes for Ansible scripts that ESM supports. In configuration mode, the tasks are executed in order to make configuration changes on the target hosts. In compliance mode, the tasks are executed to verify that the requested configuration is still in place and hence the target hosts comply with whatever we want to achieve with the script. In rollback mode, the task is executed to set the target host configuration back to the values before the configuration was done. So, the main requirement for rollback mode is that the configuration is implemented for the target host. To be able to restore the configuration in rollback, the scripts have to know which values to use when restoring. This is done in two phases. When running configuration, before setting any new values in the target host, the script read the current configuration and stores the original data. The data can be stored, for example in a backup file in the target host or sent to ESM for storage - what is our goal now. So, common backup tasks need to be executed, like a task where the backup file is initialized. When running rollback, the data received in configuration is sent back to the script.

Another interesting discovery during this week was about Git hooks. Git hooks are scripts that are triggered to run on some particular event in git. They are located in the project folder, in the directory. git/hooks. There are two groups of these hooks: client-side and server-side. Committing-workflow hooks that are triggered by operations such as committing and merging are the most used client-side hooks, like the hook called commit-msg which generates git commit Change-id, which is upper in the commit message, after our commit message. Then there is a post-commit hook, which is run after the commit is done. (Chacon & Straub, 2014)

In our project, this hook is calling one python script, called test_selector.py, in folder test/test_selector. This python script then executes a few verifications in our code. Those are case-sensitive-filenames-check, copyright-check, crlf-check, git-commit-check, polctrls-policy-check, polctrls-script-check, tab-and-trailing-whitespace-check, and yamllint-check. The last one is the one that was showing the error we were trying to solve for two days with PO.

Tools **yq** and **jq** are used for processing yaml, json, or xml files, or output from some command which is in this form, so we can use pipe (|) in Linux to send this output as input to those utilities. Few examples of usage: (Codingpointer.com, n.d.)

If file test.json exist with context:

```
{
"param1": [ "1", "2", "3"],
"param2": "testing"
}
```

and if we want to convert it to yaml file then we can just execute the bellow command:

```
$ cat  test.json | yq -y .
```

and output will be

```
param1:
  - '1'
  - '2'
  - '3'
param2: testing
```

If we want to get as output only param2, for example, we can use the next command (output is shown too):

```
$ cat test.json | yq -y ".param2"
testing
```

I discovered there is too xq utility for similarly processing xml files.

Was a little bit suppressed with the fact that sometimes it can take a shorter time to do changes in code and testing than submitting the code for review and doing verification in Gerrit.

3.4 Observation week 4

This week's goals are defined during PI planning for the ongoing release cycle. On Wednesday new sprint will start, and user stories are listed in the backlog. My work tasks will be similar to the previous week since there are still controls that need implementation of the new rollback functionality.

Monday

The first day of the week and the new team setup. Two weeks ago, we were introduced to the new PO and Scrum Master. Previously our team did not have Scrum Master. Now we are going forward with the Scrum Master setup, who will attend and lead the daily meeting from now on.

Today's agenda is to start with rollback implementation on the same asset but for the ssh scripts. But since this morning I was informed it is my turn to do UI manual testing, I gave priority to that task.

Manual UI testing. The testing happens bi-weekly, and my team needs to cover different parts of the products (ESM) each time. In the past, there weren't that many volunteers for doing UI tests, so it was decided to introduce a rotation between team members. I thought that there is a Wiki page created for the PC development UI testing, but it is just a proxy for the UI Testing Matrix for easier access. There is a link on the top of the page for the current nightly build of ESM, which is an actual object under test. Now my team got the Threats submodule, and I have to test functionality with different use cases and try to possibly find any UI bug. If there is any, there is also a guide on the linked page for the bug report, and if no one is found, just tick the box for Zeus on the ongoing week column on the UI testing page.

Performing manual UI tests is something new for me. Naturally, first I introduced myself to the term and read some guidance. The next step was searching through CPI to find relevant information about Threat Management and threat detection roles. The threat detection function in ESM is used to monitor the network by collecting and analysing data generated by the assets in the network. I prepared the checklist which holds suggested users (roles) that I need to evaluate by functionality. There are three roles with gradual rising the authorization of user:

- Threats analyst: can create, modify, and delete threat rules for basic analytics and can monitor alarms and threats

- Threat detection package administrator: can deploy, read, and delete threat detection template package
- Threat administrator: read threat log search results for assets, create, delete threat detection logics, etc.

Only exploratory testing should be done, so I executed the test steps and compared the results with expected by logging in ESM as each user and performing tasks that are listed above. The goal is to test all UI elements and their functionality, which ensures that the application is performing the right way. This testing was intuitive and done randomly, meaning that there is no documentation and process to follow. The main goal of this testing is to find any kind of bugs by random checking. Manual UI testing took a full day, I have not found any bugs. The positive thing with this kind of testing is, from my point of view, that it allows me to take a wider picture of the product, in which the PC is an important part. But there were also some concerns from my side that I maybe missed some bugs due to a lack of knowledge and experience.

Tuesday

The goal for today is to complete rollback implementation and do some testing with the same logic as the previous week.

The ssh scripts, like SSH Idle Timeout Interval, Maximum Allowed SSH Sessions, SSH Protocol Version, have in common the SSH configuration files with the options and values as defined in the control variables. So, configuration items are **sshd_config** file that holds all configuration parameters for the controls. I spent some time looking at the file and searching the Linux Documentation available on the site linux.die.net.

Going back to my task, currently, we have one ansible script which is doing backup on local disk, and as input variables, it expects file name (variable named ‘file’) and backup path (variable named ‘backup_path). So, before doing configuration, we are defining those two variables:

```
    ssh_services_names:
      - name: "lde-sshd@sshd_config.service"
        file: "/etc/ssh/sshd_config"
        do_configuration: yes
    backup_path: "/opt/esm/backup"
    backup_path_owner: "root"
    backup_path_group: "root"
    backup_path_mode: "0600"
```

Now task, defined in a different yaml file, can be called and it will perform an actual copy (backup) of above file to the backup folder.

```
tasks:
  - name: "Include SSH backup tasks from ansible_shared_files"
    include_tasks: ansible_shared_files/ssh/common_backup_tasks.yml
```

After this line, backup is made. But, with the new way of backing up files to the ESM database, the same functionality implementation looks like below.

```
common_backup_file_list:
- "/etc/ssh/sshd_config"
tasks:
  - name: "Backup files"
    include_tasks:
ansible_shared_files/linux_file/common_linux_file_backup_tasks_001.yml
```

So, the script for backup to ESM database has different input variables name, and of course different name of calling task, but looks similar. The main problem to realizing such kind of task is to recognize how old and new functionality is working, to see which parameter is expected by old and new scripts, and to do the modification. And it should be done in eleven files. Even it looks quite simple at the end, sometimes it can take a long time to catch all required variables, and it requires building a new catalog, uploading to ESM, testing, solving bugs, etc.

Wednesday

Today is the last day of the ongoing sprint, so after the regular daily meeting, we had a brief discussion within the team together with Scrum Master about the next sprint and what kind of tasks are planned to be done in Sprint 3. This is the first time having this kind of meeting,

at least since I have been on the team. Usually, all activities are planned during PI planning. The idea of the meeting was to go through the ongoing items and make sure that everyone has item to work on, shortly what kind of tasks we plan to fulfil in Spring 3.

As the last step in this sprint, each team member needed to update ticket status according to the phase. I have two tickets with “Review” status, pending for review, therefore for now waiting that my colleague gives me feedback. After I got +2 for review in Gerrit from my colleague and passed verification, I was ready to merge my changes with the master branch. But I noticed that in one file still exists a logic for backup, released not with including backup scripts before configuration, but with explicit logic, released with file and copy ansible module. The logic is like in the code below.

```
- name: "Ensure backup directory exists, correct ownership, permissions"
  file:
    path: "{{ backup_path }}"
    state: directory
    owner: "{{ backup_path_owner }}"
    group: "{{ backup_path_group }}"
    mode: "{{ backup_path_mode }}"
    recurse: false
- name: "File(s) backup"
  copy:
    src: "{{ item.item[1].file }}"
    dest: "{{ backup_path }}/{{ item.item[1].file | basename }}_{{
        script_number }}_{{ inventory_hostname }}_{{
        esm_data.facts.execution_run_id }}"
    owner: "{{ item.item[1].owner }}"
    group: "{{ item.item[1].group }}"
    mode: "{{ item.item[1].mode }}"
    remote_src: true
  when: item.stdout_lines[0] != item.stdout_lines[1]
  loop: "{{ cipher_mac_original_and_expected_value_diff.results }}"
```

So, first with Ansible file module code ensure that the backup folder exists, and then with ansible copy module files are copied one by one (loop command) to the backup folder. This is not necessary code, as with my changes we will use logic for backup in the ESM database explained from yesterday's work.

I informed my colleague, and he agree and removed +2 for review. We made a new file without this inline backup, we did not just delete the code original script as it is maybe used in some other module that is still using the old way of backing up. But he was pushing the changes for review like separate commit, I did a review and before giving him +2 for review, I needed to test his code with mine. So, I have used, by his recommendation, the first time one git command called cherry-pick. The purpose of this command is to take only one

commit from some other branch and cherry-pick it (it means to merge it in my branch). As this full command exists in Gerrit in one of its menu, I just copy that command from Gerrit and paste it into my repository on the local machine.

```
git fetch ssh://ericssong-gerrit-url/usm/catalog refs/changes/79/11948879/3
&& git cherry-pick FETCH_HEAD
```

Before executing this command, we just need to be sure that we are on the right branch where we want to merge this commit. As usually I have plenty of local branches in my local repo, I first need to list all branches, to see the name of the branch on which I am working on this feature.

After cherry-picking his commit, I tested all functionality, and it was working correctly. I gave him +2 for review, so he was able to push the changes to remote master. Then I need to do a few things. First, as I don't need his commit (which I cherry-picked) as it is already in the remote master, I need to reset my branch to point to my previous commit, then go in master and pull changes from remote, then again to go on my branch to rebase my changes on the top of current master, and then to push the changes in Gerrit for review. All these operations in the same order are shown below.

```
git reset HEAD~1 (on my branch vIPWorks_rollback_sshd)
git checkout master
git pull
git checkout vIPWorks_rollback_sshd
git rebase master
git push
```

Eventually, I inform my colleague that everything is ready and tested in ESM, he gave me +2 for review, and after pressing a Submit button to push changes to the master branch, I need to wait 20 minutes to pass the final verification. When we push the changes to master from Gerrit, usually there is a queue with few changes which are waiting for the same, and they will be pushed into master one by one, after final verification. All was done correctly, and my changes are finally in the master branch. All I need to do more for today is checkout, my master, again, and pull changes from remote to my local machine. Now everything is ready for the next day's work. Overall, a successful day, and new things were learned.

Thursday

For today, I have a new ticket. I will continue to work on the same asset but this time on Password Lifetime and Warning time scripts. These scripts are about policies and controls that enforce password-based authentication settings. Therefore, I will work with scripts for controls such as Password Maximum Lifetime Restriction, Password Minimum Lifetime Restriction, and Password Change Warning Time. This time parameters are defined in **login.defs** configuration files. Once again, I searched for basic information from Linux documentation on linux.die.net.

Today's task wasn't demanded, and I completed the ticket that I have been working on. The task was like earlier, so I had procedures that I referred to and apply here.

I was planning to perform the test for modified scripts, but I run into some problems while I was trying to import the custom catalog into ESM. The catalog import failed because error during installation and I needed to get the logs. I ping my colleague for help. He proposed to use the script that is under the tool's directory in the catalog repo. The script provides several options, as can be seen in the figure below (Figure 8). The purpose of this script is many, and some of them are Software Gateway Certificate installation, adding Policy Catalog file to ESM, creating Policy Catalog, etc.

Figure 8. Script for pre-release tests

I used option (S) to fetch logs, but I could not get any conclusion from that. So, I decided to troubleshoot and pinpoint where the issue could be. The first logical thing was to generate a new custom catalog without my last code changes and import it into ESM. And it was successful. Therefore, my conclusion was that I have an error in modified scripts. Solving the problem lasted until the end of the day, but I found the reason for the error, and after I fixed the error, the new custom catalog was imported without any problems. I will continue testing tomorrow.

Friday

The agenda for today is testing Passwords scripts. The aim is to perform the test with the same logic that was explained last week. So, I will manually write an incorrect value for some control variable value in a login.defs file, for example, PASS_WARN_AGE parameter value is 7, and I will change it to 8. Running compliance check should report an incorrect value and running configuration should correct the value. Running rollback should now return the incorrect parameter value in login.defs file that I made at the beginning on purpose. In this way, I will prove that the rollback functionality was done correctly.

This time there were no issues, so I decided to spend the rest of the day working on my thesis. In the afternoon we had a weekly meeting with the line manager where we summarized this week's topics.

Weekly analysis

Overall, this week was mostly focused on the implementation of the new rollback functionality in the scripts, where I also needed to test them in ESM. Since some scripts must be able to use the same rollback data, and the rollback data must be stored only once when the first script is running configuration, it was necessary for learning purposes to analyse backup files, in this case, files like login.defs and sshd_config file.

The **/etc/login.defs** file provides default configuration information for several user account parameters. This file is a readable text file and provides configuration parameters such as name and value where parameter values may be strings, Booleans, numbers, and long numbers types. (die.net) This file provides configurations like password minimum days, password maximum days, password warning age, etc. (Baydan, 2020) Configurations that were of interest to me are: Password Maximum Days used to set how long a user password will be valid and after specified days should be changed which is forced in login, Password Minimum Days used to configure the minimum days to change password and Password

Warning Age used to set when the warning messages about the password change will be shown. Each of those configurations has the default value.

The location for **sshd_config file** usually is /etc/ssh/sshd_config but can be changed. So, the configuration item is /etc/ssh/sshd_config file that holds all configuration parameters for the controls. Below is an example of ssh config file (short version, for description purposes):

```
#       $OpenBSD: sshd_config,v 1.103 2018/04/09 20:41:22 tj Exp $
Include /etc/ssh/sshd_config.d/*.conf
#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
#HostKey /etc/ssh/ssh_host_rsa_key
PasswordAuthentication no
#Banner none
```

sshd config file is the main configuration file for OpenSSH server. We can see at the beginning that it includes all other optional config files from the sshd_config.d subfolder, line `Include /etc/ssh/sshd_config.d/*.conf` contains many values, most of them commented, which means the value used for these commented parameters will be default values. For example, the well-known parameter for OpenSSH server (and other ssh servers) is port number (#Port 22), and it is the port number where ssh servers are waiting to accept connections from other clients. As the default value is 22, this parameter is commented (using character #), and if we want to change it, we should delete the comment character and write desired value. On the other hand, the parameter `PasswordAuthentication no` is set explicitly to no, as its default value is yes. (die.net) To do my task, of implementing rollback functionality for sshd config file, it is not needed to understand the meaning of all of these parameters, but it is good to know some of them.

Another thing is UI testing. To understand completely what the major steps are for performing the UI test, I begin with a definition of the term UI. It is an acronym for User Interface, the part of the application visible to a user and that user interacts with. UI testing is about checking two things: how an application (software) handles actions performed by a user and whether the visual elements (text fields, buttons, links, etc) are displayed correctly and work as intended (Testim, 2020). I first read some guidance for UI testing, especially about exploratory testing since it is a type of testing where test cases are not created in advance, but testing is done on the “fly”. It is widely used in Agile models and is

about discovery, investigation, and learning. The main characteristic is the personal freedom and responsibility of the tester, and that can be on one hand advantage for this type of testing but in another hand, a disadvantage since this testing fully depends on the tester's skills and knowledge of the application. (Hamilton, 2022)

Also, this week I became more confident with Git after pushing code for review in Gerrit and using a cherry-pick to pick a commit from my peer and apply it to my commit. It was necessary for applying changes in one of the included files used in my commit. The reason for using a cherry-pick is to test the changes made by my peer work, before pushing them into master for future usage. Cherry-pick is one of the Git features, mainly used for collaboration. Git cherry-pick is a command that takes changes from a target commit and places them on the HEAD or the current working branch. This feature can be a useful tool but not always a best practice. There are a few scenarios when is recommended to use a charry pick, for example, team collaboration, undoing changes, and restoring lost commits and bug hotfixes. (Atlassian, n.d. b) Even though there are many useful scenarios for using a charry pick, there is also a recommendation to use it as rarely as possible, and the main reason is that it easily creates duplicate commits. Learning about Cherry-picking, and Git, Atlassian’s guide is usually helpful, and simple to follow.

3.5 Observation week 5

The work for this week is to continue with the implementation of the rollback functionality. I have tasks in the “to do” stage on the Kanban board, assigned to me at the beginning of the PI planning, and they will concern the same asset. Also, I have the freedom to choose in which order tasks will be done.

Monday

I started to work on rollback implementation for the script which is doing configuration for SSH Banner. The first thing to understand is for what and how the banner is used, then to analyze and understand our scripts for banner configuration, and to change a code to use backup functionality provided by ESM.

The purpose of banner is to show some warning message when someone wants to log in via ssh protocol to the company’s server. So, it will try to discourage unauthorized access to the server. It will be shown on the remote computer from which the user is trying to ssh to the company’s server, and it appears before asking for a login password. How banner will look like is defined in some file, usually put in the folder /etc. The full path and filename are then added in the sshd_config file in /etc/ssh directory, as shown below.

```
Banner /etc/filename
```

Understanding the scripts included in banner configuration was not a hard task. They are quite like the previous week's rollback implementation.

Later, during testing, I noticed that in configuration files still exists a logic for backup, released not with including backup scripts before configuration, but with explicit logic, released with file and copy ansible module. The same problem I had the previous week, but this time I know what needs to be done. So, I created the new ssh configuration banner file with no backup by removing backup-related tasks which are now running after backup tasks are executed.

This day was relaxed and the goal for today was achieved. The changes were pushed in Gerrit for review.

Tuesday

Today I moved to another ticket. Like yesterday I need to do rollback for MOTD (Message of the Day) file. The MOTD is displayed immediately afterward after the user is successfully logged into the server via ssh. What will be shown is defined in a file called motd under the directory /etc.

The script must contain some new information about rollback_tag for the asset. So, once again this information will be added under a new field script_attributes, and this changes the catalog definition like this:

```
rollback:
        applicability: full
        details: Backup file based rollback.
        shared: true
        tag: asset_name_LINUX_motd
        scope: asset
```

Understanding the scripts included in MOTD configuration was not a hard task. They are quite similar to Banner rollback implementation. The main difference is the way how display messages. One way is using the issue.net file and displaying a banner message before the password logging prompt, and another way is using the motd file for displaying a banner message after the user has logged in.

Wednesday

Moving to another thicket. Today I will be working with the Event Audit Logging control, and it seems that I can expect more challenges. As usual, I started by looking at the script and analyzing the backup file, in this case, audit files.

Log files are collecting some information about when and what happened. Specifically, audit log is responsible for collecting important system events, so the system admin can see what is happening in the system and the reason for some specific behavior. This is important for the security of the system and sometimes for debugging some problems in the system. This information is written in the file audit.log in the directory /var/log/audit. (Hahn, 2021)

The steps are the same as for the previous controls. Backup file-based rollback is applicable for this control; therefore, I used the same yaml files for the backup and rollback block in

the code. Since I finished the assignment for today, I spent the rest of the day writing the thesis.

Thursday

This morning I started looking into the NETCONF files. The task is the same, implementation of the rollback functionality. Based on my understanding, the scripts that include tasks for backup and rollback are already in master, and my job is to update the script's attributes and include those files. As for previous rollback functionality, released for ssh and pam scripts, the first thing I did is to understand what NETCONF is and what the scripts are doing. I will do a more detailed description in a weekly report of what I learned about NETCONF. The difference compared to ssh and pam file which I have done in previous weeks is that now backup and rollback is done as value base, not file-based, which means there is no file to back up, but we have to save the values of every specific parameter from NETCONF to EMS backend. Understanding this functionality took quite a lot of time for reading all information on the internet and looking into the existing code.

Implementation at the end was relatively simple. I will explain it only one of the controls we are testing via NETCONF, Interval for Counting Consecutive Invalid Logon Attempt. Implementation of other controls is similar. So, first adding the rollback variables

```
rollback:
  applicability: full
  details: Value based rollback.
```

and then including files that are doing an actual backup before configuration starts (so before any changes in NETCONF configuration).

```
- block:
  - name: "Include NETCONF backup tasks from ansible_shared_files"
    include_tasks: ansible_shared_files/netconf/netconf_backup.yml
  - name: "Include NETCONF backup tasks for superuser"
    include_tasks: ansible_shared_files/netconf/SU_netconf_backup.yml
```

In the block, as can be seen above, the included file is used to create a backup of the original configuration and it is used only in NETCONF scripts with some exceptions. The task in this file expects some variables from previous includes such as pre-conditional tasks.

In the end, I have to add the scripts which will do a rollback, and they will be executed when the user requests it (rollback variable in the block when statement will be defined in that case).

```
# ----------
# ROLLBACK
# ----------
- block:
  - name: "Include NETCONF rollback tasks from ansible_shared_files"
    include_tasks: ansible_shared_files/netconf/netconf_rollback.yml
  - name: "Include NETCONF rollback tasks for superuser"
    include_tasks: ansible_shared_files/SU_netconf_rollback_tasks.yml
  when: rollback is defined
```

The rollback is defined as a separate block of tasks. In this case, the included file is used to rollback a backup of the original configuration.

Friday

Today on a daily meeting I was informed that there is a possibility that the ticket about rollback for pam.d files will be rejected, and implemented changes abandoned in Gerrit. The reason is removing configuration functionality from the pam.d related scripts, so no need for roll-back anymore. There was no detailed explanation why we remove the configuration from the pam.d scripts.

After implementation of the rollback for NETCONF files yesterday, I come to the testing part. Manual testing is now different as backup is done by value, so there is no file where I can manually change a parameter to a value different than a configuration value. I will do then do configuration, which will write the correct value in the config file, and then do a rollback, which will return a non-valid value in the config file. I had to think carefully about how to test value-based backup and rollback. Following the same logic as for file backup functionality, I changed a configuration part to write first some different values than how it should look in the default configuration. After running the configuration, now this value should be in the NETCONF configuration. Returning the value to the default one, I run a compliance check, and it should fail on the value which is currently in NETCONF and which I am testing. Then, after running configuration, and compliance check, everything should be ok, and after running rollback, and then compliance it should fail as it will contain the wrong value which

I set by purpose at the begging. In this way, I will be sure rollback is running correctly. And, after successful testing, I was doing the same for all parameters which we are configuring, so testing rollback functionality passed successfully. At the end of every test, I had to return the parameter to the correct value and write the configuration, so the server will be in the correct state at the end.

In the afternoon I had our weekly meeting, just to summarize the week. I am informed by my colleague that she checked two of my commits and for one of them, she put some comments. Shortly afterward, while I was still in the meeting, I received a message from the Scrum Master to ignore the comments, because in any case, the commit with pam.d files will be abandoned, which I did. Now, all changes made will no longer be counted as an open review and will not be committed to the master branch.

Weekly analysis

Like the previous week, this week has been fulfilled with new findings and new knowledge. SSH banner and MOTD are quite easy to understand, and I already described them in a daily report. Understanding the event audit log took a little more time, and NETCONF almost a full day.

An audit log also called an audit trail, is essentially a record of events and changes. IT devices across the network create logs based on events. Audit logs are records of these event logs, typically regarding a sequence of activities or a specific activity. They capture events by recording who performed an activity, what activity was performed, and how the system responded. (DNSstuff, 2020) The audit log can give an administrator information about the time and order of operations, were they successful or not and it is extremely important for security reasons and for debugging the system if something fails. In Linux, the audit log is captured in folder /var/log/audit directory, in file audit.log. When a file becomes bigger than some size, specified in system configuration, it is rotated, which means it will be renamed by adding the number at the end of the full filename (for example audit.log.1, audit.log.2, …). There are many types of events that are added in this file, for example, ADD_USER, so when a new user is added to the system, then USER_LOGIN, when a user is logged into the system, and so on. For example, for RedHat Linux, all types of events can be found on the documentation page. (Red Hat, n.d. b) A short part of audit.log file is shown below.

```
type=USER_START msg=audit(1647672879.241:12400182): pid=27624 uid=0
auid=4294967295 ses=4294967295 msg='op=PAM:session_open acct="root"
exe="/usr/bin/sudo" hostname=? addr=? terminal=? res=success'

type=CONFIG_CHANGE msg=audit(1647672879.273:12400183): pid=27624 uid=0
auid=4294967295 ses=4294967295 op=tty_set old-enabled=1 new-enabled=0 old-
log_passwd=0 new-log_passwd=0 res=1

type=USER_END msg=audit(1647672879.273:12400184): pid=27624 uid=0
auid=4294967295 ses=4294967295 msg='op=PAM:session_close acct="root"
exe="/usr/bin/sudo" hostname=? addr=? terminal=? res=success'
```

Fully understanding every record is part of the Linux administrator job, but shorty those events shows the type of event, process id (pid), user id (uid), and so on.

NETCONF is a standardized interface for making changes in network configuration on the remote machine. It contains the server and client. So, on a remote machine NETCONF server must run to do read or write some network configuration parameter, and the client is connected to the server and gives the command for reading and writing of parameter. Reading and writing are done using a Remote Procedure Call (RPC) from the client against a server. NETCONF configuration is done using XML data formatting, and parameters reading is done using the so-called xpath, which is analog to a path in the directory when we are searching for a file, but here we are looking for some attribute in xml file. (Tail-f, n.d.; NOKIA)

In our scripts, we are configuring a few parameters in NETCONF, and every script is configuring only one parameter. We are configuring how many times normal and superuser can try to log in without success and in how long interval, how long will lockout take after unsuccessful login attempts, parameters for the complexity of the password, and how long time password will be valid (minimum and maximum value), parameters for network configuration. Reading and writing are done via a new ansible module which is the propriety of Ericsson and done as a python script.

The relatively simple and repeatable tasks for this week turned out to be time-consuming at the end of the week due to a lack of experience with the NETCONF interface. There is also a lot of code I have to go through to understand specific methods, logic, and Ansible's modules used in scripts. My learning for this week is more theoretical and basic because it was necessary to understand the general terms used in scripts.

3.6 Observation week 6

My week's goal is mainly based on finalizing the epic for implementing rollback functionality. This week the final sprint will start, so by the end of the week, I should know what my next assignment will be.

Monday

The objective of the day is to try to finalize rollback implementation for the rest of the controls (scripts) for asset X. I will have a few tasks to look into, and I hope there will not be any issues.

I spent most of the day doing the modification of the scripts in the same way as the previous week, so here I will not go with details. The main learning thing for me today was a deeper understanding of the audit logs file. One of the scripts that I was working on was ‘Permissions on Audit Logs’. The difference here is that target for this control is audit.log file but the rollback should be value-based. The script saves only file permissions and only permissions for the active log file is restored, not the file since that file is continuously growing. If we roll back the file, then file content will lose plenty of information.

Rest of the day I took off from work and dedicate my time to writing the thesis report.

Tuesday

Today’s task is documentation update. I will go once again through all scripts for the asset and double-check attributes.

To enable a better understanding of controls, function, and target applicability of each baseline automation script the goal is to automatically build an applicability matrix from the Policy Catalog. Therefore, I will follow the proposed solution to add details to each script (what was mainly done with rollback implementation) and in that way making easier to generate the matrix. There is a python script, that generates csv raw data by analyzing policy descriptions from the policy_config.yml file. As result, the excel document is used to visualize policy coverage for assets and interface.

For each script following details need to be defined:

- Platform: to define the operating system or another common platform name for it.
- Layer: where the control operates (OS, application).

- Target: the target function/module/application for the script.
- Compliance: to define applicability level and details for it.
- Configuration: to define applicability level and details for it.
- Rollback: to define applicability. If the configuration is implemented, rollback is implemented also.

The goal for today is achieved. One observation from today is that I need better organization when taking notes on progress and work done during the day.

Wednesday

Today's daily meeting was longer than usual. The main reason for that is the sprint review and it happens on the last day of the sprint. The meeting is attended by every team member and PO. We went through product backlog items and all items are updated according to the state. So, following the definition of done, some of the tickets are closed, while others are still in the work process. We have been informed that a one-hour meeting for planning the next sprint (transitional sprint) will be scheduled tomorrow. There was also a discussion about the Policy Catalog release. It seems that now someone else needs to get involved in that process because the PO is leaving soon and someone else needs to know how to do it in case

I was interested in the proposal for joining the group that is responsible for the release process. I spent some time looking at the Wiki page, which I found, about the Policy Catalog release process steps. I was reading and trying to understand the requirements, and tasks that are involved in the process. The process consists of several activities, such as merging all development items, compatibility tests, documentation updating, etc. It seems interesting, not complicated but time-consuming. There will be plenty of new things for me to learn, so I decided to send a message to the PO that I am interested to be part of the process if he thinks that I can manage it given to my experience. I am expecting more information on this topic tomorrow.

At the end of the day, a colleague from another team pings me for help. He was assigned to develop the rollback functionality but for the other asset, and he is running out of time he asked for help. Since I am still waiting for a review on most of my commits, I can say that I have free time and I could help him. So, either this week or next week I will take over some of his tasks.

Thursday

The agenda for today is not clear, depending on outcomes from the sprint planning. But in any case, I think I will find time to talk with a co-worker about the help he needs.

This is the last sprint in the release cycle – the transition sprint. At the meeting, there was a discussion on what needs to be upgraded, improved, unfinished tasks, etc. If there were no specific problems with task execution, all thickets need to be closed in the next two weeks, or sooner. Each of the team members takes time to discuss progress and problems if any. All feedback is considered and based on my understanding the PO or Scrum Master will update the product backlog to be ready for the next sprint.

Not long after the meeting, a have noticed that I skipped one ssh script in the commit that I made last week, and that this script holds some additional information that is unfamiliar to me. I ask one of my peers for an opinion and we spend almost two hours finding the best solution. I need to make some changes in all ssh files and then test it again.

Shortly: The Disabling Direct SSH Root Login script edits the same file as other ssh scripts (/etc/ssh/sshd_config) but also /cluster/etc/cluster.conf. My logic was that we need to take a backup of the file that is included in the script, but the point is that as log the tag name attribute is the same for all ssh scripts and it is, all files changed, regardless of whether they are included in the particular script or not, need to be under the backup_file_list.

Basically, customers cannot do rollback only on a chosen script but if they click on rollback then all changes will be restored but only once and it will be the oldest of all backups that used the same tag. And that is the reason why we need to include all files in all ssh scripts with the same tag even if no changes are made.

Therefore, for the rest of the day, I was adding the relevant files in backup_file_list for all ssh scripts and implemented rollback for Disabling Direct SSH Root Login script. Tomorrow I will spend more time on testing since everything comes down to the testing and results.

Friday

The plan for today was to test and confirm does rollback works fine, and what was the case. Later, in the afternoon we will have a short discussion about the PC release.

During the call it was suggested that it will be good if we deploy the server together. The goal was to install an ESM environment that will be reserved only for release testing of the catalog. The easiest way to install ESM is to use Jenkins, which will build the working environment and pull it up ready to use. The available environments and the reservation table can be found on the Wiki page. There is information about what environments should be used for Policy Catalog development, what slots on the server, the name of the environment, the description of each parameter, etc.

Installing the environment is a CI job. The actual work in Jenkins's lab takes only a few minutes. What is important here is to be sure that you have all relevant information (parameters) for the concrete installation. We need to build and deploy manually ESM by entering the build parameters, such as slot, ESM version, release branch, etc. Once the build is triggered, we can easily check what is happening by going into console output. If is something wrong can be seen immediately at the beginning. This CI job takes about 15 minutes, sometimes much more. In our case, it was about 20 minutes. Now the environment is ready for testing.

Plan for Monday is to establish how far are we with the content, are there any pending items, or some problems, and usually build the first version of the PC without those items, and later as items are getting ready for marge to the master, the new version of the catalog can be made and upload to ESM.

Weekly analysis

In general, this week finished with new things and findings. It started as previous two weeks, working on the implementation of the rollback, updating the scripts, and in the end, I felt more satisfied than in the previous one, because I got a much deeper understanding of how configuration and rollback work together.

In case that asset hardening could not be completely configured, or it is done but for some reason need to be rollback to what it was before ESM configuration, the user needs an easy way to perform a rollback. It was not sensible, in long run, to store all configuration files backups on the assets, especially since there are also nodes (assets) that do not support local storage. So now it is possible to make a backup copy of the configuration before enforcement of all controls – each control individually. It is done by using shared rollback functionality. The oldest of all rollback data sent to ESM by configurations, so the user cannot do rollback only on a chosen script (control) but by making a full rollback all changes will be restored but only once and it will be the oldest of all backups that used the same tag.

We will get a message: Script passed: the script was not executed because it has shared rollback data with the script “Script_name.yml” which performed the rollback,

This week, I spent time analysing the PC release process. Under consideration is the new PC version release, and the goal for me is a basic understanding of what each new release brings to the project and why it's important to document all changes. Each new version means some editions with some major or minor differences from the previous version.

What is release management? Release management is the process that includes the management, planning, scheduling, and control of an application (product) build – from development to testing to deployment. The release management is utilized anytime when a new product is produced, or a new version of an existing product is released based on changes in the existing version. (Simplilearn, 2021)

Every organization has elements of release management in its product management process. Information Technology Infrastructure Library (ITIL) is the most widely used framework for the governance of technology products and services. According to ITIL, the purpose of the Release and Deployment Management process is: To plan, schedule, and control the build, test, and deployment of releases, and to deliver new functionality required by the business while protecting the integrity of existing services (BMC Blogs, 2016). What is important here is to understand that is not every release has a version that will be distributed to the customer. Sometimes versions are created within the organization only for internal testing and development.

As can be many versions of existing product /system/application it is important to keep tracking of the versions. Sometimes is required to retrieve the old version, so versions management needs to use some techniques for components version identification. One of the wild used techniques is version numbering. As can be seen from the picture below (Figure 9), a version number is added to the product name. The first version is always marked as 1.0, and each subversion is marked as 1.1, 1.2, and so on. Once the new release is created (release 2.0) the process starts again at version 2.1.

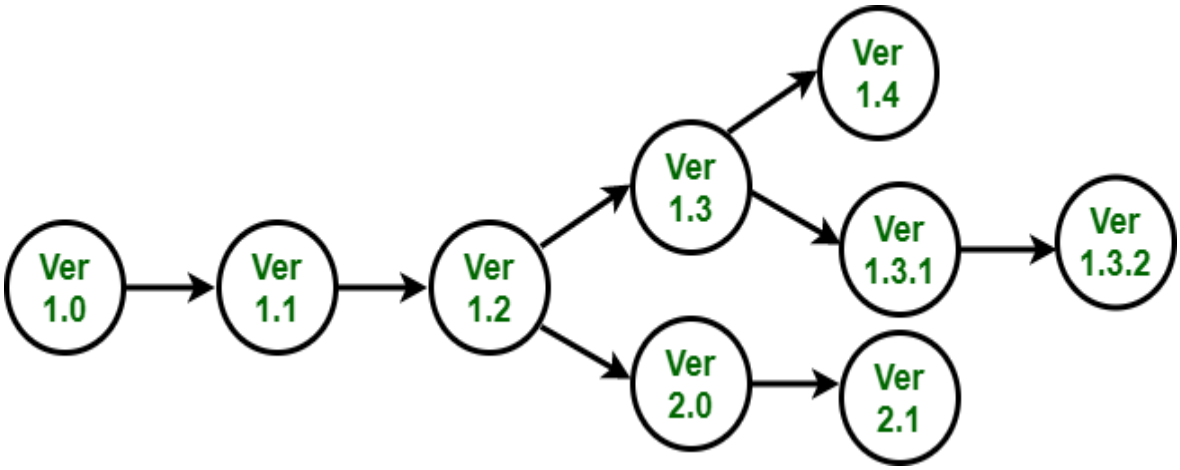


Figure 9. Version Numbering (GeeksforGeeks, 2021a)

According to (semver.org, n.d.) in the world of software management, the bigger the system grows, the more packages are integrated into the software, and releasing the new versions can quickly become a nightmare. So, following the system “Semantic Versioning”, also known as SemVer represents the scheme for using meaningful version numbers as the way to follow the modification that has been made from one version to the next. This versioning system has been on the rise over the last few years, and follow the form of Major, Minor, and Patch version.

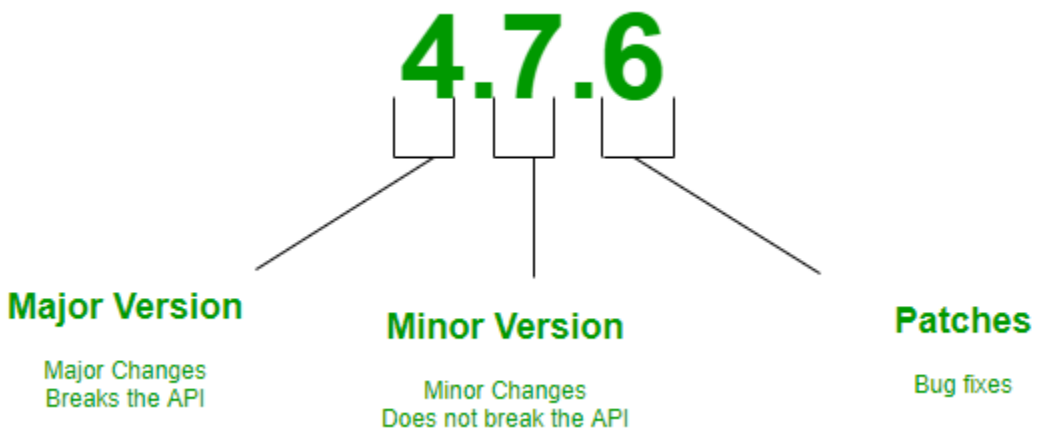


Figure 10. The components of SemVer (GeeksforGeeks, 2021b)

The SemVer is a number that consists of the 3 components in the format of X.Y.Z, where X stands for MAJOR version when incompatible API changes are made, MINOR version when new functionality is added, and PATCH version used for bug fixes (semver.org, n.d.). When the minor version is increased the patch version must be reset to zero. If the current version is, for example, 4.7.6 then the next upgrade for a minor version must be 7.8.0. In this way, all changes in the software development phase can be tracked, and what is more practical the versioning helps with an easier following on what type of changes have taken place.

Therefore, to summarize the previous text and the Policy Catalog release process: A release is a process that consists of specific steps, depending on the unique characteristics of the product/application, but the common steps are:

- *Request*. It is the first step where requirements are gathered and can be the new feature or changes to existing functionality, or just bug fixing.
- *Plan*. The plan typically addresses what changes will be included, what risks may occur during the process, etc. The plan is done during the PI planning.
- *Build.* Working on epics, users' stories, etc. In this stage planned work is executed.
- *Testing.* Once every issue is closed on Jira, and every commit is merged to master, the PC is deployed to a test environment (ESM). Every asset needs to be tested that has changed in the current PI. Different tests are performed in this stage, for example, upgrade testing – PC should always be imported to an ESM where the previous policy catalog is installed already. The reason for this is to test the customer Policy Catalog upgrade scenario.
- *Released.*

One of the main steps in the PC release process is documentation updating in the catalog repository and that involves updating the policy_config file. This includes the following steps:

- Update the supported asset list if there are any new assets for the release
- Update the changelog. The changelog is a record that keeps track of changes that have been made. The changelog holds the following values: new features, new controls, bug fixes, removed features, changed features, etc.
- Update the version to match the new Policy Catalog version
- Update the version of the controls
- Update the policy domains file which contains information regarding the supported asset types and domain mappings.

The main objectives of release management are managing risks, an overview of all versions of the product, and maintaining alignment and harmony between software development and the business (Simplilearn, 2021).

This week was productive even though I did not have any new assignments. Sometimes I need to remind myself that I am an intern on the team and that there is no big pressure, and the main goal for me is to grow professionally by taking the small steps, day by day.

3.7 Observation week 7

The plan for this week was to work within the team on the PC release, but due to sickness I needed to take the whole week off, therefore daily entries for this week are missing.

I was hoping to take on more responsibilities by being involved in the PC release process and in this way improve my knowledge, communications skills, and teamwork. But things did not go as planned. Therefore, I moved my interest more to the tools used in the release process, and my focus come to Jenkins, which will I try to explain in further text.

Weekly analysis

As the goal of a software project is to deliver a solution to the customer, the first step in this process is to build a solution. In this process very often plenty of developers and testers are included. Integrating the work of programmers and testers into one solution is a very challenging process. To answer this challenge, a completely new process is discovered and is called Continuous Integration (CI). “Successful CI means new code changes to an app are regularly built, tested, and merged to a shared repository. It’s a solution to the problem of having too many branches of an app in development at once that might conflict with each other.” (Red Hat, 2018c)

Delivery and deployment term is used many times interchangeably, but this process can be viewed as separated. The delivery part, which is following the process of automation of build, unit, and integration tests, involves test automation and release automation. The final stage, deployment of the new solution, automates releasing a software project to production, so the product will be finally available for use by the end-user. (Red Hat, 2018c)

The figure below (Figure 11) shows the process that consists of a series of tasks that are executed in a particular order. The process is starting with development. The first event (code committed in Git) triggers a build, test, staging, and development, and finally, the product is in production, ready to serve users.

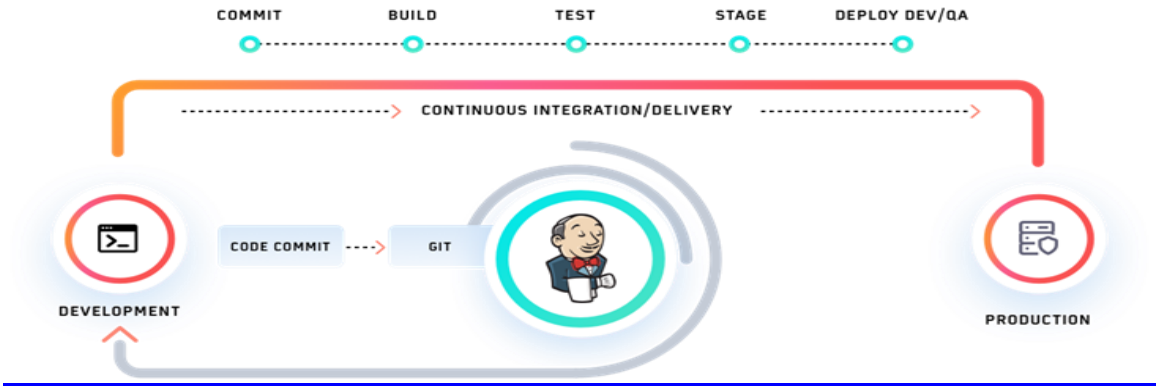


Figure 11. CI/CD (Shetty, 2020)

This whole process is called CI/CD, and the most popular tool for implementing this process is Jenkins. McAlister defines Jenkins as a highly acclaimed award-winning build and automation orchestration solution. It represents the cumulative efforts of hundreds of open-source developers, quality assurance engineers, and DevOps personnel worldwide. What makes this solution uniquely innovative is it is continuously updated, improved upon, and supported by this cohesively vibrant open-source community. It is through this open-source development effort that Jenkins has remained at the forefront of Continuous Integration, and Continuous Delivery practices. (McAlister, 2015, p. VII)

Jenkins is built to be easily adaptable to the changes and requirements of the environment and to be extremely scalable, so it can support from the simplest to the most complex software projects, independent of programming languages used for the project. This is achieved first by looking into CI/CD as a separate software project and allowing easy adaptation by a modular approach to every single step in the process, which will be executed by modules (plugins). There are many already available plugins, over 1500, and they can be used in the way which is the most suitable for the current project. The smallest unit in Jenkins, called jobs, can be written by developers using Groovy language. Jenkins is written in Java programming language and executed by Java Virtual Machine (JVM), so the natural choice for the language which will be used for jobs definition comes to Groovy, which is an excellent scripting language producing the bytecode being possible to execute on JVM.

Crucial concepts in Jenkins are pipeline, node, agent, stages, and steps. Pipelines are Jenkins jobs enabled by the Pipeline (formerly called “workflow”) plugin and built with simple text scripts that use a Pipeline DSL (domain-specific language) based on the Groovy programming language (Jenkins, n.d.). From an implementation point of view, the pipeline is just one script that defines necessary steps and jobs to be executed to finish one process in Jenkins.

```
pipeline {
    agent any
        stages {
            stage ('Build') {
                steps {
                    echo
                    'Running build phase. '
                }
            }
        }
    }
```

Agent defines where the Pipeline will be executed. It can define on which machine some pipeline will be executed, and in this way work in Jenkins can be divided to be executed in parallel on multiple machines. An agent can be defined as single at the beginning of the pipeline script, or it can be defined in a separate stage. (Shetty, 2020) In this way jobs in multiple stages can be executed on different machines in parallel, so the process can be done much faster.

A node is a machine that is part of the Jenkins environment and is capable of executing a Pipeline. It is very similar to Agent, but it is used in a scripted pipeline, and Agent is used in a declarative pipeline. (Jenkins, n.d.)

Stages include all the works which needs to be done. It can consist of multiple stages, and every stage consists of one or multiple steps to be executed for every specific task to be done. So, for example:

```
pipeline {
agent any
    stages {
        stage ('Build') {
            steps {
                echo
                    'Running build phase. '
        }
        stage ('Test') {

        }
        stage ('QA') {

        }
        stage ('Deploy') {

        }
        stage ('Monitor') {

        }

    }
 }
```

Given that Jenkins is a very complex tool utilized by developers in various ways, I tried to build a high-level understanding of the usage of Jenkins. The main point was to understand what is happening in the background when I commit code to the repository (Gerrit). So, the actual involvement of Jenkins is to be triggered by some event, compile the code if there is a change, notify if there is a failed build with a possibility to track the error, and finally, if the build is successful the code can be merged to master. This is a very big picture of Jenkins, and in reality, it needs, for complex projects, a team of DevOps engineers who will continuously monitor and build new scripts, always improving a process, and making it faster and reliable.

3.8 Observation week 8

This week is mostly about PI planning and given that we will have more meetings within the team. A new sprint starts on Wednesday, so I will start working on new items from the backlog. The main goal this week is to catch up on the missed things and start working on the new tickets. I hope that I will be able to back on track and figure up all steps necessary for the next release process.

Monday

There is no particular plan for today. I am still far from well, but I wanted to continue with work.

There are a lot of things that I missed, especially related to the release process. On daily, I was informed that there were some problems with one of the scripts on what I was working on, but lucky since I was in contact with one of the teammates for this particular control, they manage the bug. I can conclude that for most of the things related to the rollback functionality there was no full information and that was expected to get some error. Since we have just started using the shared rollback feature, and during the previous PI only I was working on this, and given my limited experience it was expected to have some “problems”. Based on my understanding the main problem was the wrong scope definition of where the data is shared – at what level: interface or asset.

I decide to contact one of my peers for a more detailed explanation of the asset vs interface situation and conclusion after the new information that I received is: If the configuration files are synchronized on the components, such as they are common ones stored in, for example under /cluster then the scope should be “asset” because we only want one of the scripts to restore that file and only once. The asset that was in question has multiple components and the config file is not synchronized so we needed to use the scope to be interface because we want the script to backup and restore the file on each interface level (component). Therefore, each asset is unique, and the main thing to have in mind here is to understand at what level the data is shared.

For most of the day, I was trying to pick up all information regarding the PC release process. Luckily there are some posted videos from the previous week where are all the important things presented.

Tuesday

Today we had the PI planning. The focus is on managing tasks for the upcoming sprints. We started with a brief retrospective of what was done in the previous PI and continued planning for the next PI. For most of us, there are no current tasks, and the agenda for today is mostly an analysis of current work, what was good and what went wrong in the previous release cycle, what steps are required to improve task execution, and better cooperation between team members. The common session is scheduled for the afternoon, at which multiple teams working on ESM development gather to review plans and risks.

While the overall planning is done by the SM and the PO, the main purpose of the meeting is to introduce the team to the most important tasks – presenting the highest priority backlog items. The flow of information is two-way, which means that the development team has technical information that needs to be shared with PO and SM. The work that needs to be done is defined and risks are identified.

Wednesday

The day started with a daily meeting. There were not many things to discuss since we had a PI planning session yesterday.

The testing of the new PC release began last week. The master branch already has all the content. The usual target is to complete development work in 3rd sprint in order to have enough time to release testing and PI preparation in the 4th sprint. The practice is to create Epic in Jira for all the activities that are required for the release during the PI. The Epic contains user stories like Manual UI Testing that is done bi-weekly and testing of the new and modified scripts.

Most of the tasks have been completed, but there is still a need to perform some tests in the new “clean” environment. So, the rest of the day I dedicate time to performing final tests in ESM just to be sure that there no issues.

Thursday

Today I will start with the new thicket. Once again, I have to implement rollback for one of the assets which are deployed in our lab. The task is familiar but given that there were some problems with the rollback scope, and understanding the same, I spend some time reading the Wiki page that was written by one of the teammates. The page holds some new

information and an explanation of how we need to use the new shared rollback feature. Therefore, the most common usage is file content backup for Linux-based assets. Usually, we use scope “interface” when multiple scripts are modifying the same file, but it is not in a shared/synchronized directory, and “asset” when the file location is on a shared/synchronized directory. So, the location of the configuration file is the starting point in understanding the scope of the rollback.

Once I understood the usage, the implementation did not take much time. There are common include files for file backup purposes, and files for backup/rollback can be defined in a list common_backup_file_list and these include files will loop over it.

Today we had a short session where a member from another team that is also responsible for PC development introduced us to the Virtual Network Functions (VNFs) Booking Tool. The VNF Booking application is a simple booking solution, that makes it easier for people working on the VNF nodes to manage their timetables. VNFs are software applications that deliver network functions. Common VNFs include virtualized routers, firewalls, and network address translation (NAT) services, and they are run in virtual machines (VMs) on virtualization infrastructure software such as VMWare and Linux KVM. (Red Hat, 2021d)

Why is the existence of this kind of tool important for us? Even though this tool is relying on the goodwill and attention of people who are working on the VNF machines, and it is still under development, the main purpose of this tool is to prevent interference of working time and different VNFs. Booking the VNF will not prevent others from logging in or possibly restarting the same VNF, it is just a helper tool at this point. All other information, like IP address, username, password, etc. about VNFs installed in the cloud are listed for each node (VNFs) on the internal Wiki page.

Friday

Today's objective is to continue with rollback implementation for the rest of the controls. During the daily meeting, it was suggested by the SM to work together with a new member of our team and that this can benefit both if we share work and spend more time on testing.

So, not long after daily, I had a long call with a new member of our team. After the scripts have been carefully analyzed, we divided them into logical groups, and based on that user stories will be created by SM. Most of the Linux files were straightforward, but we spent more time on NETCONF files. We concluded that 2 of the 3 NETCONF scripts do not have an implemented configuration, so restore is not supported, and one of the scripts, Lockout

duration already has an implemented ESM side backup and rollback. The only task is to update the attribute details of the script to indicate a value-based rollback. But just to be sure, since we are both relatively new, we agreed to bring up it at the daily meeting next week.

In the meanwhile, I had a short call with SM where we agreed on which scripts will be involved in each User Story, and also a group call where one of the most experienced developers explained in more detail the new rollback feature. The aim was to have precise information on rollback functionality, what were problems with the previous rollback, and what kind of improvement the customers could get. As I mention there is already a Wiki page made by PC developers, but this is more on a general level and during the long call now more of us have a much clearer picture of what needs to be done and why.

The rest of the day I spent working on the ssh scripts where I implemented the rollback by including ansible shared files that are doing file-based backup and rollback for Linux-based asset types.

Weekly analysis

This is the last reporting week for the diary thesis. The first observation week was just after the previous Release cycle, and I will finish the reporting period with the new PI and with the first sprint in the release cycle. I can say that this week was a little bit frustrating for me. Having in mind that I was on sick leave a whole previous week and that things were not going according to plan, I needed to find a way to summarize the new information and use that information for the future. I learned to adapt to the situation and find a way to take the next productive step and focus on the ongoing and future work.

Looking back at the goals I can say that I progressed well on the new tickets for rollback implementation. The progress comes from the much better collaboration between me and some teammates. The information was shared during discussions that I had with the group and separately with my colleagues. Since now more of us working on a new feature called “shared backup/rollback” it was important to make sure that everybody involved is on the same page. As I mentioned before, the formerly used script_attributes have been extended to contain the extra information for this shared backup/rollback feature. Each time when someone updates the relevant part of script_attributes of control scripts, regeneration of policy_config.yml is needed and that must be done during the policy catalog build process.

I can say that my communication skill improved during this week. I struggled with communication which sometimes led to frustrations. I think that my main problem was how to define the right question since I am still unfamiliar with technical terms and sometimes, I find it difficult to express my thoughts and my findings. The reason for that is mainly working alone on tasks that did not involve others. I felt more secure asking questions to resolve some doubts that I had. So, better communication showed results, even for some of us was time-consuming but the things learned and shared brought me a better understanding of the tasks.

This week I dedicate time to analysing the way of working in my team with theoretical. There is a new team setup that involves a little bit different way of working and planning. What I meant is that now we have an official meeting for sprint planning that occurs every 2 weeks as a working session. One of the main questions for me is the purpose of the Program Increment (PI) planning and when it happens within the Agile methodology. But first I will start with point out the difference between PI planning and Sprint planning.

There was a talk during this week about sprint planning and PI planning, and while I am familiar with the main concept and the purpose of sprint planning there are still some questions open for PI planning. Therefore, sprint planning is happening among the team members, and it is a session where the team discusses ongoing tasks, problems if any, and plan for completing product backlog items for the next sprint, while my knowledge of PI planning is limited. A PI happens between teams that share work on the same product/project. A PI is typically 8 -12 weeks long, in our case 8 weeks (Figure 3).

Based on my research PI planning concept comes from the Scaled Agile Framework (SAF). SCF is a registered trademark of Scaled Agile, Inc and it is a “knowledge base of proven, integrated principles, and practices for achieving business agility using Lean, Agile, and DevOps.” (Leffinwell, 2021) Given that neither images nor text can be copied from this site without the written permission of the copyright holder, my research on this topic will be limited due to providing the right source and I will summarize the main findings based on gathered information and the company’s practice.

The SAF is a set of practices and workflow patterns that guides organizations seeking to practice Agile at an enterprise scale. SAF helps development teams tackle the challenges of coordinating multiple teams, processes, and programs to deliver a unified product. (Lynn, n.d.)

PI Planning is scheduled at the beginning of each PI. The common session for PI planning is split into two days with one week between sessions. Each team has its session during the week between two common sessions – Team Breakouts. The planning preparations start when POs attend backlog refinements by being aware of what epics are prioritized for the next release cycle. Epics need to be sorted per their priorities as much as possible. The important thing here is to consider the team's velocity, for example, is someone going to be on vacation or is there a public holiday ahead that could affect velocity. The main points for consideration are:

- Plan user stories for the three sprints.
- Clarify priorities.
- Identify dependencies and needs towards other teams and communicate findings in the final plan review.
- Identify risks, and document them in Wiki under the team's title.

Later, PO and SM together with the development team estimate user story points and assign tickets to team members. For user story estimation the team needs to agree on what simple tasks look like and set its value to 1 User Story Point. As a reference, the Fibonacci scale is used. The Fibonacci scale gives agile teams a realistic way to approach estimates using story points that are used to represent the complexity and effort needed for completing a task (Lucid Content Team, n.d.). Because the Fibonacci numbers form a sequence, where is each number the sum of the two preceding ones, so numbers are not linear, in this way, each number in the Fibonacci scale is much larger than the previous number and it makes it much easier to recognize the difficulty and differences of each user story.

Stories	Software Engineer estimate	Product Owner estimate	SM estimate (Story Points)
User Story 1	This Jira can take a day or two	X-Small (1-2 days)	1
User Story 2	This Jira can take up few days , may be two such stories in a week	Small (2-3 days)	3
User Story 3	It can take up almost whole week	Medium (3-5 days)	5
User Story 4	This will take more than a week but not whole sprint	Large (5-7 days)	8
User Story 5	This will take up the whole Sprint	X-Large (7-10 days)	13
User Story 6	This will take more than one Sprint but less than two	XX-Large (10-15 days)	21

Figure 12. Fibonacci numbers and mapping to estimate User Stories (Pravesh, 2021)

The figure above (Figure 12) suggests ways of estimation that can be mapped to story points. The story points go (SP) from extra small to extra-large. For example, SP 1 means extra small. In this case, developers feel they understand requirements and consider the user story to be relatively easy and most likely can be completed in one to two days, while SP 13 means that task execution will take maybe two weeks or more and is going to take probably more than one developer to complete the task. It is important to understand that using SP has nothing to do with the developer's performance. It is based more on knowing the team's velocity and risks such as bugs appearing, as well as being aware level of knowledge and experience for the specific task.

During the second common session, the plan is reviewed. Teams present key outputs, such as objectives for the next PI, potential risks, and dependencies if any. The main outcome of the PI common session is to determine how successful each team was in completing the planned content from the last PI cycle. In this way, everyone has the opportunity to give their own opinion by marking the level of satisfaction on a scale of 1-5. If the score is too low, planning will continue.

PI planning requires good preparation and communication. The main actors are POs, management, and agile teams. PI Planning helps teams that share work on the same product/project to establish much better communication and collaboration by defining common goals and objectives. "The goal of PI Planning is to get all your teams aligned strategically and enable cross-team collaboration to avoid potential problems" (Blake, 2022)

4 Discussion and conclusions

Every journey begins with challenges, and this one was not an exception. Starting an internship in an array completely different compared to my study, as a DevOps engineer, was very exciting, full of expectations and wishes to learn as much as possible and solve as many as possible tasks in my first IT job. At the time I started the internship I can say that I was an absolute beginner in the world of Linux, Cloud, Virtual Machines, Ansible, Source Control Management, and Security, where the diary thesis was extremely useful to clear my mind, recapitulate and systemize accumulated knowledge.

The first thing to learn was the Ericsson Security Manager, where, why, and how it is used. As the product is very big, used in 5G network solutions, and contains many thousands of code lines, I started with product documentation, to get the big picture about it. The next area of my focus was the subproject in which I am participating, Policy Catalog. From the technology point of view, the first thing learned was about Virtual Machines, Ansible, and Linux pam files. For this purpose, I installed Virtual Box on my computer, popular virtualization software from Oracle. Using it, I have made three virtual machines, one used as Ansible Controller, and the other two as ansible hosts. Ansible controller contains playbooks from where a play will be started, and hosts are machines where tasks from playbook will be executed. This setup I am still using for learning and trying purposes whenever I need to test the new Ansible module and to see how it works. Learning about small modules in a big project is much harder, as they are part of the bigger system. Experimenting is not possible as I worked with real servers which always need to be in a ‘healthy’ state as it is used by many developers.

For job purposes, as I got a Windows machine from the company, and the source code is done for the Linux platform, I had to learn too about WSL – Windows Subsystem for Linux, which is a kind of no full virtual machine. It has fewer capabilities than a fully flagged Virtual Machine with Linux as a guest, but it does not require a lot of resources from the host machine, and it is much faster. So, for my purpose where I will have only source code in a Linux machine, running a few scripts, ssh-ing to remote servers, and will run some other smaller tasks, this was an ideal solution. Development Environment (Visual Studio Code) will be in Windows and will have an already available extension for accessing source code in Linux.

Step by step I was learning many things: bash shell, as one variant of Linux shell, bash commands, which I am learning even today, pam files as the way how Linux is doing configuration for user authentication, general thinking for the first time about security of a

computer. Writing a diary which looks like extra duty at the beginning, actually shows immediately to be very useful to make learning wider and more studious and to help me to finish successfully my real tasks. Very soon learning about source control management was necessary, as I needed to submit my first task. Git and Gerrit were looking complicated at the beginning, especially Git which possibilities I am learning even today, I recognized soon all the advantages which they bring.

Learning about how to secure communication between two network nodes (computers) using SSH and encryption with the private and public key, how authentication is working in Linux, and how to configure many parameters used in those processes deeper my knowledge in the security area.

What I think will be very useful in my future career is learning about CI/CD and how it is done using Jenkins. As every modern software project has to include this knowledge to make it stable, organized, and automated, the usage of this technology is excepted just to grow in the future.

It was not easy to follow the everyday routine, sometimes to do a task requires full concentration and extra work, so dairy has to be done a little bit later. Overall, looking backward, the most important things I learned in this internship is Linux OS, the modern way to automate tasks in the cloud using Ansible, SCM, the way how CI/CD is working today, and how complex product is done with many people involved in developing, testing, release, and LCM (Life Cycle Management). How to organize so many people, manage them, how Agile in reality is functioning. Many things are of course different in practice than in theory and specific to different organizations and teams.

What I would do a little bit differently, is to put more effort into learning the tools and languages at the beginning and less focus to implement the task, but it is always some kind of compromise and depends on the real project and team situation.

Doing the thesis at the same time as working in a big international company was a good decision, even if I had situations where a found myself overwhelmed. During the observation period, I needed to find a way to overcome some challenges. Facing challenges and learning how to appreciate them in a way to get a positive experience from them was one of the many lessons learned.

One of the biggest challenges for me was time management. I was working full time, being in front of the screen all day, sitting at home, and then needing to open my computer once

again and start writing for my thesis. In the beginning, it was a very big problem for me. I ended up writing everything in two days, and usually during the weekend. Of course, I was making notes every day for the daily entries, but this way of working was very exhausting, especially not having even one free day to just relax with my family. Therefore, I decided to change my way of reporting and working. I started to write the report every day for that day and learned to add comments and citations for the weekly analysis as I write, rather than as part of the final step in the process.

Changing the mentality of being a student into being an employee was also a challenge I had to overcome. There is always a possibility that the work done by me will be changed later. Someone else will do some additional modifications to the code based on the changing requirements.

The skills I have gained and the knowledge I am taking with me are very important for my further development and growth in professional life. The most important aspects of my daily work during the time I spent writing a diary-based thesis I tried to summarize in this chapter.

As a result of my work on the thesis, I can categorize the development process into three phases. The first phase was at the very beginning when it was more about self-learning and learning about the technologies and tools used in daily work. The second phase was when I started working on the more complex tasks requiring much more collaboration with others. It wasn't enough just to browse the Wiki pages and the Internet. It was necessary to turn to more experienced co-workers and seek advice and opinion. The last phase can be described as the stage of the development process when I felt confident enough to express an opinion and raise questions without fear of sounding "unprofessional". Perhaps one of the most important things I had to learn at work is how to work with people I had never met in person, with different cultural backgrounds and previous work experiences.

In the end, I would like to highlight the importance of soft skills. From my perspective, they are skills that everyone needs to excel in the workplace. Therefore adaptability, positivity, good communication, and collaboration are skills necessary to develop yourself in a professional as well as personal life. Anyone can learn hard skills but learning how to communicate with others and being part of a team is something that requires an open mind and motivation to be part of something bigger than you.

References

Ansible. (n.d.). *Ansible Documentation*. Retrieved from docs.ansible: https://docs.ansible.com/ansible/latest/

Atlassian. (n.d. a). *What is Jira used for?* Retrieved January 2022, from Atlassian: https://www.atlassian.com/software/jira/guides/use-cases/what-is-jira-used-for

Atlassian. (n.d. b). *Git Cherry Pick*. Retrieved March 2022, from Atlassian.com: https://www.atlassian.com/git/tutorials/cherry-pick

Baydan, I. (2020, December 27). *Linux /etc/login.defs Tutorial*. Retrieved March 2022, from LinuxTect: https://linuxtect.com/linux-etc-login-defs-tutorial/

Blake, S. (2022, January 25). *The Ultimate Guide to PI Planning [2022 SAFe Edition]*. Retrieved April 21, 2022, from EasyAgile: https://www.easyagile.com/blog/the-ultimate-guide-to-pi-planning-2020-safe-edition/#:~:text=So%2C%20the%20goal%20of%20PI,to%20avoid%20these%20potential%20problems.

BMC Blogs. (2016, December 22). *ITIL release and deployment management*. Retrieved April 2022, from BMC: https://www.bmc.com/blogs/itil-release-deployment-management/#:~:text=According%20to%20ITIL%2C%20the%20purpose,the%20integrity%20of%20existing%20services.

Chacon, S., & Straub, B. (2014). *Pro Git* (2nd Edition ed.). Apress. Retrieved March 2022, from https://git-scm.com/book/en/v2/Appendix-C%3A-Git-Commands-Getting-and-Creating-Projects

Codingpointer.com. (n.d.). *yq Command-line YAML processor*. Retrieved March 2022, from Codingpointer.com: https://www.codingpointer.com/linux-tutorial/yq-yaml-processor

die.net. (n.d.). *login.defs(5) - Linux man page.* Retrieved March 2022, from die.net: https://linux.die.net/man/5/login.defs

DNSstuff. (2020, April 20). *What Is an Audit Log? Audit Trails and How to Use Audit Logs*. Retrieved March 2022, from DNSstuff: https://www.dnsstuff.com/what-is-audit-log#:~:text=An%20audit%20log%2C%20also%20called,activities%20or%20a%20specific%20activity.

Drumond, C. (n.d.). *What is Scrum?* Retrieved January 2022, from atlassian.com: https://www.atlassian.com/agile/scrum

Ericsson. (n.d.). *Security Manager*. Retrieved January 2022, from ericsson.com: https://www.ericsson.com/en/portfolio/iot-and-new-business/new-business/security-and-risk-management/security-manager

Fernando, J. (2021, August 19). *Stakeholder*. Retrieved February 2022, from Investopedia: https://www.investopedia.com/terms/s/stakeholder.asp

GeeksforGeeks. (2021a, January 18). *Version and Release Management in Software Engineering*. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/version-and-release-management-in-software-engineering/

GeeksforGeeks. (2021b, August 21). *Introduction to Semantic Versioning*. Retrieved April 2022, from GeeksforGeeks: https://www.geeksforgeeks.org/introduction-semantic-versioning/

gerrit-review.googlesource. (n.d.). *How Gerrit Works.* Retrieved February 2022, from gerrit-review.googlesource.com: https://gerrit-review.googlesource.com/Documentation/intro-how-gerrit-works.html

Hahn, P. (2021, January 14). *Linux audit system.* Retrieved March 2022, from https://pmhahn.github.io/: https://pmhahn.github.io/audit/

Hamilton, T. (2022, February 26). *What is Exploratory Testing? Techniques with Examples.* Retrieved March 2022, from Guru99: https://www.guru99.com/exploratory-testing.html

InformIT. (2001, March 23). *Making the Most of Pluggable Authentication Modules (PAM).* Retrieved February 2022, from InformIT: https://www.informit.com/articles/article.aspx?p=20968

Jenkins. (n.d.). *Getting Started with Pipelines*. Retrieved April 2022, from Jenkins: https://www.jenkins.io/pipeline/getting-started-pipelines/

Knowledgehut. (n.d.). *Product Increment*. Retrieved March 2022, from knowledgehut.com: https://www.knowledgehut.com/tutorials/scrum-tutorial/product-increment

Leffinwell, D. (2021, February 10). *Welcome to Scaled Agile Framework® 5!* Retrieved April 20, 2022, from Scaled Agile: https://www.scaledagileframework.com/about/

linuxCommand.org. (2020, April 17). *ssh — OpenSSH remote login client*. Retrieved February 2020, from LinuxCommand.org: https://linuxcommand.org/lc3_man_pages/ssh1.html

Lucid Content Team. (n.d.). *Using the Fibonacci scale in Agile estimation*. Retrieved April 20, 2022, from Lucidchart: https://www.lucidchart.com/blog/fibonacci-scale-for-agile-estimation

Lynn, R. (n.d.). *Program Increment Planning (PI Planning)*. Retrieved April 2022, from Planview: https://www.planview.com/resources/guide/scaled-agile-framework-how-technology-enables-agility/program-increment-planning/

Mahmood, A. (2011, Octobar 27). *Basics Of Minilink Microwave Networks.* Retrieved February 2022, from SlideShare: https://www.slideshare.net/atifmahmood12/basics-of-minilink-microwave-networks

McAlister, J. (2015). In J. McAlister, *Mastering Jenkins.* Packt Publishing Ltd. Retrieved April 2022

Microsoft. (n.d.). *Windows Subsystem for Linux Documentation.* Retrieved February 2022, from docs.microsoft.com: https://docs.microsoft.com/en-us/windows/wsl/

Mraz, T. (n.d.). *pwquality.conf(5) - Linux man page.* Retrieved February 2022, from linux.die.net: https://linux.die.net/man/5/pwquality.conf

NOKIA. (n.d.). *NETCONF.* Retrieved March 2022, from infocenter.nokia: https://infocenter.nokia.com/public/7210SAS209R1A/index.jsp?topic=%2Fcom.sas.system.m%2Fhtml%2Ftsr_netconfig.html

Pravesh. (2021, May 1). *Story Point Estimation for Techies!* Retrieved April 20, 2022, from Dev Genius : https://blog.devgenius.io/getting-started-with-story-point-estimation-ac58b4eac499

Red Hat. (2018c, January 31). *What is CI/CD?* Retrieved April 2022, from Red Hat: https://www.redhat.com/en/topics/devops/what-is-ci-cd

Red Hat. (2021d, May 7). *VNF and CNF, what's the difference?* Retrieved April 19, 2022, from Red Hat: https://www.redhat.com/en/topics/cloud-native-apps/vnf-and-cnf-whats-the-difference

Red Hat. (n.d. a). *CHAPTER 2. USING PLUGGABLE AUTHENTICATION MODULES (PAM)*. Retrieved January 2022, from RedHat: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/managing_smart_cards/pluggable_authentication_modules#About_PAM

Red Hat. (n.d. b). *AUDIT RECORD TYPES*. Retrieved March 2022, from Red Hat: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security_guide/sec-audit_record_types

Schwaber, K., & Sutherland, J. (2020, November). *scrumguides.org.* Retrieved from The Scrum Guide: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf#zoom=100

semver.org. (n.d.). *Semantic Versioning 2.0.0*. Retrieved April 2022, from semver.org: https://semver.org/

Shetty, R. (2020, September 18). *Best Jenkins Pipeline Tutorial For Beginners*. Retrieved April 10, 2022, from LAMBDATEST: https://www.lambdatest.com/blog/jenkins-pipeline-tutorial/

Simplilearn. (2021, Octobar 28). *Understanding the Release Management Process: The Secret to a Successful Software Release*. Retrieved April 2022, from Simplilearn: https://www.simplilearn.com/release-management-process-article

Tail-f. (n.d.). *NETCONF OVERVIEW*. Retrieved March 2022, from Tail-f: https://www.tail-f.com/what-is-netconf/

TBG Security. (2018, March 28). *What is an information security framework and why do I need one?* Retrieved January 2022, from techslang.com: https://tbgsecurity.com/what-is-an-information-security-framework-and-why-do-i-need-one/

Testim. (2020, December 17). *UI Testing: A Beginner's Guide With Checklist and Examples*. Retrieved March 2022, from testim.io: https://www.testim.io/blog/ui-testing-beginners-guide/

Weidner, K. (2004, Jun 29). *EAL3 Evaluated Configuration Guide for Red Hat Enterprise.* Retrieved February 2022, from SourceForge : http://ltp.sourceforge.net/docs/RHEL-EAL3-Configuration-Guide.pdf

What Is Exploratory Testing In Software Testing. (n.d.). Retrieved March 2022, from Software Testing Help: https://www.softwaretestinghelp.com/what-is-exploratory-testing/