Kristian Sigtbakken Holm
Martin Kvalvåg
Nikolai Fauskrud
Olav Henrik Hoggen

Automated Malware Analysis Platform

Bachelor's project in IT-Operations and Information Security
Supervisor: Basel Katt
May 2019

Bachelor's project

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Information Security and Communication Technology

NTNU
Norwegian University of
Science and Technology

NTNU

Norwegian University of
Science and Technology

Automated Malware Analysis Platform

Author(s)

Kristian Sigtbakken Holm
Martin Kvalvåg
Nikolai Fauskrud
Olav Henrik Hoggen

Bachelor in IT-Operations and Information Security
20 ECTS
Department of Information Security and Communication Technology
Norwegian University of Science and Technology,

20.05.2019

Supervisor Basel Katt

Sammendrag av Bacheloroppgaven

Tittel:	**Automatisert Skadevareanalyse-plattform**
Dato:	20.05.2019
Deltakere:	Kristian Sigtbakken Holm Martin Kvalvåg Nikolai Fauskrud Olav Henrik Hoggen
Veiledere:	Basel Katt
Oppdragsgiver:	Kongsberg Defence & Aerospace
Kontaktperson:	David Lee Andersen, david.lee.andersen@kongsberg.com, 48227979
Nøkkelord:	Innholdsanalyse, Antivirus, Infrastruktur som kode, Virus
Antall sider:	86
Antall vedlegg:	13
Tilgjengelighet:	Åpen

Sammendrag:

Alle bedrifter trenger en måte å sikre virksomheten deres, og i dagens digitale tidsalder er dette spesielt fokusert mot å beskytte digitale systemer mot skadelige og ondsinnede entiteter. Denne beskyttelsen blir i dag ofte betjent av ett enkelt antivirus i det private domenet. Hva om flere antivirus og andre analyse verktøy ble samlet inn i en spesialisert beskyttelse og analyse plattform? Det er ideen bak "open souce", "self-hosted", VirusTotal alternatives: IRMA. IRMA mottar usikre filer, sjekker dem mot et sett av analyse verktøy og forteller oss om de er trygge eller ikke. Vår oppgave var å undersøke denne løsningen, tilpasse den, og følge "open-source" mantraet, slik at den blir tilgjengelig for oppsett for enhver interessert aktør, for selskaper eller for privatpersoner. Sluttresultatet ble et modifisert IRMA system for vår infrastruktur med en blanding av andre og våre egne analyseverktøy modifisert for IRMA.

Summary of Graduate Project

Title:	**Automated Malware Analysis Platform**
Date:	20.05.2019
Authors:	Kristian Sigtbakken Holm Martin Kvalvåg Nikolai Fauskrud Olav Henrik Hoggen
Supervisor:	Basel Katt
Employer:	Kongsberg Defence & Aerospace
Contact Person:	David Lee Andersen, david.lee.andersen@kongsberg.com, 48227979
Keywords:	Malware, Antivirus, Infrastructure as Code, Analysis
Pages:	86
Attachments:	13
Availability:	Open

Abstract:

All businesses need to ensure their security, and in today's digital age, this need stretches to protecting digital systems against harmful and malicious parties. This protection is often served by a single antivirus in the private domain. What if multiple antiviruses and other analysers were aggregated into a specialised protection- and analysis platform? That is the idea behind the open source, self-hosted VirusTotal alternative: IRMA. IRMA receives insecure files, checks them up against a set of analysers, and tells us whether the files are safe or not. Our task was to research this solution, adapt it, and follow the open source mantra, making it available for setup and usage for any interested actor, be it business or home. The end result was an openly available, modified IRMA with extended security functionality through a variety of analysers, some of which custom-added.

Foreword

Our inspiration to choose this assignment was because we wanted a challenging project with both security and infrastructure aspects. After four months of development we got both a fun and a quite challenging project.

We would like to thank Kongsberg for providing us with all the guidance we could ask for, assistance, and always being available to bounce ideas off of and discuss. We would especially like to thank our contact person and product owner, David Lee Andersen. We would also like to thank Basel Katt for guiding us during the project, referring us to multiple practical sources for information, and challenging our view of the project.

We would like to thank our families for constant support through the whole academic ordeal.

And finally we would like to thank each member of the bachelor group for working hard throughout the project, for compromising when necessary for progress, and rising above conflicts, and for carrying each other the whole way.

Contents

List of Figures

List of Tables

Glossary

VCS Version Control System - A tool used for tracking changes to code

ICAP server is a server which can be queried through the ICAP protocol which is a protocol designed to off-load some Internet-based content to dedicated servers.

Automation fear is an inherent distrust for automation in Infrastructure, with lack of testing as a cause.

Cloudflare is a content delivery platform which additionally helps mitigate DDOS attacks.

WinRM is a windows connector made for remote management, somewhat similar to SSH.

Executable is a term used in the report about files that can execute, this includes any files that contain executable components.

Bottleneck a part of the code or system that severely limits the efficiency of the rest.

Active Directory is a directory service that allows administrators to store and manage resources on the network.

A [Play] is when a playbook is executed.

Playbook is a recipe in Ansible for how the configuration should be done.

Roles is a method in Ansible for modularising code. These modules provisions for one service, and multiple roles are combined to provision a larger service.

Probe refers to either a machine for hosting analysers or an analyser.

Host is a machine from which something is run or is running on.

Configuration drift is a concept where servers slowly over time being configured differently, mostly due to ad-hoc fixes. Leads to snowflake servers.

Snowflake server is a concept a server that have gone through numerous ad-hoc fixes which are not documented or easily reproducible.

1 Introduction

1.1 Problem area, delimitation, and defining the assignment

The transfer of unknown, insecure files is a constant challenge between employer and supplier. To ensure the safety of external files they need to be scanned for malicious content before being transferred to the internal infrastructure. Solutions providing this as a service does exists, some examples are VirusTotal and Opswat MetaDefender. These were quickly ruled out based on features not compliant with our requirements. Kongsberg needs a solution to ensure that whatever is imported to their systems is safe and non-malicious, while not sharing confidential data with third-parties. Kongsberg requires an in-house solution to avoid entrusting their security to an unrelated third-party.

This project aims at providing an in-house solution using primarily open-source software where possible to cover the need for security. The system will primarily handle data from physical mediums connected with USB. This system will inherently be advanced, keeping it updated and easy to configure are significant challenges that needs to be met in design and final product. The final product needs to be easy to use, implement, setup, and must be low maintenance. The final product should be a "blueprint" solution for a system that detects malicious software or files. It should be a good starting point for further development, but require no or minimal user interaction during operations. Maintenance should be easy and need minimal work. An automated setup with good descriptions and explanations could be a fitting solution.

The project development starts 1. February 2019, and ends about 20. May 2019. We're limited to 4 developers with a short time frame, which restricts our capability to complete such a large project.

1.2 Purpose of the assignment

Kongsberg Gruppen is a Norwegian company supplying high-technology systems and solutions to maritime-, army-, aerospace-, and offshore oil and gas industries, both nationally and internationally. The company is split into multiple branches. One of which is Kongsberg Defence and Aerospace (KDA), the specific party responsible for handing out this assignment[1]

On a daily basis, KDA handles large amounts of confidential data transferred from third parties. The assignment arises from no previous effective solution in place to scan incoming data. Every business needs effective and secure solutions. Hence, their current approach of designated "dishwasher" machines running one single anti-virus with no supporting infrastructure, while lacking integration with their other systems, is insufficient.

KDA wants a dynamic system, capable of scanning files and discovering potential threats in a complex environment. One typical use case is plugging a flash drive into company kiosk hardware, automatically scanning all the data in an analytic system after a company authentication. This analytic system should contain a wide array of automated

[1] Kongsberg Gruppen:https://www.kongsberg.com

analysis tools. The wide array of tools should ascertain the security of files with a high degree of certainty. In the end this generates a report with scan results. The generated report is relayed back to the user initiating the scan, and is also stored in a logging platform for any further analysis deemed necessary in the future.

The purpose is automation of menial, manual tasks. The result will be less repeating tasks for employees ensuring company security.

1.3 Target audience

This report is aimed at people expected to have a good grasp of general IT concepts, e.g. students ready to start a bachelor project, professors, IT workers, e.g. KDA. With this in mind, general concepts will not be explained in further details, but system specifics, related concepts and ruminations will be explained.

1.4 Students' backgrounds and qualifications

We are a group of 4 bachelor students in IT-Operations and Information Security at the Norwegian University of Science and Technology (NTNU) on our final year.

1.4.1 Nikolai Fauskrud

Scrum master, developer: Nikolai is a 21 year old student straight out of a high school in Lillehammer where he studied general studies. His work experience includes SlettMeg where he works alongside the bachelor thesis.

1.4.2 Kristian S. Holm

Developer: A 22 year old student of BITSEC, has taken multiple extra curricular classes, most of which in the programming field, while volunteering for reference group responsibilities in multiple classes. Commands a wide field of knowledge. Like the rest of the group, capable of handling most aspects of this project.

1.4.3 Olav H. Hoggen

Developer: Olav is a 23 year student studying operations and security. Has been involved in student volunteering and has had a position as board member in "Studentenes Hus Gjøvik". Has the general knowledge needed to complete the bachelor project, but not as much infrastructure as code experience as the other group members and thus had to spend some time at the start of the project learning infrastructure tools.

1.4.4 Martin Kvalvåg

Developer: Martin is a 24 year old IT operations and information security student from Fredrikstad. In his free time he has tinkered a bit with self hosting different services as an option to commercially available services. He has also tinkered with a couple of simple Raspberry pi projects. Both cases using infrastructure as tools and containerisation. This combined with a couple of previous courses related to infrastructure and a passion for information security makes this project both exciting and realistic to execute with a good final product.

Most of us have some experience with infrastructure tools such as Heat and Puppet. While these were considered, they did not end up being used. Previously we mostly had theoretical knowledge and we had to learn practical parts necessary e.g. Vagrant, Packer and Ansible. Therefore large quantities of time went into learning specifics of the selected technologies.

1.5 Project scope

1.5.1 Field of study

Malware detection

We will look at and evaluate different technologies, frameworks and tools used to detect malware. The goal is to evaluate the different options, their pros and cons and choose the best fitted options. We will also explain why we believe these best meets the requirements.

Automation and orchestration

Automation should be a priority for any developer today. In a world with an increasing amount of available tools, platforms, and interfaces for handling the abstraction of configuration and setup, automation is a critical part of time and software management. We will evaluate different automation and orchestration tools such as Terraform, Chef, Puppet, Ansible etc. The chosen tools will be based on functionality and performance. Previous knowledge of group members and preferences from KDA if any will also be taken into consideration.

Scalability

None of the members of the group have any previous experience with development of scalable programs. We cannot determine with certainty whether scalability is realistic to get implemented properly alongside the rest of the project within our given time frame. If we deem scalability out of reach for the current project we will do our best to facilitate further development.

Logging

Logging every action done upon the supplied files are necessary for the system to be used in classified or business settings handling sensitive files. It is therefore necessary to evaluate different means of logging actions done upon the given files. Some options could be to use extended system logs using AppArmor[2]/SELinux[3] or internal logging in the tools we decide to use.

1.5.2 Delimitations

Scalability

Scalability is an unknown area for all the group members and is not the main focus of this project. Because of that it will have a secondary priority and might only be an evaluation and not explicitly implemented.

Web and email delivery

Having a custom web interface or an email that any user inside the organisation can access and submit files to is a great and useful feature but is not within the main scope of this project. Considering IRMA, Cuckoo and many of the tools we will evaluate already

[2] AppArmor: https://wiki.ubuntu.com/AppArmor

[3] SELinux https://en.wikipedia.org/wiki/Security-Enhanced_Linux

have a web interface for submitting files, it's should be fairly easy to create a collective web interface for submission of files. Though it is not something we know the complexity of at the current time, neither do we know the requirements to implement it as an email service. Therefore this is a secondary priority and will be evaluated, but might not be implemented.

Summary report

An automatic collective and summarised report of all analyses done is a useful feature, it is also something that might be very time consuming to develop. It is our opinion that a well functioning system with high accuracy is more important and valuable than a system with low accuracy and a beautiful report. A simple collective report is necessary, but a detailed and highly customised report has a lower priority than a good functioning system.

Detection Rules

Malware can be very complicated and intricate. The main focus of this project will be to develop a system that can use detection rules as a way of detecting malware. It is not to create new and custom rules for known or custom malware. That is not to say we won't implement any but we will mainly use existing publicly available detection rules.

1.5.3 Project description

A composite system that scans and analyses a given file or a set of files. The first and primary way of delivery will be by automatic retrieval of the files from newly connected storage media. Secondary ways of delivery is wanted and planned but given the uncertainty of the project size these are not the primary focus. These include, but are not limited to: a web page for uploading suspicious files of unknown origins, delivery by email with files as an attachment, and more.

The finished system should handle the files internally and not share specifics of the files or its behaviour with any external entities. It should analyse each batch of files separately, thus minimising the chance of cross contamination, and the user should end up with a report of the results.

1.6 Other roles

Product owner (KDA):	David Lee Andersen david.lee.andersen@kongsberg.com
Position:	Security Analyst
Involvement:	Product owner
Employer (KDA):	Thomas R. Andersen thomas.rivrud.andersen@kongsberg.com
Position:	IT engineer
Involvement:	Occasional
Counsellor:	Basel Katt basel.katt@ntnu.no
Position:	Associate Professor at NTNU
Involvement:	Supervisor

1.7 Project process

1.7.1 Central project process

Scrum has been chosen and adapted due to its inherent fit to handle complex projects in an organised fashion, and the ability to keep a constant dialogue between us and the product owner for discussing all manner of details. Bearing in mind the size, complexity and openness of the assignment, it is deemed beneficial to continually challenge our vision and the one of the product owner for different alternative implementations. We aim to continually utilise the experiences from the product owner and our NTNU supervisor.

1.7.2 Theory

A scrum team consists of three roles: the developers, scrum master, and the product owner. The process is based on sprints, with meetings for sprint review, sprint planning, and daily sprint meetings[4].

- The sprint planning meeting is where the scrum team meets to plan what to prioritise for the coming sprint.
- The sprint review is where the scrum team meets to discuss the previous sprint; what was accomplished, what went wrong, and what needs to be re-prioritised.
- Sprint meetings are daily fifteen minute meetings between the developers to discuss how progress is going, what they plan to do, or whether they have found some obstacles.
- The developers are the students responsible for this bachelor. Our scrum master was elected to be the member of the team responsible for communicating with KDA, our employer, who has agreed to act as product owner.

1.7.3 Plan for status meetings and decisions

There will be bi-weekly meetings with our NTNU supervisor with the opportunity to arrange weekly meetings when it's deemed necessary. All of these are initially set to Fridays.

Our sprints have been decided to have a duration of two weeks, starting from the sprint planning with all relevant actors, and ending with the sprint review. In between there are daily sprint meetings.

Scrum sprints will start Fridays with sprint planning and go on for two weeks until the next Friday sprint review, immediately followed by the next sprint planning meeting.

In addition to the short scrum meetings, the developers will meet up weekly to discuss and work in tandem.

Reports will be actively written to constantly document findings, decisions, considerations done, and other significant events.

1.7.4 Tools used

We have used some different planning and productivity tools during the project and will in this section go through the most significant of these.

[4] Scrum: https://en.wikipedia.org/wiki/Scrum_(software_development)

Toggl

Toggl is a logging tool used to log time spent. In the bachelor assignment we have used it both to fulfil the logging requirement of time spent. Furthermore, it is used to document time spent on each task for our sake. Each group member has been responsible for their own individual logging.

Trello

Trello is a productivity tool for organising tasks into what stages of development. We utilised boards for the following stages: product backlog, sprint backlog, in progress, review/testing, and task done. These boards were used to organise our Scrum sprints and backlogs. Tasks agreed upon at Sprint planning meetings were taken from product backlog to sprint backlog, further on claimed by a developer and moved to in progress. When finished, the task would be marked for review. This led to a self-organised, organic work-flow where essential tasks were picked up and worked on, and minimal collision between different developers.

Google Drive

Google Drive was used to store files and notes covering all aspects of the development, from meeting summaries to documenting experiences and challenges in development, design drafts, and more. The only files that did not end up on Google Drive were anything containing confidential data, and code that belongs in a dedicated version control system(VCS).

GitHub

GitHub is a widely used online platform for project collaboration. Using GitHub as our version control system, it can also act as remote storage for finished code. There are many administration tools on the platform to manage code, e.g. what is added, what is removed, whether it fulfils quality standards or not, etc. GitHub was used to host all our code except any sensitive test data that should stay private. Furthermore, all open-source software used in this project was already hosted on GitHub.

Overleaf

Overleaf is a collaborative editor for writing the typesetting language LaTeX. It was specifically used in writing the entirety of this report and the compilation of it. This saves a significant amount of time from setting up local LaTeX environments and streamlines cooperation.

1.8 Thesis structure

The team and the project is introduced in the first chapter, *Introduction*.
In the next chapter, *Requirements*, the requirements are specified e.g. through usage of use case diagrams and high level use cases.
These, introduction and requirements, lay the foundation for the next chapter which is *Theory and Technologies*, where requirements for specific technologies are set, theory is explained and technology is selected.
The next chapter, *Implementation* goes through how the technologies can be implemented, and how they are implemented, and adds some discussion around these im-

plementations. Sub sections to note within Implementation are *Configuration and set-up of IRMA* and *Configuration and set-up of Cuckoo*.

The next chapter is *Testing and Analysis*, which tests efficiency, checks for probe flaws and what the probes shares, and how accurate scans are.

The next chapter, *Results and Discussion*, summarises the results achieved through the implementation and testing. Additionally, it mentions our recommendations for the infrastructure and further work.

Lastly, the *Conclusion* summarises the whole project, what we have learned, an assessment of the project and summarises future work required.

2 Requirements

2.1 Initial requirements

The initial assignment outlined multiple requirements. Early during our dialogue with KDA these were asserted to be floating stepping stones to bounce ideas and discussion off of and to initiate the research phase. These are the initial requirements:

- Analysis
    - Static (header, certificate)
    - Known good / bad
    - Yara
    - Antivirus (several)
    - Dynamic analysis (network, IDS)
- Integrates with authentication solution and file systems
- Website for uploading files and view status
- Results via API, exported to a logging platform

KDA also specified certain "nice to haves":

- Solution should be highly automated.
- Scalability (10 000 - 20 000 users)
- Upload via email and web-site as well as API.
- Should cause little administration from KDA if put into operation.
- Should have tracking possibilities. [1]
- Should have the possibility of automatically generated reports.

See the section Scope 1.5 for requirements deemed possible within the project time frame, developer experience, and resources available.

[1] Tracking actions done upon the submitted files, by for example a dynamic analyser

2.1.1 Use Case

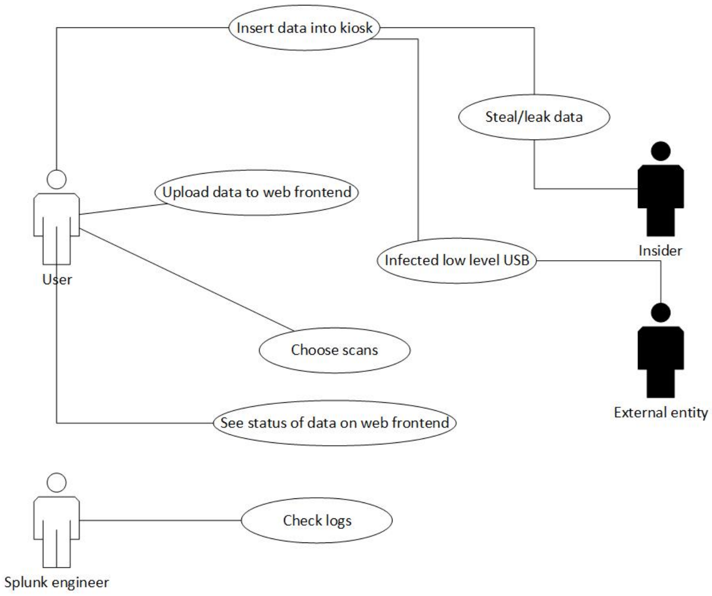


Figure 2.1: Use case and misuse case-diagram

2.1.2 High level use case

Use case	Insert data through kiosk
Actor	User
Description	A user inserts a USB stick or a hard drive into a kiosk. The kiosk will automatically detect the input, then asks user for authorisation through Active Directory. If authorised: upload files for scanning.

Table 2.1: Use case - Insert data through kiosk

Use case	Upload data to web front end
Actor	User
Description	A user can manually upload files to the web front end. It is not possible to get any files from the web frontend - scan results are available.

Table 2.2: Use cases - Upload data to web front end

Use case	Choose scan engines
Actor	User
Description	The user can explicitly decide which analysis engines to be used in their scan.

Table 2.3: Use cases - Choose scan engines

Use case	View scan status on web front end
Actor	User
Description	After a scan has finished the user can see a list of all the files that have been scanned. File name on the left and number of scanners on the right. Results are coloured red for malicious and green for safe. The user can also click on individual files for a more detailed look from each individual scanner.

Table 2.4: Use cases - View scan status on web front end

Misuse case	Steal/leak data
Actor	Insider
Description	A potential insider can be the biggest threat to the system. If a USB stick or a hard drive is left unattended, important and sensitive files can be stolen and/or leaked. It may also be possible for files and/or information to be stolen/leaked after a scan has been done.

Table 2.5: Misuse case - Steal/leak data

Misuse case	Infect low level USB drivers
Actor	Hacker
Description	A sophisticated malicious program could potentially inject malicious code on the kiosk USB driver. This would lead to the infection being too low level for an Antimalware program to easily discover, if at all. This could lead to every scan done from the kiosk to potentially be contaminated, or worse.

Table 2.6: Misuse cases - Infect low level USB drivers

Misuse case	Sophisticated malware
Actor	Hacker
Description	A sophisticated malicious program could pick up confidential Active Directory user credentials during authentication/authorisation phase and store this for later extraction.

Table 2.7: Misuse case - Sophisticated malware

Misuse case	Access to file storage
Actor	Insider/Administrator
Description	An administrator with access to the file storage, can extract potentially confidential or otherwise sensitive files.

Table 2.8: Misuse case: Access to file storage

Misuse case	Malware escaping VMs
Actor	Hacker
Description	Highly advanced malware can escape VMs and spread across the system, potentially gathering data for later extraction, or dealing damage to the infrastructure.

Table 2.9: Misuse case - Malware escaping VM

2.2 Functional requirements

The system must be able to register outside input, authenticate the input as sent by an authorised user. Afterwards, the system needs to send the input across an isolated network, then it needs to scan the input with several types of known good/bad, static, and lastly dynamic analysers. It must return a result from the analysis that shows whether the input was malicious or safe.

There should be logging to ensure the KDA security department can examine files at a later occasion if they are retroactively discovered to be suspicious. KDA especially wants to log files which have been altered or ran through dynamic scans and exactly what actions were done on the files. This logging is especially important when doing business with actors like the Norwegian Armed Forces. It is required to follow strict requirements to log any action to ensure repudiation and explicitly abide with the law.

2.3 Operational requirements

KDA requested that the system should not involve a high degree of administration from KDA. Once configured and deployed, it should operate with minimal human supervision. This implicitly means the system should be highly automated and easy to configure. The web front end for uploading files should be stable and be user friendly to a degree where regular users can intuitively use it to upload a relevant files for scanning. KDA should spend minimal time on troubleshooting.

2.4 External requirements

KDA does not want any scanned data to be shared with third parties and wants all of the data to stay in-house. This quickly closes the highly known VirusTotal out of the question. The data is highly confidential, meaning the security requirements are high. It is not an option to allow any data leaks. That means any transfer must be encrypted, and the system has to be as isolated as possible inside a secluded network with minimal connections to any other KDA network until all data has been approved by scans.

KDA operates in a way where they try to use and adapt open-source systems for their use cases to a high degree. That is why it's a preference for this project to use open source software wherever possible and feasible. If open source software is adapted into a complete system for free use by both KDA and other interested parties of any affiliation it is seen as an ideal outcome.

3 Theory and technology

3.1 Malware detection

Anti-virus giant, Norton[1], summarises malicious software as any software specifically designed to access, exploit, or damage computers. This software is commonly referred to as malware. Malware tends to act without the knowledge of the owner, but there are also exceptions to this where the malware can exist to scare a user into believing something is wrong, and in that way gain leverage on a user.

Malware programmers come from many backgrounds and with several motivations. Usual motivations include: profit, vandalism, information gathering, and many more. To reach these goals, the malicious actors have devised a multitude of malware types, including viruses, worms, Trojan horses, ransomware and more.

3.1.1 Static detection

Global Director of Threat Research at NTT Security, Jeremy Scott[2], writes that static detection is a way of determining whether a file is malicious or not from static, technical indicators of the file by using several tools and techniques. Technical indicators include file names, checksums or hashes, file types, and file sizes. These techniques allow scanning for any abnormalities without executing the files, further leading to lower risk of infection, but also having a considerable risk of false positives or negatives.

The main techniques that are used in static analysis are:

- Signature-based detection
- Heuristic-based detection

Signature-based detection will check static identifiers of files against enormous databases of known malware. This is the simplest and quickest way of doing malware detection. When new or previously unknown malware is discovered, its signature is added to one or multiple malware signature databases. Experts agree[3][4] a major downside to this approach is that this type can not discover any type of unknown malware not already in a database.

Heuristic-based detection, on the other hand, is used to discover both new types of malware, and altered versions of already known malware. This is done by analysing files that do not have a signature match, and then comparing that with statistically suspicious characteristics. One such characteristic could be unusual instructions or junk code seemingly serving no purpose. Seasoned information security professional currently teaching at SANS institute, Lenny Zeltser[5], informs that this type of detection is often used in conjunction with signature-based detection but has a higher rate of false positives.

[1] https://us.norton.com/internetsecurity-malware.html

[2] https://technical.nttsecurity.com/post/102efk4/detecting-malware-through-static-and-dynamic-techniques

[3] https://www.infosecurity-magazine.com/opinions/malware-detection-signatures/

[4] https://searchsecurity.techtarget.com/tip/How-antivirus-software-works-Virus-detection-techniques

[5] https://searchsecurity.techtarget.com/tip/How-antivirus-software-works-Virus-detection-techniques

3.1.2 Dynamic detection

Jeremy Scott[6] also explains **dynamic detection**. It is based on running files in a controlled and simulated environment and observing what they do. Functionality will be analysed and technical indicators identified. These technical indicators may include domain names, IP addresses, file paths, etc. Dynamic detection can also identify communication with an attacker-controlled external server for command and control purposes. Dynamic analysis can be seen as what most sandbox environments do today.

Curtis Cade at OPSWAT blog[7] describes **sandbox detection** as a technique that can be summed up as using isolated, often virtual, "sandbox" environments to analyse suspicious files by executing them and recording their behaviour. When a file is executed it is automatically analysed through a weight system or by a malware analyst, often both. With sandboxing, a detailed report on the behaviour of the malicious files is compiled and generated. One specific example of a sandbox detection system, Cuckoo, is explained in detail later in section 3.2.4.

According to Lenny Zeltser at the SANS institute[8], **behavioural-based detection** reads the execution of suspicious files and observes potentially malicious activity. Some particular parameters that are observed are access requests, network connections, modification of host files, etc. When analysing these, potentially malicious activity can be observed and based off of this activity it is often possible to determine whether a file is malicious or not. As this is a type of dynamic detection, it requires running files to perform scans.

3.1.3 Malware Anti-VM techniques

With the advancement of dynamic detection and sandboxing, malware authors have designed countermeasures to avoid detection. One of these countermeasures is detection of virtualised systems and sandboxing[9]. If a malware with anti-vm countermeasures detects that it is executed within a virtualised environment it will not deploy malicious code or its payload to try to avoid detection.

3.2 Analytical Infrastructure

To support the requirements from section 2.1, we will need an infrastructure that can manage files throughout an entire scanning process. That includes handling relevant user input, managing full scans from start to finish, supporting a large number of static and dynamic anti-virus engines, controlling scans and distributing them effectively to the different engines, compiling results and storing these along with other needed data.
An infrastructure fulfilling these criteria can be reasonably abstracted to three main parts:

- front end
- task handler
- analysers

The front end should be the point of access for all user interaction, and consist of an

[6] https://technical.nttsecurity.com/post/102efk4/detecting-malware-through-static-and-dynamic-techniques

[7] https://www.opswat.com/blog/understanding-heuristic-based-scanning-vs-sandboxing

[8] https://searchsecurity.techtarget.com/tip/How-antivirus-software-works-Virus-detection-techniques

[9] Anti-VM and Anti-Sandbox Explained: https://www.cyberbit.com/blog/endpoint-security/anti-vm-and-anti-sandbox-explained/

API and a graphical user interface (GUI). Significant user actions such as file input and viewing both new and old results must be available through the front end.

The task handler should act as a controller for all analysers, meaning it should distribute and organise all tasks. In other words, it should be the middleware between the front end and analysers. The relationship between front end and task handler should be one to one, while task handler to analysers should constitute a one to many relationship. The task handler will need to be aware of all available analysers, and must be able to present files for analysis and get a result back, and further return results back to the front end to present to the user.

The analysers will provide the required analysis for the previously described infrastructure. This involves handling the actual scanning of files and detection of malware. The infrastructure can consist of anti-viruses, dynamic analysis tools, metadata retrievers, etc. There are multiple ways to host these, where the two most relevant are; all tools gathered in one host or hosting relatively light weight tools together and hosting resource intensive tools by themselves. Especially dynamic analysis tools demand a high amount of resources, and can be thought of as their own platforms. Dynamic tools often rely on using their own probes to perform their analysis.

- All-in-one will allocate one shared resource pool for all analysis tools. This works for small scale usage, development, and testing. It would likely lead to congestion and starvation because of lacking resources, and worst case a complete crash and forced shut-down when used at a larger scale. Dedicating resources efficiently within such a solution would be a challenge.
- Separating and grouping analysers based on resource usage should prove beneficial at larger scale by allowing hardware resources to be efficiently allotted. This entails grouping tools like independent static analysers together to share a resource pool when compatible. It is necessary to carefully set the amount of available resources to match the need of these analysers. More advanced and demanding analysers like dynamic tools should receive their own allotted resources to allow their required sub-probes to function optimally.

With an all-in-one solution used during development and testing, it was discovered how important it can be to separate probes appropriately. Trying to run both the general system and a dynamic analyser off the same host proved very unstable and resulted in multiple complete crashes forcing manual reboots. Hence why option two should be considered if the system is to face moderately scaled or higher usage.

An aspect yet to be mentioned, is storage of the data in an infrastructure like this. The data includes long term storage for analysis; scan history, individual data from all past scans, compiled scan results, logs; long term storage to ensure repudiation; logs of actions performed in dynamic scans, specific user responsible for requesting scans; and short term storage to handle logic; fast access to tasks, queues to track tasks and delegate them for analysis, keeping track of results from individual analysers before compiling them together. The natural approach to this would be to store long term data on disk in a normalised database in the front end for quick retrieval, or more securely; the same database approach on a dedicated logging platform. To potentially increase efficiency, the database could cache hashes from all previous files to reduce redundant actions by not scanning previously scanned files. Short term storage should consist of quicker memory based databases and small data structures like queues to track tasks, scans, when and where to delegate, etc. These would logically be situated on the task handler for efficiency.

One last aspect to consider is employing a kiosk solution. A kiosk solution would simply act as a gateway to the front end for users. Input from users could be handled here before reaching the front end and the actual analysis infrastructure. This would be a natural place to add user authentication before input is sent to the infrastructure. This would make it possible to enforce a policy for only authenticated employees to initiate scans, leading to removing a small risk of abusing an insecure entry point.

This infrastructure should explicitly be allowed minimal network connectivity to compensate for the fact it is a zone dedicated to analysing insecure files. This should obviously be constrained to one limited area as to avoid spreading any malware found to the rest of the organisations network. The kiosk would benefit from specific and limited connectivity to an authentication service. Otherwise, the front end can benefit from controlled network access to; accept requests from authorised users from the organisation network; share secure files after a scan to the responsible users file space, e.g. in an Active Directory structure.

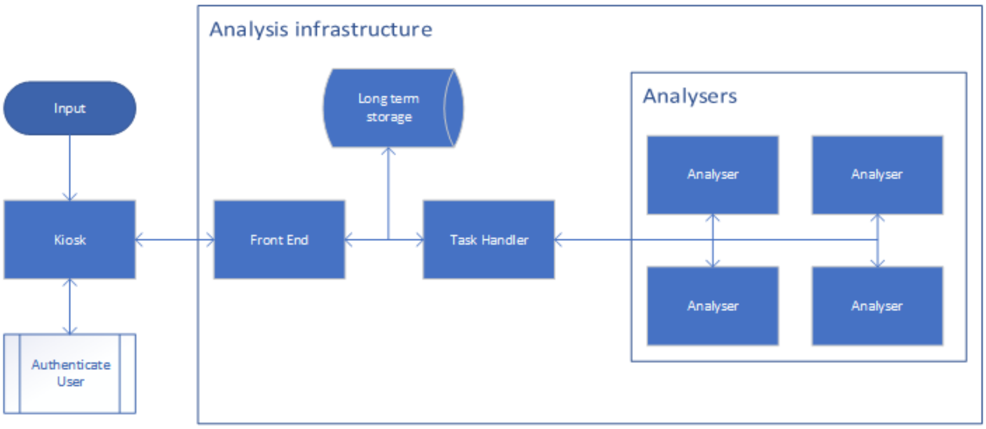


Figure 3.2: Conceptual Infrastructure

3.2.1 Infrastructure Alternatives

Creating an infrastructure of this complexity from scratch was too large of a task for a relatively short bachelor project, therefore existing alternatives were researched. Discovered alternatives were:

- IRMA [10]
- OPSWAT MetaDefender [11]
- VirusTotal [12]
- Multiscanner [13]

[10] IRMA:https://irma.readthedocs.io/en/latest/intro/supported_probes.html?highlight=analyzers

[11] OPSWAT MetaDefender:https://www.OPSWAT.com/products/MetaDefender

[12] VirusTotalurlhttps://www.virustotal.com

[13] Multiscanner:https://multiscanner.readthedocs.io/en/latest/overview.html

Requirements/Alternatives	IRMA	OPSWAT	Multiscanner	VirusTotal
On-site	Yes	Yes	Yes	No
Open source	Both	No	Yes	No
Automated setup	Yes	Unknown	Yes	Unknown
Yara support	Yes	Yes	Yes	Yes
Dynamic analysis	No	Yes	Yes	Yes
Known good/bad	Yes	Yes	Yes	Yes
Static analysis	Yes	Yes	Yes	Yes
Pipeline scans	No	Unknown	No	Unknown
API	Yes	Yes	Yes	Yes
Web interface	Yes	Yes	Yes	Yes
Free version	Yes	Trial	Yes	Yes
Number of probes	30	30+	30	70+
Machine learning	No	Yes	Yes	Yes

Table 3.10: Analytical Infrastructure frameworks comparison

VirusTotal

To understand the goal of the product, an understanding of VirusTotal is required as it is the main inspiration for the entire assignment. It sets the baseline for expectations and requirements of the project.

"*VirusTotal aggregates many antivirus products and online scan engines to check for viruses that the user's own antivirus may have missed, or to verify against any false positives.*" [14]

VirusTotal[15] works by either uploading a file, searching by hash, or linking to a file or website for scanning or searching in the VirusTotal known hashes database. Additionally it gives the user the ability to rate the result and comment on it which can prove useful. The main issue is as explained that it is not available as an on-site, in-house solution; that the scan results are shared with the world. This allows malware creators to automate and change their malware when its discovered, and it also breaks KDAs requirement of strict confidentiality (Section 2.4).

OPSWAT MetaDefender

OPSWAT, the company behind MetaDefender, has the philosophy *Trust no File. Trust no Device*. They are focused on reducing insecurities within the platform and creating a secure environment for malware detection and analysis.

The analytical platform in itself is very similar to Virustotal, but have added further features such as a kiosk solution. It also have data sanitisation built into the solution which removes potential malicious elements from files.[16] The analytical system is compromised of 30+ antimalware engines and checks for unusual content in different file types, 30 file types are supported[17].

MetaDefender has a lot of similarities with VirusTotal, but can also be acquired as a on-site solution for a price and this approach includes options for the customer organisation to decide every facet of what is shared and what is kept private. This module is also well established in the market, but it lacks publicly available documentation of capabilities and design. As KDA stated a clear wish for the system to available for free usage for anyone (section 2.4), a closed source solution as MetaDefender is hardly qualified for use in this project.

IRMA

IRMA: Incident Response Malware Analysis[18] is an on-site infrastructure framework made by french company Quarkslab. The module is available with built-in orchestration and provisioning options; Hashicorp's Vagrant and Packer using pre-built images, and automatic Ansible provisioning. Setup is almost completely automated with a few errors which can be mostly fixed through available configuration. The main issue with IRMA is that it does not support dynamic analysis by default, though it can be added through additions in the publicly available official code base on Github [19] An enterprise

[14] VirusTotal: https://en.wikipedia.org/wiki/VirusTotal

[15] About VirusTotal: https://support.VirusTotal.com/hc/en-us/articles/115002126889-How-it-works

[16] Data sanitization: https://www.opswat.com/technologies/data-sanitization

[17] Opswat: https://www.opswat.com/products/metadefender

[18] IRMA:https://irma.readthedocs.io/en/latest/

[19] Quarkslab/IRMA at Github: https://github.com/quarkslab/irma

option is available, but it is barely documented nor open source.

IRMA was selected as the framework for the project as the sole candidate found during the research phase to support on-site hosting and an open source code base. Though IRMA lacks features such as dynamic analysis support, pipelined scans, kiosk support, and have been found to have idempotency issues during provisioning. [20] Pipelined scans is not a direct requirement, but is a functionality that increases scanning efficiency by a large degree, therefore, also improving scalability. This lead the development team, in cooperation with KDA, to set the project goal to testing, fixing and adding as much as possible.

Multiscanner

Nearing the end of the development phase Multiscanner by Mitre was discovered. Through the documentation it seems to be very good candidate, but it was not considered as an alternative as it was not known as of late in research and early design phases.

Multiscanner supports a wide array of probes, the main difference to IRMA being the different architecture and the probe types it supports. It has native support for sandbox execution and machine learning which is not a feature in IRMA. Multiscanner has a wider array of metadata modules but lacks in anti-virus modules natively supported. It supports three sandbox modules:

- Cuckoo
- FireEye API
- VxStream

Based on a simple overview Multiscanner fulfils a wider array of the required capabilities, but lacks in its provisioning[21].

Additionally, it has been tested in containerised environments, which is something IRMA itself struggles with, though it does not seem to be a difficult fix. [22]. Containers should not be used for modules of the system such as dynamic scanning, but if used for modules not executing potential malicious content, the decreased security should not be a problem. By using the containers the setup speed of the system should increase.

3.2.2 Chosen framework: IRMA

IRMA has two versions, one enterprise edition and one open source. The open source alternative is officially termed IRMA OSS, but this report will continue to use IRMA to reference the open source edition.

In addition to VirusTotal inspiring the assignment, KDA also had knowledge of IRMA beforehand, and it helped shape the expectations for the system. As mentioned in the earlier IRMA introduction, section 3.2.1, the development team and KDA decided IRMA was the top candidate for the project. This due to the open source nature of the framework and the on-site focus, together with a high degree of out-of-the-box automation, relatively covering documentation, and a high number of supported analysis tools.

An important sentiment shared between the KDA interests and the Quarkslab team;

[20] IRMA: https://irma.readthedocs.io/en/latest/

[21] Multiscanner https://multiscanner.readthedocs.io/en/latest/

[22] Twitter post: https://twitter.com/mboman/status/865473084270059521

"you keep control over where data goes and who gets your data."[23]

Analysis Process

The IRMA analysis process[24] can be summed up like this: start by uploading files to the front end. When files are uploaded their SHA256 values are checked against a database of past results to test if any of the individual files have been scanned before. All unscanned files will be added in the front end FileSystem under their SHA256 values and the values will be stored in the database for scan tasks to be started. The scan tasks relay the files to the task handler, the brain, where the files are temporarily stored in a FTP server while the brain is diverting different tasks to all running analysis tools, the probes. After the probes return the results from their analysis the brain will process the results and return them to the front end for storage. The results will be stored in the database on the front end, and be available to the user. See appendix D for figures displaying a simple overview of the dataflow within IRMA.

Architecture

Like the analysis process is hinting towards, the IRMA architecture is matching the proposed analytical architecture design discussed earlier in the section 3.2. This is seen in IRMA being built with a front end, a controller (brain) serving as a task handler, and analysers in the form of numerous probes.

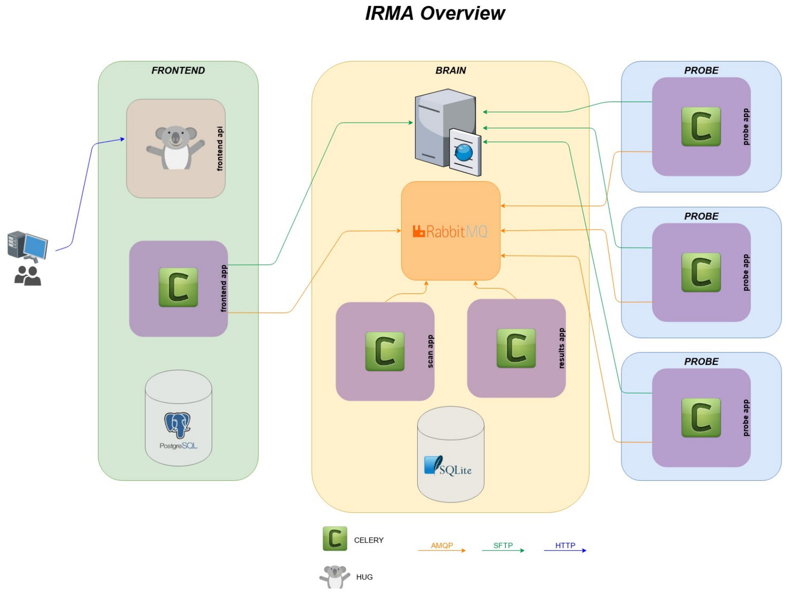


Figure 3.3: IRMA architecture

Every component uses python-based applications to function, and a key aspect is the Celery distributed task queue library. The official technical description can be found at IRMA.readthedocs.io [25].

[23] IRMA official purpose: `https://irma-oss.quarkslab.com/`
[24] Analysis Process:`https://irma.readthedocs.io/en/latest/intro/process.html`
[25] Technical description:`https://irma.readthedocs.io/en/latest/technical/index.html`

Front end

The IRMA front end has a web-gui and an command line interface available. These are based around a RESTful Hug API running on a NGINX web server with a uWSGI application server to handle user interaction. The earlier mentioned long term storage requirement (section 3.2) is covered by a PostgreSQL database on the front end. The database contains all results from earlier scans and information about current scans.

When the API receives a scan request with new files, it will immediately search the database for any matching file-hashes from earlier scans unless explicitly stated to force a new scan. If the database does not contain matches, or a force parameter is set, Celery workers start a scan task to transfer the files to the brain, requesting scans and expecting a complete result in return.

Brain

In IRMA the brain complies with the role earlier defined as file handler (section 3.2). It contains an SFTP server to house files for scanning, a RabbitMQ server to broker queueing between the front end and scan tasks on brain, relaying them to the probes for analysis, and lastly returning the result back to the front end. To ensure efficiency in the ideally short life span of scan tasks, an SQLite database is implemented to store the required data.

These elements can be equated to objects in object-oriented programming. Celery workers and handlers are responsible for handling the tasks to and from the front end, and between probes and brain. They tie the different systems together to form the main logic of the IRMA anti-malware platform.

Probes

IRMA natively supports multiple methods for detection and analysis. Out of the box there are four different supported categories of probes[26]:

- **Anti-virus**
  There are 22 natively supported anti-virus programs. Seven run on Windows, with 15 on GNU/Linux. Adding new anti-virus software should be easy as Quarkslab has provided a skeleton and a thorough description[27] of necessary steps to integrate new anti-virus programs properly.
- **External analysis**
  VirusTotal and ICAP servers are supported. File hashes are sent to these and results retrieved. VirusTotal will not be used as it opposes the whole purpose of keeping data purely in-house.
- **File databases**
  Only one is natively supported. NSRL is a public database available of known signatures that can be downloaded for usage in isolated networks.
- **Metadata**
  IRMA supports five metadata analysers that extracts and returns metadata from files. If mimetype filtration is activated for a scan, it means files are filtered based on their file type so only files compatible with the respective analyser will be analysed.

[26] Supported probes: https://irma.readthedocs.io/en/latest/intro/supported_probes.html

[27] Writing your own Analyzer for the Open-Source Multi-Scanner IRMA: https://blog.quarkslab.com/writing-our-own-analyzer-for-the-open-source-multi-scanner-irma.html

The details surrounding why this works are very low-level and beyond the scope and relevance of this project. An introduction to file system concepts necessary to understand for this, see Complete Tour of PE and ELF: An Introduction.

3.2.3 Dynamic Malware Analysis tool

As IRMA doesn't support a dynamic analysis probe by default, one needs to be selected and added. Luckily it is easy to extend IRMA to support it. Main requirements set for a chosen dynamic detection tool: it should be extendable by other tools, open source and easy to use with an API. FireEye AX and VxStream was also considered, though at a later point in the development. They were simply not fit as they are neither open-source nor free (Section: 2.4).

Requirements/Alternatives	Bro IDS (Zeek)	Cuckoo	Joebox	ThreatAnalyzer
On-site	Yes	Yes	Both	No
Open source	Yes	Yes	No	No
Automated setup	No	No	Unknown	Unknown
Yara support	No	Yes	Yes	Yes
API	Yes	Yes	Yes	Yes
Host monitoring	No	Yes	Yes	Yes
Network monitoring	Yes	Yes	No	Yes
Free	Yes	Yes	No	Trial

Table 3.11: Dynamic Analysis Tools Comparison

Cuckoo resulted in the best fit for our project as the only other open source option is Zeek and it only supports network monitoring. A huge time sink in adding a sandbox is that none have official automatic provisioning. Luckily some unofficial were found. They did not work perfectly, but they were a good baseline for an Ansible module developed for Cuckoo for this project. Additionally IRMA needs an interface with which it can communicate with the analyser. This interface took inspiration from the code shown in the documentation on extending IRMA [28].

3.2.4 Chosen dynamic analysis tool: Cuckoo

What is Cuckoo

Cuckoo Sandbox is a dynamic analyser that analyses unknown or suspicious files behaviour within a controlled and contained environment on demand, also called a sandbox analysis tool. The sandbox should be as similar as possible to the production environment of the recipient/target. It should emulate a normal environment to not wake suspicion with advanced malware and programs. A file should be submitted if the sender is unknown, if its content is deemed suspicious or the target environment is of sensitive enough nature. A file is analysed to understand how it would behave if it were opened or executed on a device used by employees, a regular user or a server (Windows, Linux, Mac, etc.). It is also possible to add support for Android Virtual Device (AVD). Cuckoo officially describes itself as:

> "*Cuckoo Sandbox is the leading open source automated malware analysis system.*"[29]

[28] Extending IRMA, for a probe that is not a antivirus: `https://irma.readthedocs.io/en/latest/extending/add_probe.html?highlight=mimetype#for-a-probe-that-is-not-a-antivirus`

[29] https://cuckoosandbox.org/

How Cuckoo functions

Cuckoo is usually accessible through a web page and an API where you can submit files for analysis, see status of scan, results and reports. The API has the same functions as the web page but enables automation and extending the interface of cuckoo, e.g a email service.

When a file is submitted, Cuckoo will download the file on an incoming interface. Cuckoo reverts to a specific VM snapshot set in the machinery configuration and starts the VM. When the VM is started and the Cuckoo agent is ready on the guest VM, Cuckoo will send a file for execution and analysis. During execution Cuckoo will gather data for later analysis. Dependent on what modules are available and enabled Cuckoo will trace API calls, dump and analyse network traffic (also encrypted) and perform advanced memory analysis[30]. These are default features when following the official cuckoo installation guide. More features can be enabled[31], for example Suricata. When execution and monitoring is finished, Cuckoo will process the results and create a report. To see a overview of the dataflow of Cuckoo check hatching.io

Why use Cuckoo?

In rapidly evolving and complex environments there are a lots of potential weaknesses and exploitable parts. The bigger and more complex an environment becomes, the more important it is to have control and decide what files are allowed entry within its borders. Most, if not all of the probes supported by IRMA are static analysers and meta analyser. These only look for predetermined patterns and values, a set of signatures, heuristics or a set of rules, they are not able to analyse what is done during execution. They quickly and efficiently find known malware and malicious patterns. Some of them even have an Host-Based Intrusion Protection System (HIPS)[32].

With mostly, or only analysers that analyses files in a static environment Cuckoo aims to fill the need of a dynamic analyser. The purpose of Cuckoo is to catch the malicious files that the static analysers cannot detect. Cuckoo should detect malicious files using new, non-public, or obscure techniques not yet classified by the other analysers. Cuckoo should ideally also detect targeted software from APTs or other advanced threats. It is a last defence against advanced malware that manages to bypass other detection methods.

3.3 Configuration Management

3.3.1 Automatic provisioning

Kief Morris' Infrastucture as Code defines provisioning as making an infrastructure element, such as an infrastructure server or network device, ready for use. Depending on the infrastructure elements being provisioned this could involve a couple of things[1]:

- Assigning resources to the element
- Instancing the element
- Installing software onto the element
- Configuring the element
- Registering the element with infrastructure services.

[30] https://cuckoosandbox.org/

[31] Adding processeors: `https://cuckoo.sh/docs/customization/processing.html?highlight=process`

[32] https://cdn1.esetstatic.com/ESET/INT/Products/Home/EAV/v12/ESET_NOD32_Antivirus_Product_Overview.pdf

At the end of the provisioning process the element would be fully ready to use. There are three main points you need to have an effective provisioning process:

- Any existing infrastructure element can be rebuilt on demand
- Any element can be rolled out and replicated to several instances and environments after being defined once.
- The process for provisioning and defining any element should be transparent and easy to modify.

3.3.2 Push vs pull model

Kief Morris' Infrastructure as Code explains the two models: push and pull[1]. In a push model for configuration management, a central/master server pushes updates/configurations to the servers it manages. This model is used by the configuration management tool Ansible. Some systems using the push model may need a client running on the servers the configuration are pushed to. Other systems, like Ansible, uses SSH to connect and execute commands. The advantages of push systems is that they give you more control over where and when to update and apply new changes.

Another model for configuration management is the pull model, where an agent is installed on each server to schedule and apply changes. The agent runs periodically, and checks a master or central repository for the newest configuration and applies it to its own server. This method is used in the configuration management tools as Puppet and Chef.

An advantage with the pull model is that it's a more simple security model. The servers in a push model must find a way to expose themselves to central repository/master and thus need to implement an authentication solution. Ansible solves this by connecting over SSH.

A pull based system may also be more scalable than a push based system. When a push system needs to scale up it must open up connections to more hosts, which could cause bottlenecks.

3.3.3 Configuration Management Strategy

The Configuration Management System section explains a type of configuration management, configuration synchronisation, but what is a configuration management strategy? Configuration management strategy is the different ways/strategies to approach configuration management in an infrastructure. Configuration synchronisation through configuration management systems is one of several approaches to avoid configuration drift which is the main purpose of configuration management systems. Configuration drift is in the words of Kief Morris:

> "*... the phenomenon where servers in an infrastructure become more and more different from one another as time goes on, due to manual ad-hoc changes and updates, and general entropy.*" [33].

Configuration drift in a production environment, or any environment, means inconsistencies across servers and possibly services which is bad for maintenance, concurrency and possibly stability and performance. Configuration drift makes troubleshooting and a applying fixes a lot harder since you cannot necessarily know with certainty that the

[33] http://kief.com/configuration-drift.html

failure you found is the correct one, the only one or the same across all the different systems. These are the systems we try to avoid with configuration management also called snowflake servers[34]. The name comes from their potential brittle and fragile existence in a bigger collected environment.

Ad Hoc Change Management

Ad Hoc is basically just manual maintenance and configuration of infrastructure and servers. There is no significant automation and synchronisation across services except what is done manually. It is not preferable in any form of production environment or any environment at all where consistency and and control is in focus.

Mutable

Mutable according to Merriam Webster is described as:

> "*capable of change or of being changed*"[35]

That is also exactly what it means in the context of Configuration Management. A server, or infrastructure managed by changing the configuration on the fly on a running server. This can be done manually, with a Configuration Management System, updating the system or any other way of intentionally changing a running servers configuration.

Configuration Synchronisation

To update and manage a system or infrastructure with a Configuration Management System across all relevant systems/servers with the same configuration, in theory ensuring that all systems have the exact same configuration. If a system is running over an extended period there is a chance configuration drift might occur even when using a Configuration Management System. A Configuration Management System cannot control all aspects of an individual system or server and all of it's components. An update to some or several components can for example change the behaviour of some but not all parts of an infrastructure or system. In an ideal environment this would never happen.

Immutable

Immutable is the opposite of mutable, "*not being capable of change or susceptible to changed*"[36]. This is usually done with virtualisation or Containerised services. In the case of virtualisation it is usually done by have a "clean" base image, then manually or with a Configuration Management System install and configure the relevant programs and services for that server. For a Containerised service it is usually with a "recipe" where you specify what base image should be used and what programs and services should be installed. The main difference in comparison to mutable servers is how a update of configuration or programs are handled. Instead of changing the configuration on a running server a new image is created preferably from the previous one. The old server with the outdated service/configuration is removed and a new server with the updated service/configuration is created/started. This allows for quick scaling and restoring of the services as long as the current image is available. It can both be quicker and slower on update all depending on size of update/changes and infrastructure.

[34] https://martinfowler.com/bliki/SnowflakeServer.html

[35] https://www.merriam-webster.com/dictionary/mutable

[36] https://www.merriam-webster.com/dictionary/immutable

Containerised services

Containerised services is run usually run alongside Linux, or other supported operating systems. This means it shares resources and services with its host system. This allows for less overhead. This also means that the services on the subsystem are closer to the host system. Containers have existed for a long time as LXC on Linux[37] but is today most popularly used as Docker containers. [1]

3.3.4 Configuration Management Tools

There are many configuration management tools and approaches. After establishing an overview of the different options we decided to evaluate the four biggest and most known and established configuration management tools; Ansible, Chef, Puppet, and Salt. Each has different advantages, disadvantages, and a approaches to reach the same result. The main differentiator in execution is master vs masterless / client-server approach.

Puppet

Puppet was the first of these four developed with an initial release in 2005. Being developed 14 years ago it has matured a lot. Many modules have been developed by both Puppet and third-parties. As one of the first proper configuration management tools it was the first to do a lot of things. Not having any previously known pitfalls, a lot of sub-optimal decisions were made.[38]

One of the main functionalities of Puppet is definition of configurations for each and every host. Once a configuration has been defined Puppet will continuously check if the required configuration has been altered and revert it back to its original state. Puppet uses a master-node architecture, a change made on the centralised master-repo will automatically spread to other configured nodes. [39]

Modules are created using Ruby or Puppet's own DSL[40], creating a bigger threshold for new developers and users having to potentially learn a new language. Puppet has strong reporting capabilities built into it reporting back from client to server. Puppet Enterprise has built in tools to view the reports in a web browser. Puppet Opens Source does not have a built in viewer but can use third-party viewer such as Puppetboard or PuppetExplorer. This needs to be explicitly enabled in configuration to work since it is not enabled by default[41]. It can also use git for version control through r10k. Though the link is to a Puppet Enterprise site, r10K and its capabilities are also available in Puppet Open Source[42]. Learning Puppet can be quite challenging since you will need to learn Puppet DSL, how Puppet works, and potentially Ruby.

Chef

In comparison to Puppet that uses it's own DSL in combination with Ruby, Chef uses a Ruby-based DSL[43]. Chef uses git as a deployment platform which brings along strong version control called chef-repo[44]. Chef has a command-line tool called knife that can

[37] https://en.wikipedia.org/wiki/LXC

[38] Puppet: https://puppet.com/company

[39] What is Puppet: https://www.edureka.co/blog/what-is-puppet/

[40] Puppet Wikipedia Page: https://en.wikipedia.org/wiki/Puppet_(software)

[41] Puppet Documentation: https://puppet.com/docs/puppet/6.4/reporting_about.html

[42] Puppet r10k https://puppet.com/docs/pe/2019.0/r10k.html

[43] Chef Wikipedia page: https://en.wikipedia.org/wiki/Chef_(software)

[44] Chef Documentation: https://docs.chef.io/chef_repo.html

help you manage your environment [45]. It can have quite a steep learning curve as you may need to learn Ruby and how Chef works at the same time, assuming you do not know Ruby beforehand. To manage a Chef environment, a workstation is needed to manage the master to manage the clients.[46] [47]

Ansible

Ansible is masterless. Its playbooks are written in Python with YAML formatting. The playbooks are pushed to clients over ssh [48]. Ansible nodes can pull configurations but requires Ansible to be installed on the node to execute as local-host. When serving many nodes at the same time it can slow down needing a lot of resources due to limitations of SSH [49]. Ansible does not provide a good way to view and manage your hosts, which roles are currently deployed or which version used on which host. Ansible Tower or AWX is a good option with a web-based user interface which solves a lot of these challenges [50].

Salt

Salt uses SSH, but the scalability over Ansible is greatly enhanced by the use of agents called minions [2]. It has an asynchronous file server to speed up file serving to minions [2]. It has the potential for greater scalability and resiliency with support of multiple levels of masters and tiered arrangement [51].

3.3.5 Configuration Management requirements

A provisioning tool is needed to automate the setup and start the necessary processes to run IRMA. Ansible is natively supported, but the setup has it flaws and Ansible its disadvantages. The flaws in the IRMA Ansible provisioning is mentioned later in the implementation, the disadvantages can be summed up as:

- OS restriction
  Ansible has great support for Unix based OSes where ssh is installed by default, but the moment it tries connecting to Windows problems arise. This is mostly because it is agentless and push based with SSH as its main method of connecting to machines. It also supports WinRM, but this isn't enabled by default on Windows machines and the setup can be fragile. This is solved in the system by simply creating images with the correct configuration.
- Stateless
  This leads to tasks being repeated multiple times and decreasing the overall speed of the setup, though this is very dependant on the way the Ansible modules are designed. If they have a very modularised design, the playbooks are created for small sub tasks and are combined to complete larger tasks which is why this won't be a large issue.
- Speed
  The speed can be lacking when deploying larger infrastructures, but even this can be solved by modularising playbooks into smaller sub tasks. Additionally, speed can

[45] Chef knife: https://docs.chef.io/knife.html
[46] Chef System Requirements: https://docs.chef.io/chef_system_requirements
[47] Chef workstation: https://www.chef.sh/docs/chef-workstation/about/
[48] Ansible Architecture: https://en.wikipedia.org/wiki/Ansible_(software)#Architecture
[49] Ansible Performance: https://opensource.com/article/19/3/ansible-performance
[50] Ansible AWX: https://github.com/ansible/awx
[51] Salt Multi Master: https://docs.saltstack.com/en/latest/topics/tutorials/multimaster.html

be improved by creating asynchronous tasks that can be run in parallel. The speed can also be increased as with all other configuration management tools by having a local file server for downloads.[52]

Advantages:

- Ansible Galaxy
  It is a Puppet Forge-like online repository for existing modules which are easy to use and extend upon.
- Agentless[53]
- YAML playbooks
  YAML is more readable, allows for comments and makes it easier to use variables than the JSON format often employed by other tools.
- Easy to learn/understand

We see no reason to move away from Ansible as it is already implemented in IRMA. There is little to no potential benefit to rewriting IRMA in a different provisioning tool, as it would be of an immense cost to understand the code and redevolop it. For the benefit to outweigh the cost the provisioning speed and idempotency using that tool would need to increase by a huge margin without losing any advantages already present.

3.4 Orchestration

According to Kief Morris' Infrastructure as Code an infrastructure definition tool, also known as an orchestration tool, allows people to specify what infrastructure resources they want to allocate and how they should be configured.[1]

Orchestration is used to described various aspects of computing. In our case it is specifically in the context of orchestration, allocation and management of processing resources. Configuration of applications and services are not the main objective of orchestration in our case. There are often overlapping features or integrated certain functionality used for configuration of applications or services in orchestration tools. It is also the other way around for configuration management tools. Orchestration can be managed in several ways; manually, with a configuration management software (Infrastructure as Code), virtualisation, provisioning/orchestration tools etc[54].

3.4.1 Orchestration requirements

Our case relies on an in house solution with changing requirements for resources, virtualisation initially seems like a good option. Regardless of the solution we choose, it needs to be able to dynamically allocate and reallocate resources from one service to the other. By default IRMA uses Vagrant in combination with Packer to orchestrate machines for running IRMA, keep in mind this solution is not meant for production, but only for development environments[55]. KDA requested that the infrastructure was capable to support VMWare, which Vagrant does although through a plugin which needs a licence. The plugin lacks some features, none of which seems to be of immediate concern[56].

[52] Asynchronous playbooks: https://docs.ansible.com/ansible/2.4/playbooks_async.html

[53] Agentless: https://www.upguard.com/blog/agent-vs-agentless-and-why-we-chose-agentlessskrivom

[54] Orchestration: https://en.wikipedia.org/wiki/Orchestration_(computing)

[55] Vagrant: https://www.vagrantup.com/intro/vs/terraform.html

[56] Vagrant VMWare: https://www.vagrantup.com/vmware/index.html

During the development phase for the thesis a combination of Vagrant and Packer was used which comes with some downsides:

- Resource limitation of working environment
- Differences in implementation between hypervisors
- Security concerns
  Vagrant uses an ssh key retrieved from a public Github repository to access the machines it changes. This can be changed but is something to be aware of.

For a production environment the requirements differ, Vagrant isn't as capable when it comes to cloud orchestration, which is then often solely doable through community plugins (e.g. for VMWare)[57], which have a habit of becoming deprecated and flawed. Requirement specification for a production environment include:

- Cloud agnostic
  Support a variety of different cloud architectures
- Boot or use a machine from a snapshot
  Pre-build an image or snapshot so the setup is quicker

For the production environment that KDA would hopefully deploy, Vagrant should in theory not be used, as its explicitly not designed for production[58]

3.4.2 Packer

Packer creates base images called boxes for selected hypervisors and configures these to fit the system. This makes provisioning easier at later stages in the process by enabling SSH on Windows machines as an example. Packer boxes should only contain parts that are necessary for provisioning in the later stage and parts that are rarely changed. By doing an analysis of what should and shouldn't be built into an image, one can reduce setup speed drastically.

Different methods exists for increasing packaging speed of boxes. Packaging can be splint into multiple stages, images built with a rarely done configuration and uses this image/box as a basis for the other boxes.

Boxes built with Packer only supports the hypervisor they were built for by default. This means it is necessary to build for each hypervisor separately for each configuration. There is one exception, which is Packer's capability to rebuild OVA/OVF boxes into Virtualbox boxes. OVA/OVF are open virtualisation formats which supports multiple different hypervisors. Multiple hypervisors are capable of exporting to and importing from the formats, though this is not natively supported in either Packer or Vagrant. By tinkering with APIs for different hypervisors it should be possible to automate the process of importing and exporting between the virtualisers.

If used in an infrastructure one can add smoke testing at the end of the packaging workflow to verify that everything appears to be working. This will decrease automation fear.
An example of the workflow can be seen below in figure 3.4.

[57] VMWare plugin: https://www.vagrantup.com/vmware/index.html
[58] Vagrant production: https://www.vagrantup.com/intro/vs/terraform.html

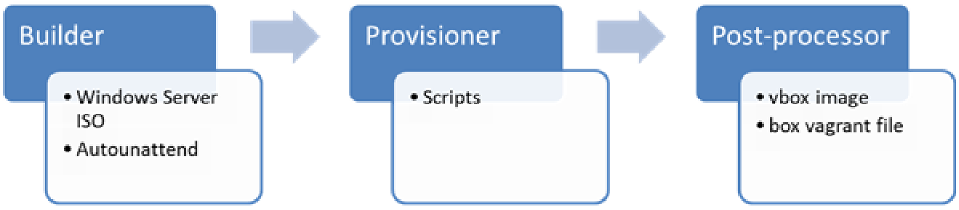


Figure 3.4: Reproduced with permission. Copyright 1999-2019 QuinStreet, Inc. All rights reserved.

3.4.3 Vagrant

Vagrant is a tool for orchestrating virtual machines in a single workflow. The Vagrantfile is used to configure and provision your virtual machine environment. In the Vagrantfile you can specify which type of box you want to use for your machines, network configurations and programs to be installed, which is called provisioning. Once the vagrantfile is configured you only have to run vagrant up to set up your virtual machines, vagrant halt to shut them down and vagrant destroy to remove them.

It shouldn't be used in production environments per its own documentation as it is mainly used for managing local VMs. It can be extended to support other platforms (e.g. Openstack), but this is often through community created modules which has a varying dependability.

Vagrant is the default orchestration tool used by IRMA and is used for creating machines out of boxes that are stored in a Hashicorp Box repository or locally. IRMA has specialised boxes for its machines which only supports a small amount of hypervisors[59].

3.4.4 Virtualisation

Virtualisation, or hardware virtualisation in our case:

> "*refers to the creation of a virtual machine that acts like a real computer with an operating system.*"[60]

This allows a full separation between the two systems, that is logically in software. This makes it possible to run malicious software in a controlled and segregated environment separate from the host system. There are a lot of different type of virtualisers and providers. There are both Type 1 and Type 2 virtualisers. Type 1 is a virtualiser running directly on hardware while a Type 2 virtualiser is running with a host operating system like Windows or Linux. Since Type 1 virtualisers are running directly on hardware with no middleware, also called bare-metal, it is reasonable to assume that there is less performance overhead compared to type 2 virtualisers. Depending on the type of software/load running on a virtual machine, load on the host system and type of hardware the difference isn't necessarily that big between type 1 and type 2 virtualisers. There are option of both types of virtualisers.

Type 1

- ESXi

[59] Box repository: https://app.vagrantup.com/boxes/search?utf8=%E2%9C%93&sort=downloads&provider=&q=quarkslab

[60] Virtulisation Wikipedia article: https://en.wikipedia.org/wiki/Virtualization#Hardware_virtualization

- OpenStack[61]
- Xen[62]
- Hyper-V[62]
- KVM[62]

Type 2

- Virtualbox
- VMware Player/Workstation
- QEMU

[61] OpenStack is not a good comparison to the others as it is a lot more complex to setup and aimed at large scale infrastructure hosting and management.

[62] There are arguments that these are not technically type 1 virtualisers, but for simplicity that's what they are categorised as in our context.

4 Implementation

This chapter describes implementation details found during the project development phase. It highlights interesting challenges met, how these relate to the project, and workarounds, solutions, and potential fixes.

4.1 Configuration and set-up of IRMA

IRMA already had a mostly working installation method for the base system which only requires some configuration for it to be fully functional. Adding new analysers were more problematic as the installation is a bit clunky or lacking in many cases. We forked the official Quarkslab repository[1], and expect to send a pull request for our additions in the near future after performing quality assurance.

4.1.1 Hardware

As seen in the theory section IRMA consists of three modules; the brain, the front end, and possibly several probes. These can either be installed on one or several hosts. The brain and the front end must be installed on a GNU/Linux machine. At the start of the bachelor project our group was given a laptop with Ubuntu 18.04 that were installed the brain and the front end, the specifics of are described in section 6.2. IRMA does not estimate performance on any set ups, but for a small deployment the entire IRMA platform can be hosted on one host with several systems inside virtual machines which is what we did during the project.

4.1.2 Installation

IRMA can be installed manually or by using an automated install done with IRMA. We will only look at the automated installation.
The repository can be cloned using:

```
git clone --recursive https://github.com/quarkslab/irma
```

Running without the "recursive" tag misses a required submodule for running IRMA. We found out later that a file only was linked to in the repository, so we added the flag to retrieve that as well.

First one needs to retrieve requirements with specific versions, this can be done using pip to install packages mentioned in requirements.txt which is located in the Ansible folder.

```
pip install -r requirements.txt
```

The main requirement for the installation is:

- ansible = 2.4.2.0

[1] Quarkslab IRMA: https://github.com/quarkslab/irma

also this is installed using pip to install through the requirements.txt.

Otherwise to use IRMA's orchestration solution with Vagrant, one needs Vagrant installed with the selected hypervisor and a vagrant plugin for the hypervisor if necessary. We started out using Virtualbox which doesn't require a plugin simply to test the system, and changed to VMWare workstation after a while. VMWare requires the plugin vagrant-vmware-desktop which have a licensing fee. [2]
It can be installed and licensed with:

```
vagrant plugin install vagrant-vmware-desktop
vagrant plugin license vagrant-vmware-desktop ~/license.lic
```

The license file can be acquired from
By using VMWare we also had to create boxes supporting it, this process is covered later in section 4.3.5. As written in section 4.3.5 we experienced issues creating Windows boxes with Packer, which led us to use Virtualbox once again in the last phase of the project due to IRMA already having a box for it.
As for the file structure of the repository the most relevant folders are:

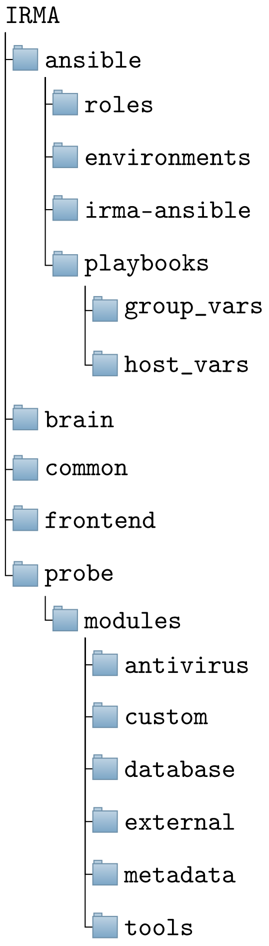


Figure 4.5: Folder structure

[2] VMWare plugin: https://www.vagrantup.com/vmware/index.html

The Ansible folder contains all code, mostly Ansible yaml code, for installing the system. The environment folder within describes how the infrastructure should look like. Some machine variables are set for each machine separately, these include IP and ansible_groups. Ansible_groups are roles within the Ansible setup, for the probes they often are the anti viruses to be provisioned. Additionally, it is important to mention that the development and production environments transfer necessary files differently. In development the transfers are performed directly with rsync, while production packs everything commited in the local git repository into zipped files before rsyncing. Configuration for a Linux probe can be seen below.

```
  - name: avs-linux.irma
    ip: 172.16.1.32
    ansible_groups: [avast, avg, bitdefender, clamav, comodo, escan]
    box: debian_vmware
    cpus: 2
    memory: 2048
```

These analysers are automatically provisioned and started if necessary. Additionally, on the bottom of an environment file there are general variables for the system in the section called ansible_vars.

```
ansible_vars:
  irma_environment: production
  vagrant: true
  irma_code_archive_generation: False
```

A full list of available variables with some description can be found in the file irma_vars.yml.sample in the Ansible folder. Most of these variables can also be configured in the configuration files placed in playbooks/group_vars and playbooks/host_vars folders. The difference between the files in the folders are that group_vars are more in general for machines while host_vars are more specific variables for hosts.

There is also a section for libvirt specific configuration called *libvirt_config*. We have not used this section in our project, but it can be used to manage the following virtualisation technologies: KVM, Xen, VMware ESXi, QEMU.
Additionally the environment files can be split into a few different categories based on their names:

- allinone
  Those with allinone in its name are installation where the whole system is installed onto one machine. In the others front end, brain and probes are installed on separate machines.
- dev and prod
  Dev specifies the environments that are created for development, in these necessary files are rsynced to the machines specified. In prod, which are created for production, files are transferred to archives which can be created manually or set to be automatically generated each time the playbooks are run by setting *irma_code_archive_generation* to *true*.

- vmware
  We created a separate environment file which supports VMWare, by specifying VMWare boxes instead of the default Virtualbox boxes. Those with VMWare in their name use VMWare boxes, while the others use Virtualbox.

In the Ansible folder there is also an Vagrantfile one can use to setup the infrastructure for Ansible to install onto. Vagrant orchestrates these machines using information provided in a environment file, specifically name, IP, box, CPUs and memory. Vagrant defaults to using the one named prod but by using the environment variable VM_ENV one can set it to the wanted file. To start and setup the machines one simply needs to:

```
export VM_ENV="name of environment file" # optional
vagrant up
```

Using the prod environment, four separate machines are orchestrated where two of them are probes, one for Linux and one for Windows (prod.yml code: G.34). In this stage the machines only contain what is defined through the boxes made by Packer (section: 4.3.5).

In IRMA, Vagrant reads information from the necessary environment files to create the specified machines. This includes: IPs, names, files to share, and some resource allocation. Additionally, boxes (section: 3.4.2) to base the machines upon are specified. Within IRMA, these only include configuration for allowing Vagrant to manage them, and for Ansible to provision them. As long as necessary files are provisioned and the configuration is similar to what Ansible expects, SSH, the machine would work just as well.

Before provisioning, we recommend adding the machines to known hosts as Ansible can't do so automatically. Without doing it is needed to manually type in *yes* during the provisioning which we have experienced as buggy, this happens because IRMA relies on ssh which questions the user when connecting to unknown hosts. The system can then be provisioned and started using irma-ansible.py which reads information from necessary files and starts the Ansible playbooks. The first time running the installation could take some time, about 15 minutes without any extra antiviruses enabled, but subsequent installations takes considerably less time. More exact numbers can be found in testing and analysis under section 5.2
The command for starting the provisioning is:

```
Python irma-ansible.py environments/prod.yml playbooks/playbooks.yml
```

Note that the same environment file must be used so that the infrastructure Ansible tries provisioning to is the same as the infrastructure actually orchestrated. The playbook playbooks.yml initiates these playbooks in the order listed:

- provisioning.yml fixes all dependencies for the machines
- updating.yml updates the analyser probes through their role ansible descriptions in their role
- deployment.yml deploys the IRMA code and sets it up which includes starting necessary services

These can also be run separately by swapping playbooks.yml with the wanted playbook in the previous command.

After provisioning the whole infrastructure should be up and running and is accessible through

```
http://<the IP specified in the environment file for frontend>
```

which should look something like the image shown in the appendix I.39. The API documentation will be available under:

```
http://<the IP specified in the environment file for front end>/swagger/
```

It is important to note that for some reason the site won't load if one doesn't include the forward slash at the end of the url.

4.1.3 Available configuration

There are some additional configurations available beside the environment files, Vagrantfile, group vars, and host vars. These are for the modules individually (front end, brain, and probes). Among the more important parts of this is the configuration for a syslog. [3] This is a boolean which allows for logging to syslog. The celery configuration is also important, especially setting a max limit of of simultaneous celery workers. There is also an optional SSL setup for the system. The configuration is well documented on the irma.readthedocs. The relevant configuration files are available under **<module name>/config/<module name>.ini**[4]

An issue we experienced is that when starting a large scan made up of a few hundred tasks (a task can be for one probe to scan one file). All the tasks are created and start as per usual, but within a few seconds the progress slows to a halt. This ends with a large amount of tasks suddenly returning a timeout error. We believe it comes from a mixture of not having a limit on the number of available celery workers, and a relatively low timeout value. This likely results in resource starvation. As more and more tasks are dispatched to the probe machines, the resources are simply used up by the celery workers and they do not manage to finish the work load before timing out. Therefore, it is important to set a max limit of celery workers through the variable *concurrency* to what the hardware is actually capable of supporting.

4.1.4 Analysers

The analysers are split into a few different categories, or probe types as its called in IRMA, with the same name as the folder the probe is in e.g. antivirus and database, this can be seen in the folder structure in figure 4.5.

IRMA have created skeleton templates following the required configuration for probes[5]. Each analyser is usually compromised of at least two files, the plugin.py and the analyser specific file, e.g. nsrl.py. The plugin file handles the interaction between IRMA and the analyser, through using calls from the analyser specific file, testing whether it works and using it. It also specifies potential requirements and registers the analyser to a plugin manager. The analyser specific file acts as an interface between the actual analyser e.g.

[3] Syslog: https://en.wikipedia.org/wiki/Syslog

[4] IRMA configuration: https://irma.readthedocs.io/en/latest/admin/index.html

[5] Analyser skeleton: https://irma.readthedocs.io/en/latest/extending/add_probe.html

anti virus and IRMA, standardising necessary calls.
Analysers can be tested through:

```
cd ansible
vagrant rsync # transfer files necessary
vagrant ssh
sudo su deploy
cd /opt/irma/irma-probe/current
venv/bin/python -m extras.tools.run_module
venv/bin/python -m extras.tools.run_module my_module file # if previous command shows it as available
```

Figure 4.6: how to test new analyser

After having tested whether the analyser works or not we recommend also adding provisioning of the necessary requirements for the analyser. In IRMA this is done through roles, one can add this in the ansible/roles directory as a separate folder. Even better, if one adds the role in ansible-requirements.yml in the ansible directory the role would be automatically pulled from a specified git repository when the provisioning command is run. It would look something like this:

```
- src: https://github.com/dummy/dummy-example.git # repository for the role
  version: v1.2.4 # optional
  name: dummy.example
```

Figure 4.7: template for role in ansible-requirements

If one then adds it to provisioning.yml under ansible/playbooks in the following format:

```
- name: Example provisionning
  hosts: example
  roles:
    - { role: dummy.example, tags: example }
```

Figure 4.8: example of role in provisioning.yml

The role can be added as with other analysers in the environment file. Some of the already supported analysers lack existing provisioning, but this can be extended through the method mentioned, by creating a new role.

4.1.5 Problems with the setup

We have met some issues when following the installation guide [6]. In general there were a lack of provisioning on some probes, updating of database signatures for anti viruses

[6] IRMA installation: https://irma.readthedocs.io/en/latest/install/automated/index.html

seeming to time out, and some issues arising during download of files from the Internet.

RabbitMQ

We had an issue where the retrieval of RabbitMQ installation files seemingly fails. The issue could be fixed temporarily by curling the site from the machine (brain) before the provisioning was run. The issue seemed to be caused by a cloudflare on the official site, and could be fixed by setting the role variable *rabbitmq_os_package* to true, which makes the role install the package through apt from an OS repository instead.

Avast installation

During installation of Avast a software requirement was not met. Avast requires the package named libssl1.0.0. libssl1.0.0 is not available in newer version of some Linux distributions, one of them being Debian 9. Since we used Debian as our base image we added a task to download and install the libssl1.0.0 from an older version of Debian if the playbook is ran on a Debian machine. Our proposed solution can be seen in Figure 4.9

```
- name: Downloading libssl1.0.0 (for Debian 9.0 (stretch) or newer
  get_url:
    url: http://security-cdn.debian.org/debian-security/pool/update
         /main/o/openssl/libssl1.0.0_1.0.1t-1+deb8u11_amd64.deb
    dest: /tmp/libssl1.0.0_1.0.1t-1+deb8u11_amd64.deb
    mode: 0644

- name: Install libssl1.0.0
  become: true
  become_method: sudo
  command: dpkg -i /tmp/libssl1.0.0_1.0.1t-1+deb8u11_amd64.deb
```

Figure 4.9: libssl1.0.0 tasks added to Avast role

4.1.6 Automatic provisioning of antivirus and tools

IRMA has a total of 22 supported analysers, 1 database of known good/bad hashes (NSLR) and 4 supported metadata analysers. IRMA can automatically enable six of those supported analysers, and one metadata analyser. They are enabled when provisioning with the default Ansible roles by adding them to the environment file being used. In Ansible, a role can be used to break up a playbook into directories with their own files, variables and tasks. [7] This helps breaking down complex playbooks into different reusable parts. A role in IRMA is referenced to in the yaml provisioning file and in the environment file. For some of the supported analysers IRMA has not made Ansible roles. Adding some of this provisioning is some of the work we have done.

4.1.7 Activation of analysers and possible bug fixes

Often the provisioning will fail during a downloading or updating task. This can be bypassed by running the provisioning again or by using a local mirror which is discussed in section 6.3.4.

[7] Ansible role: https://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse_roles.html

The following anti viruses works with automatic provisioning without any changes

- McAfee VirusScan Command Line scanner (Windows)
- Clam AntiVirus Scanner (Linux)
- F-PROT Antivirus (Linux)
- eScan Antivirus (Linux)
- FSecure Antivirus (Linux)
- AVG AntiVirus Free (Linux)
- Trid (Metadata analyser)

Antivirus requiring license keys/files

- Bitdefender (Tested with free trial and confirmed working)
- Zoner
- Dr Web
- Avast
- Kaspersky
- EsetFileSecurity

The file to add license keys to can be found in playbooks/ either in host_vars/probe.irma.yml or one of the files in playbooks/group_vars/. If you are given a license key IRMA will most likely have a files directory the file needs to be put in.

Windows AV's without roles implemented

The following Windows AV's are according to IRMAs documentation supported but does not have any roles implemented, meaning they are not provisioned for.

- Avira
- G Data Antivirus
- Kaspersky Internet Security
- Sophos Endpoint Protection
- Symantec Endpoint Protection

Emsisoft Command Line

When first running Emsisoft during provisioning we got this error:

```
"rc": 2, "start": "2019-05-17 06:49:12.642023", "stderr": "", "stderr_lines": [], "stdout": "Please always use the /s command when running Emsisoft Commandline Scanner from within the same folder that contains a
n active Emsisoft Service component or shut down the Service if it is no longer needed.\r\n", "stdout_lines": ["Please always use the /s command when running Emsisoft Commandline Scanner from within the same fol
der that contains an active Emsisoft Service component or shut down the Service if it is no longer needed."]}
```

This was easily fixed by adding a /s to the command in quarkslab.emsisoft_a2cmd_windows/tasks/update_auto.yml

```
- name: Install a2cmd signatures
```

```
  win_command: "\"{{ remote_programfiles }}/{{
↪   emsisoft_win_install_path }}/a2cmd.exe\" /u /s":
```

Yara

IRMA had implemented an Yara analyser, but this lacked provisioning. We added this to through a role with automatic update on each run. The requirements Yara needed to work was

- yara-python
- Yara 3
- Yara rules

We also added a method to add rules from a specified git repository in this role:

```
- name: Yara | add rules
  get_url:
    url: "{{ yara_rules_url }}"
    dest: "{{ yara_rules_path }}/yara_rules.yar"
  become: yes
  become_user: root
```

The repository where the rules are retrieved from can be updated by setting *yara_rules_url* in the environment, it is by default set to:

`raw.githubusercontent.com/Yara-Rules/rules/master/Antidebug_AntiVM/antidebug_antivm.yar`

Metadata analysers

IRMA uses several metadata analysers. These can easily be enabled by running: "*pip install -r*" on the path to the requirements.txt from the irma/probe directory. For example: pip install -r modules/metadata/lief/requirements.txt You can do this on the following analysers:

- Lief
- Static Analyser
- Peid

We have provided a playbook for activating all three metadata analysers that can be run with the command: ansible-playbook metadataProvisioning.yml. KDA still needs to edit the path corresponding to their own infrastructure. We have tried to merge the playbook commands into the different metadata analyser roles, but during provisioning the roles were unable to find the requirement files they needed to install.

clamAv

```
fatal: [avs-linux.irma]: FAILED! => {"changed": true, "cmd": "freshclam
↪   --quiet", "delta": "0:00:00.100503", "end": "2019-05-15
↪   15:21:53.806114", "failed_when_result": true, "msg": "non-zero
↪   return code", "rc": 1, "start": "2019-05-15 15:21:53.705611",
↪   "stderr": "", "stderr_lines": [], "stdout": "", "stdout_lines": []}
```

We were not able to interpret this error but a possible fix could be to halt the running Vagrant VMs and start them up again with vagrant up.

When we have managed to provision ClamAv it has worked for a short while, but after a short while only returns errors. When doing troubleshooting we found this error:

```
clamav: ERROR: Could not connect to clamd on LocalSocket
 ↪  /var/run/clamav/clamd.ctl: Connection refused'
```

The troubleshooting was done by connecting to the probe machine running ClamAV and debug as specified:

```
sudo su deploy
cd /opt/irma/irma-probe/current
./venv/bin/python -m extras.tools.run_module ClamAV requirements.txt
```

We suspect the reason for this is a short timeout number (30 seconds) on the clamd process shutting it down after a short time of inactivity. [8] Note that the error is somewhat inconsistent, so this might not be the reason.

Lacking memory

```
fatal: [avs-linux.irma]: FAILED! => {"changed": true, "cmd":
 ↪  ["/opt/f-prot/fpupdate"], "delta": "0:00:01.545158", "end":
 ↪  "2019-05-16 13:50:47.572828", "msg": "non-zero return code", "rc":
 ↪  1, "start": "2019-05-16 13:50:46.027670", "stderr": "Error: Update
 ↪  - Out of memory", "stderr_lines": ["Error: Update - Out of
 ↪  memory"], "stdout": "Downloading update (\%100)", "stdout_lines":
 ↪  ["Downloading update (\%100)"]}
```

We got an error during provisioning that it could not allocate memory. We also got a similar error at other points during testing of provisioning. This is probably due to several previous failed provisioning attempt. Our quick fix was to reboot the VM's with vagrant halt and vagrant up to free up memory. This is can potentially be avoided by initially allocating more memory and not running all AV probes on the same host. If the recommended infrastructure (section 6.3) is followed this problem should not arise to begin with.

VirusBlokAda

VirusBlokAda gives an error during provisioning. The error says the following:

```
fatal: [avs-linux.irma]: FAILED! => {"changed": false, "dest":
 ↪  "/tmp/vba32/vbacl-linux.tar.gz", "msg": "Request failed",
 ↪  "response": "HTTP Error 404: Not Found", "state": "absent",
 ↪  "status_cod··················
e": 404, "url":
 ↪  "http://anti-virus.by/pub/vbacl-linux-3.12.26.4.tar.gz"}
```

If we follow the link described in the link we get a permission forbidden error. By further investigating VirusBlokAdas website we are unable to find a linux antivirus installation file and conclude that IRMAs documentation here is outdated and that VirusBlokAda is no longer supported. We did not find the same anti virus anywhere else either so we could not provide the role with a new url.

[8] clamd dokumentation: https://manpages.debian.org/testing/clamav-daemon/clamd.conf.5.en.html

Sophos

Sophos for Linux is a free antivirus that is not provisioned by IRMA. In its Ansible role in ansible/roles/quarkslab.sophos/defaults/main.yml you can see some hard coded variables that must be edited. KDA must most likely edit those themselves. The ones that the bachelor group changed was sophos_archive_url and sophos_archive_name. We found a linux installation for Sophos at Sophos website and copied its link address into sophos_archive_url and its file name into sophos_archive_name. When we tried running the provisioning playbooks we got this error:

```
fatal: [avs-linux.irma]: FAILED! => {"changed": false, "msg": "Failed
 ↪  to find handler for \"/tmp/sophos/sav-linux-free-9.tgz\". Make sure
 ↪  the required command to extract the file is installed. Command
 ↪  \"/bin/tar\" could not handle archive. Command \"/usr/bin/unzip\"
 ↪  could not handle archive."}
```

After some research we discovered that the public download link requires human interaction to start the download. A solution can be to download and locally host the package, or get an official link form Sophos.

The original task in the Sophos role Quarkslab used an ansible specific function to unzip the download file which failed.

```
fatal: [avs-linux.irma]: FAILED! => {"changed": true, "cmd": ["tar",
 ↪  "-xvzf", "/tmp/sophos/sav-linux-free-9.tgz", "/tmp/sophos"],
 ↪  "delta": "0:00:00.003650", "end": "2019-05-16 10:23:44.852778",
 ↪  "msg": "non-zero return code", "rc": 2, "start": "2019-05-16
 ↪  10:23:44.849128", "stderr": "\ngzip: stdin: not in gzip
 ↪  format\ntar: Child returned status 1\ntar: Error is not
 ↪  recoverable: exiting now", "stderr_lines": ["", "gzip: stdin: not
 ↪  in gzip format", "tar: Child returned status 1", "tar: Error is not
 ↪  recoverable: exiting now"], "stdout": "", "stdout_lines": []}
```

For this reason we suspected that the file could not be properly unzipped, so we replaced the Ansible function with a bash command that would unzip the file. The proposed soultion can be seen below:

```
command: tar -xvzf "{{ sophos_temp_dir }}/{{ sophos_archive_name }}"
 ↪  "{{ sophos_temp_dir }}"
```

It is also possible that this happened because of the first error we encountered, that a proper archive were never downloaded.

To summaries the three possible solutions could be:

- Host the downloaded file locally and edit the hard coded installation directory.
- You can download the file and host it on a cloud platform.
- You can download the file and put it into the files directory in the Sophos role and copy the files with the Ansible native rsync or copy command.

4.2 Configuration and set-up of Cuckoo

For consistency across components of the finished product, among other things, we decided to go with ansible as the configuration management system for configuring and managing Cuckoo. There are several option available. Several were tested with a varying degree of success. In the end the repository created by Github user julianoborba[9] was chosen as our basis. As of writing this report a pull request is not created from our fork [10] with all the latest and relevant updates.

4.2.1 Adaption for our use case

The original Ansible playbook was developed with full support for only Virtualbox. It had machinery files already added but needed roles for handling and installations if needed. During development we used VMWare Workstation as our virtualiser. It was chosen based on simplicity to manage and maintain during our development period. After some time we realised it was not the best fit for more advanced automation. In the end to much time was used on fully automating Cuckoo with VMWare workstation and full automation was not achieved either.

The end product probably will run in a production environment running on ESXi/vSphere. A role ESXi was started but not completed see section 4.2.3 for more information and the suggested code.

4.2.2 Installation

NOTE: It should be mentioned that the source code on GitHub could have been changed after submission of the report, look there for possible updated information.

Requirements/Preparations

Hardware
There is little to none information from Cuckoo officially or the team behind it on hardware requirements or recomendations. Using a bit of common sense and comments from Cuckoo users on reddit [11] it is safe to assume that a proper production environment needs at the very least these specifications per guest VM:

- CPU: 2 (At least 1 per running VM)
- RAM: 2GB (4GB for Windows VM, 2GB for Linux)
- Storage: 60GB (anti-anti VM measures, simulating a realistic environment etc.)

Taking into consideration that these are recommendations for one guest virtual machine. You can get away with lesser hardware but these are our recommendations based on VMWare, Microsofts recommendations for Windows as well as what others have said they use. These are probably not optimal cost/performance specifications and further testing could be beneficial. Resources for the Cuckoo host is not a part of the list above, which will also vary depending on which processing modules are added and installed. It is also dependent on if all, none or some of the Cuckoo infrastructure is hosted on the same machine/server. E.g Elasticserach/logging as a separate service, hosted on its own machine or on the Cuckoo host etc.

[9] Origin playbook repository: https://github.com/julianoborba/Ansible-Cuckoo

[10] Ansible Playbook for Cuckoo: https://github.com/knaku/ansible-cuckoo

[11] Reddit comment: https://www.reddit.com/r/Malware/comments/5z8gu1/cuckoo_sandbox_hardware_recommendations/

Software

Ansible Provisioner

On the Ansible provisioner, ansible and the git repository with the playbook are obvious requirements. It is also necessary to have all the correct information regarding the systems Cuckoo will be installed on, and potential guest systems as well as root access.

Installation

Step 1 - Clone the Ansible playbook repository

```
git clone https://github.com/knaku/Ansible-Cuckoo.git \\
```

To install Ansible, run the install_ansible.sh in the root folder of the newly cloned git repository, or copy and run the commands below.

```
apt update
apt -y install apt-transport-https software-properties-common
apt-add-repository ppa:ansible/ansible -y
apt update
apt -y install ansible sshpass
```

If you don't already have git installed run:

```
apt update
apt -y install apt-transport-https software-properties-common
apt install git
```

Step 2 - Prepare Cuckoo host

Ubuntu 18.04 server/machine, 16.04 and 17.04 should work as well but have not been tested. The playbook was created for Ubuntu but should work on most Debian based distributions as well if you update the roles accordingly, it might be enough to update to use correct repositories. Other Linux distributions should also work fine with a bit more adjustments.

Ansible requires Python and a SSH-server to run. If you use a regular desktop Ubuntu there is a good chance that both are already installed.

Run these command in you terminal to install the needed requirements:

```
apt install -y openssh-server python
```

If you want Ansible to connect with the user name and password of the account, use those. If you want Ansible to connect with SSH-keys, simply add your public key to the Cuckoo host.

Step 3 - Cuckoo guest image

You need a pre-configured Windows, Linux or macOS fulfilling the requirements given by Cuckoo [12]. You can see a quick summary of Windows requirements below:

[12] Cuckoo software requirements: https://cuckoo.sh/docs/installation/guest/requirements.html

- (Optional) All anti-anti VM counter measure configured and installed
- (Optional) All user programs (Microsoft Office, Adobe Acrobat/Reader etc.), and other requirements to emulate a normal environment.
- Installed Python.
- agent.py that starts automatically, preferably with administrative rights. agent.py can be started manually when creating the snapshot. It is not necessary for it to start automatically but it makes repeatablity a lot simpler. Use .pyw instead of .py if you want the terminal window to be minimised.
- Turn off the Windows firewall.
- Turn off Windows updating.
- Set a static IP. Cuckoo cannot handle DHCP yet. If you leave everything as default after cloning set it to 192.168.56.111 with a default gateway of 192.168.56.1.
- Snapshots:
-
    - You can create a snapshot manually while the VM is running with requirements above
    - You can let Ansible handle it automatically (mostly if using VMware Workstation). Agent.py is required to start on startup for this to work.
    -
        - With Virtualbox you can let the playbook handle everything automatically.
        - With VMWare Workstation you must manually start and stop the VM during the play if a snapshot does not already exists. If a snapshot exists it might create a bit of issue with Cuckoo and VMWare.

Step 4 - Installing and running Cuckoo

Replace the information in ***cuckoo-playbook/inventories/production/hosts*** with the correct one for your Cuckoo guest.

- HOST is the IP address of the server to install Cuckoo to
- ADMIN is a user with sudo privileges on the server
- PASSWORD is the user ADMIN password

Now run this command in the terminal of your ansible-provisioner to start the play, look below for explanation for a description of the different variables after –extra-vars:

```
ansible-playbook -i inventories/production site.yml --extra-vars "distribution=bionic nic=ens32 vmwareNetworkAdapter=1 license=xxxxx-xxxxx-xxxxx-xxxxx-xxxxx"
```

- "distribution" is the Ubuntu distribution
- "nic" is the nic on the cuckoo host to use in the routing.conf
- "vmwareNetworkAdapter" is vmnet suffix used with the cuckoo guest. Only enter the value after vmnet. e.g for vmnet1: vmwareNetworkAdapter=1. vmnet1 is usually host-only while vmnet8 is usually NAT. If you don't set a custom vmnet it usually ends up using vmnet8. This is only necessary to add when using VMware workstation
- "license" is the license for VMWare Workstation and is only to add when using VMWare workstation.

You might want to look over and verify or change the different variables in some of the configuration files since they are hard coded and might not reflect the values you have in you environment. Directories and files to check are "cuckoo-playbook/roles/[name of role]/files/*.conf" The most relevant roles to check are Cuckoo and VMWare, the others might be relevant depending on your problem but have not been changed by us. Look at the bottom of the git repository [13], under "Things that need manual changing if you change the default" for most relevant variables that might need changing.

4.2.3 ESXi/vSphere support

Due to the decision of using VMWare Workstation as our virtualiser and time restrictions a fully functioning and automatic role were not created. An adapted role from VMWare Workstation were created. The only missing parts should be the remaining API calls for orchestrating the Cuckoo guests as well as potentially creating a snapshot if not already created. It should be mentioned that this is not tested code as is simply suggestions and should be adapted accordingly. Text in *italics* should either be replaced or are suggestions.

```
---

- name: Install pyVmomi package
  become: true
  become_method: sudo
  pip:
    name: pyvmomi
    state: latest

#check that script has required permissions with ESXi/vSphere
- name: Check permissions
  command: \textit{write API call here}
  register: \textit{permission}
  failed_when: \textit{permission == false}  # adapt this to the correct response from the API
```

Figure 4.10: ESXi/vSphere suggested role

[13] Ansible Playbook for Cuckoo: https://github.com/knaku/ansible-cuckoo

```
#check if a preconfigured image for Cuckoo guest is available  on ESXi/vSphere host for provisioning
#\textit{it is also possible to check for an already provisioned and configured machine}
#  use getallvms.py in
#  use \textit{class com.vmware.vcenter_client.ResourcePool(config) list}
- name: Check for Cuckoo image / Check for Cuckoo VM
  command: \textit{write API call here}
  register: \textit{exsisting_vm}
  failed_when: \textit{True not in existing_vm} # adapt this to the correct response from the API

#provision new Cuckoo guest from previously found image, or verify connection to already configured machine
class com.vmware.vcenter_client.ResourcePool(config) create
- name: Provision new Cuckoo machine / Connect or power on machine
  command: \textit{write API call here}
  register: \textit{provisoned}
  failed_when: \textit{True not in provisoned} # adapt this to the correct response from the API

- name: Copy Cuckoo configuration file to Cuckoo host
  become: true
  become_method: sudo
  become_user: "{{ cuckoo_user }}"
  copy:
    src: "{{ item }}"
    dest: "/home/{{ cuckoo_user }}/.vsphere/conf/"
    mode: 0644
  force: yes
  with_fileglob:
    - vsphere.conf

    # remember to customise this to the proper network interface for ESXi/vSphere
- name: Configure \textit{network interface} to be up
  become: true
  become_method: sudo
  command: ifconfig \textit{network interface} 192.168.56.1 netmask 255.255.255.0 broadcast 192.168.56.255 up

#check for existing snapshots of windows/linux guest
#   see \textit{snapshot_operations.py --list_all} at GitHub in link below figure
- name: Check for existing snapshot
  command: \textit{write API call here}
  register: \textit{exists}
  failed_when: \textit{exists == false}  # adapt this to the correct response from the API
```

Figure 4.11: ESXi/vSphere suggested role

```
#start the machine if no snapshot exists
- name: Start Cuckoo guest
  command: \textit{write API call here}
  register: \textit{started}
  failed_when: \textit{started == false}  # adapt this to the correct response from the API

#wait time should be between 2-3 minutes for Windows guests, it can probably be shorter for a Linux guest
- name: Wait for machine to complete booting
  pause:
  minutes: 3

#create a new snapshot if one does not exist
#   see \textit{create_snapshot.py} at GitHub in link below figure
- name: Create snapshot of the Cuckoo guest
  command: \textit{write API call here}
  register: \textit{created_snapshot}
  failed_when: \textit{created_snapshot == false}  # adapt this to the correct response from the API

- name: Wait for machine to complete booting
  pause:
  minutes: 1
```

Figure 4.12: ESXi/vSphere suggested role

Link for pyvmomi community samples mentioned in Figure: 4.12vmware/pyvmomi-community-samples

4.3 Changes made

From the requirements section KDA wanted several ways to conduct an analysis in an automated infrastructure. Everything from static analysis to known good/bad, Yara and more. Thus, the bachelor group searched for tools/software that could be used in an infrastructure to best solve for KDAs requirements. For the analysis platform we chose IRMA. We have installed and configured IRMA with some troubleshooting required. Much of this is described earlier in the report.

NSRL has been implemented to complement the known files functionality of IRMA. While IRMA's known files only applies to already scanned files, NSRL contains a huge database of hashes with some information about each. IRMA originally had partial support for NSRL but it was outdated and the bachelor group had to write new Python3 code for this. More about this is specified later in the report (Section

IRMA has support for automatic provisioning of antivirus but much of this comes with errors that the bachelor group has been working with debugging. The bachelor group provides in the report an overview of antiviruses that is supported by IRMA with an overview of bug fixes.

KDA also wanted dynamic analysis. There were several alternatives the bachelor group considered but we ended up choosing Cuckoo for our dynamic analysis platform. We also had a goal of implementing Cuckoo as a probe of IRMA. Implementing Cuckoo was a fairly large task, that took some time, it is such a big task that a completely different bachelor project is implementing and setting up an automatic setup for Cuckoo. The bachelor group implemented Cuckoo as a probe of IRMA.

After the requirement specification phase was over, the bachelor group came up with an idea to implement a pipeline for the infrastructure, which KDA became very interested in. The logic for how it can be implemented is completed, but it was not near enough completion to be pushed.

4.3.1 Pipeline

Scanning in IRMA is very inefficient. By default files already analysed are re-analysed and almost all probes are used for every file. The only method employed without change is MIME-type filtering. This filters out which metadata probes are used, depending on the file type scanned.

To understand how this can be improved upon one needs an understanding of how it currently works. Scanning tasks are queued through usage of Python Celery:

> "*Celery is an asynchronous task queue/job queue based on distributed message passing. It is focused on real-time operation, but supports scheduling as well.*
> *The execution units, called tasks, are executed concurrently on a single or more worker servers using multiprocessing. Tasks can execute asynchronously (in the background) or synchronously (wait until ready).*" [14]

The scan workflow is as follows[15]:

[14] Celery: http://www.celeryproject.org/
[15] scan workflow: https://irma.readthedocs.io/en/latest/technical/scan_workflow.html

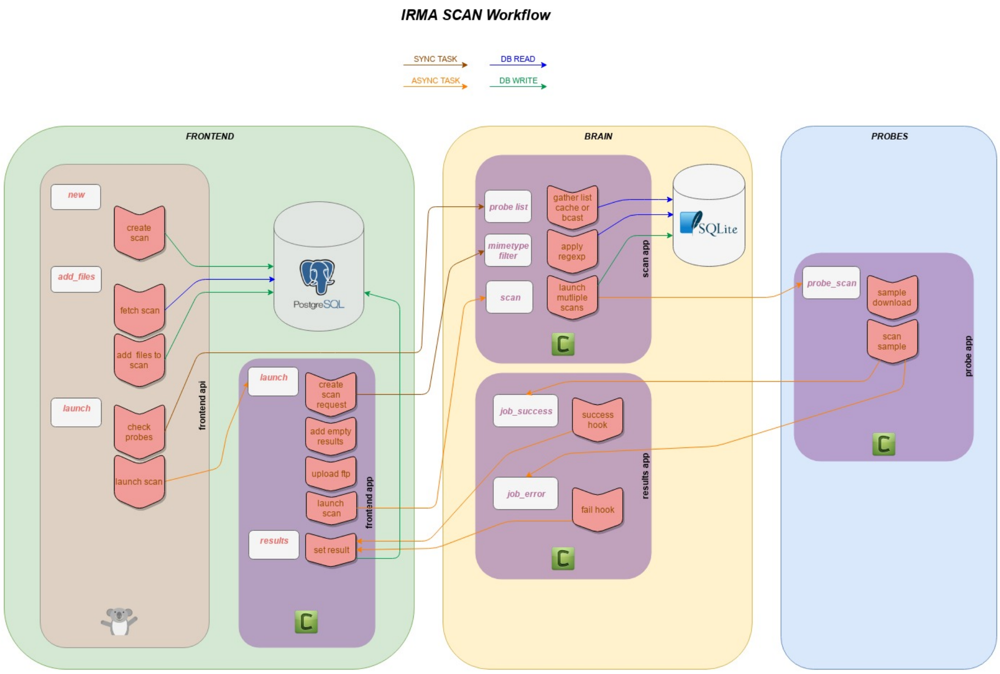


Figure 4.13: scan workflow

1. **Frontend API**
    1. A request is sent to the front end API or made through the GUI. The request format can be seen in the appendix (H.37)
    2. This creates a scan object in the PostgreSQL database which stores the parameters
    3. Files are uploaded, and stored in the file system, to the front end and registered in the same database
    4. The scan is added to the front end celery queue as an asynchronous task
2. **Front end Celery**
    1. Which probes to be used are selected based on scan options and mime type filtering
    2. Empty results are stored in the database, one each file for each probe. Some of these results can already be present depending on whether the force scan option is true or not, by default it is true.
    3. Files are uploaded to the file server
    4. A scan task is created on the brain in the celery queue for each file, a probe list is also added to this scan task.
3. **Brain celery**
    1. The scan job is stored in the local SQLite database for tracking jobs
    2. Files are sent to probes selected, for each two callbacks are set, one for success and the other for failure
4. **Probe Celery**

   1. The scan task is received with a file id
   2. the file is downloaded from the file server using the provided id
   3. file is scanned
   4. results are sent to Brain using one of the two callbacks (success or failure)

5. **Brain Celery**
   1. Successful results are marked as complete in the SQLite database and are forwarded to the front end
   2. Failed results are marked as complete and an error message is generated and sent to the front end
6. **Front end Celery**
   1. Results are received, one for each file and probe
   2. result is updated in PostgreSQl database
   3. If the scan is finished a scan flush task is created on the brain to delete the files from the file server

In simpler terms, files are uploaded and a scan job is started from the user and sent to the front end. The front end uses parameters and a probe list to send a scan job for each file to the brain. The brain creates a scan job for each file and probe, sends these to the probe, with the requested analyser on it, and listens for response. Lastly the brain notifies the front end when a job is complete, the front end adds the result to its database.

With some configuration, by disabling *force scan*, IRMA handles known good/bad files by simply not re-scanning known files. In combination with mime type filtration this reduces scan time slightly by reducing the number of jobs for each probe.

Sending every file to every probe is a huge waste of resources when it includes Cuckoo, it simply takes too much time. By stopping some files in earlier stages using a pipeline one can further reduce the amount of scan jobs each probe.

The pipeline can work in numerous ways, it really depends on what the purpose of the system is. If the goal is to analyse malicious files, these should be sent to Cuckoo or if the goal is to lessen time spent scanning, one should stop files as early as possible.

This all depends on what is wanted of the system, in our case the goal was to reduce the scan time as much as possible while retaining a low false negative.

The pipeline works by splitting the scan process into stages:

1. Cache lookup - Check if a file hash has been encountered previously
2. Static analysis including metadata extraction, Yara and NSRL
3. Every antivirus probe available
4. Dynamic analysis, only Cuckoo for now, but the basis for extending with others is done

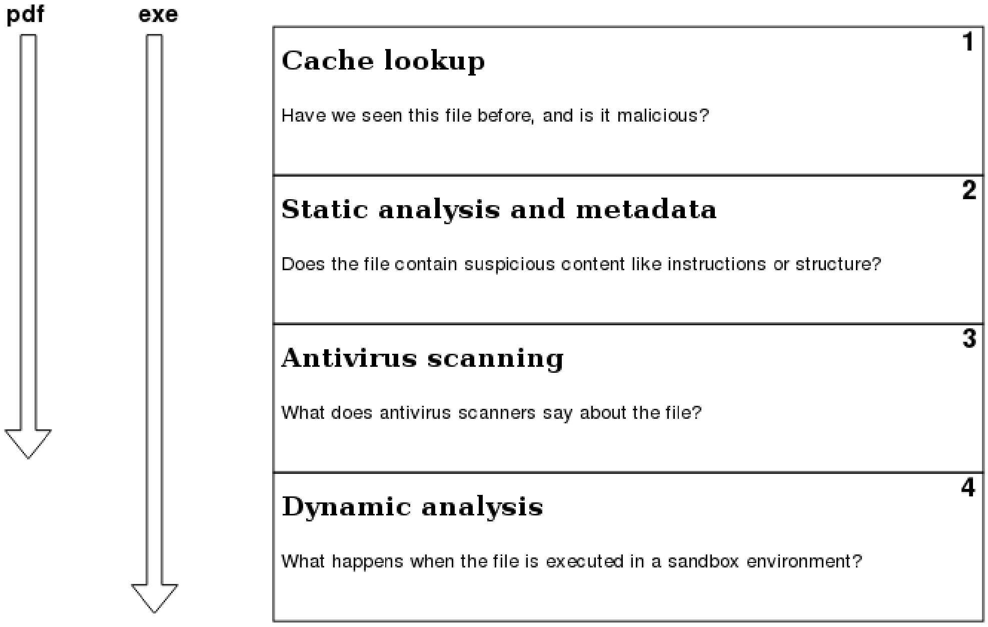

pdf
exe
1
Cache lookup
Have we seen this file before, and is it malicious?
2
Static analysis and metadata
Does the file contain suspicious content like instructions or structure?
3
Antivirus scanning
What does antivirus scanners say about the file?
4
Dynamic analysis
What happens when the file is executed in a sandbox environment?

Figure 4.14: How the pipeline can look for files

The pipeline code knows which probes to use by retrieving all available probes and their category which is a value stored in their module class. Possible categories are:

- Unknown
- Antivirus
- Database (refers to local databases such as NSRL)
- External (refers to probes contacting external services such as VirusTotal)
- Metadata
- Tools
- Dynamic (which should be added for dynamic support)

This makes it extremely easy to change the order of which probe categories are employed in. Additionally new categories can easily be added in *common/src/utils.py* and then to the class in plugin.py belonging to the probe, which allows even further customisation of the queue.

The pipeline itself is created by adding a new Boolean argument when starting new scans named pipeline. This allows a pipeline without making major changes to the code or through an external service which will explained later. Pipeline logic is added to step 1.4, where celery tasks for each file is created on the brain, in the scan workflow and works as follows when pipeline is true:

- Retrieve probe types of selected probes
- check for which stage the file should be scanned in
  - check the probe type used in the previous scan, in the SQL database, and use the next in the list, if there is none use the first

The other part of the logic is added to a function at the front end checking whether a scan for a file is finished or not. The logic checks the result of the scan and determines if it should continue to the next stage or if it is complete. If it continues to the next stage, a new scan for the file is initiated on the brain with the next probes. For each stage completed the result for the scan needs to be updated with new analysers and results in the database.

Logic to decide whether to continue

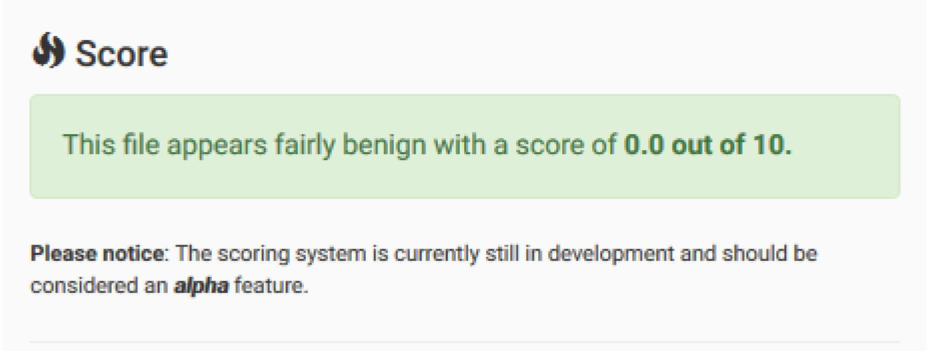
In the pipeline, results from scans have to be somewhat automatically interpreted, in contrast to the previous solution, (section 3.2.2) where interpretation is left to the end users. The question is how this can be automated.

In this implementation files are stopped if they are known, which is done before the pipeline code is even reached. The second stage is therefore always run with unknown files.

The second stage involves metadata, Yara and NSRL, which is hardly any form of static analysis as the files in NSRL are not necessarily known good and Yara having such a high rate of false positives, though this depends on the rules employed. With more methods of static detection more files can be stopped at this stage, without hardly any are stopped.

The third stage is where the majority of the files are stopped. Files should be stopped based on a percentage of how many of the analysers marks the file as malicious or not. Files determined to be malicious are stopped and those without detections, that are not executable (retrieved result from second stage) are stopped. Executable files where there are uncertainty are sent to Cuckoo, the last stage.

Cuckoo have a scoring system one can base the decision on, but a more advanced method to interpret the results would probably be advantageous for the future. The scoring system does not have a range of 1-10, but they are looking for other algorithms to use which will have a final score between 1-10 [16]. It should also be noted that the Cuckoo scoring system is not fully developed, and it should be taken as a guide rather than a definite result, see screenshot from Cuckoo's result screen at Figure 4.15.



Score

This file appears fairly benign with a score of **0.0 out of 10.**

Please notice: The scoring system is currently still in development and should be considered an ***alpha*** feature.

Figure 4.15: Cuckoo scoring

This interpretation is a very simplified view of how the logic can be done. Where

[16] Read in slack channel, search for score: https://cuckoosandbox.slack.com/

much of the data is left unused, e.g. what NSRL and Yara provides, and the system is therefore not as efficient as it could be. For maximising the efficiency while maintaining a low false negative rate, the algorithm should be modified through testing and analysis.

Possible alternative methods

Alternative ways to implement the pipeline was considered. It could as an example be handled through an external application which started a separate scan for each stage and file in the process. The difference would be that a listener would be required to wait for signal informing that a scan is done. An alternative to this can be checking if the scan is complete every 5 seconds, but this would create unnecessary load. The listener would need its own worker or thread.

Currently code for checking whether a scan is complete or not, is done through isFinished() which is interlocked so that only one celery worker can run it at a time. This can lead to a performance decrease as a queue of workers that are waiting for the resource to be available might pile up. This will hopefully not happen as the function is only run in the last stage of a scan, and the checks are not especially demanding either.

If this becomes an issue a new way to do the pipeline logic has to be made. This could potentially be done by adding the checks for whether a file should continue or not, as a task in a worker queue instead of inside isFinished(), basically starting the pipeline checks asynchronously. The external code works in such a way by having the checks externally, through a queue of its own. A requirement for the queuing and external checks is some sort of parallelisation, multiple workers and a listener is required. Considered alternatives for allowing the parallelism necessary were Rebus, Celery and Python threading. *Rebus* was the alternative mentioned to us in the Quarkslab IRC when asking for advice regarding the pipeline. It is described as:

> "*REbus facilitates the coupling of existing tools that perform specific tasks, where one's output will be used as the input of others.*" [17]

The REbus implementation could look something like the list below (list 4.3.1).

With *Celery* one could simply create a similar structure as used by IRMA as a new queue, and a few celery workers performing these. This method is most probably overkill as the tasks shouldn't be very time demanding.

Using *Python threads* are too low level as they lack a queuing system, and the parallelisation is slower as well because each thread runs in the same Python process. [18]

The external methods are not viable unless the pipeline logic grows more demanding. If that turns out to be the case an external method could look something like this:

- Start a scan with analysers for the correct (if new, first) stage using the API
- Tag (section 4.3.2) it with a common id
- Receive a notification when a scan on a file is done through isFinished()
- Check if the file should continue to the next stage
    - Repeat from the start with the next stage

[17] Rebus: https://github.com/airbus-seclab/rebus

[18] Python threads: https://www.quantstart.com/articles/Parallelising-Python-with-Threading-and-Multiprocessing

    - Scan for that file is complete, no need to do anything

- Combine scans with a common tag id

What to do with the files?

If a file is deemed safe it should be sent to the previously authorised user, who was authenticated through the kiosk at the initialisation of the scan, in the Active Directory User Storage. If the file is deemed malicious, administrators should be alerted with a link to the scan report, then they can perform further analysis. A safety precaution and simplification of the algorithm (section 4.3.1) can be to not send any file to the user space from a scan containing one or more malicious files.

4.3.2 Kiosk

At the current stage the kiosk only reached conceptual design and slight prototyping, but with no finished application. The concepts are rather simple with a simple computer dedicated to running software constantly awaiting input, e.g. flash drives, USB-disks, etc, and performing user authentication and authorisation against an Active Directory-structure. Once an authorised user has been authenticated the kiosk will simply transfer the files from the input drive to the infrastructure front end to perform analysis, finally at the end display the result.

The very main advantage with the proposed approach would be that without explicit user authentication, no drive would be mounted. E.g, without having valid user credentials, accessing or contaminating the kiosk or infrastructure illicitly will be limited further by constraining this access point. This would require disabling any auto-mount feature from whatever GNU/Linux distribution is chosen. To avoid this step wholly, a minimal build like Debian Minimal could be used. As desktop Environments based on GNOME comes with auto-mount features out-of-the-box; one solution is building a text user interface with Curses[19] to fulfil kiosk interaction as it should be highly automated and only require IO-interaction for storage input and user authentication.

[19] Curses in Python3: `https://docs.python.org/3/library/curses.html`

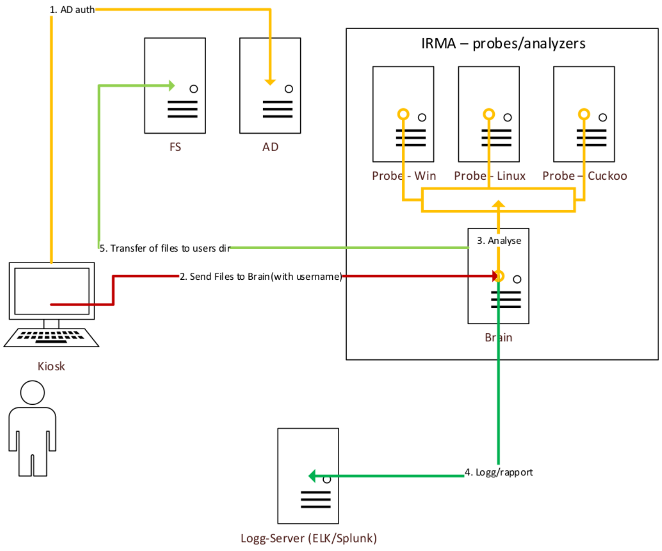


Figure 4.16: Conceptual kiosk-to-infrastructure architecture

Prototype design

An incomplete kiosk prototype was written with Python3 based on figure 4.16 infrastructure. To allow a high degree of control over detecting, mounting, and reading input drives, the design requires GNU/Linux. This is because Windows does not easily support low-level actions to the same degree.

Drive handling

The prototype code bases drive detection on the pyudev-library wrapper around the udev device manager available in the user space. Udev listens to kernel-space hardware events. By monitoring these *uevents*, it is trivial to detect added storage media devices by filtering uevents containing *BLOCK*as subsystem value, and checking the resulting device for an action parameter set to *ADD*. If that is the case, further filtering based on parameters *ID_FSTYPE* and *DEVNAME* existing and not *NULL* can be used to confirm the device as a storage media, and to automatically mount the device with the correct file system after authenticating an authorised user with an AD-structure.

After a user is authenticated and the drive is mounted the kiosk should make parallel tasks. One task returning to the listener loop, while at least one[20] should perform the file transfer straight from the input drive to the IRMA front end.

Scan tagging

At this stage, some type of identifying user credentials should also be transferred as a way of marking scans and attaching them to an owner. This would need to be accounted for in the IRMA front end by either modifying it to accept an identity parameter to apply to scans as tags, or make use of the *irmacl-module*[21] and the tags functionality that is implemented in IRMA, but never applied automatically. The use of the irmacl-module could reasonably be implemented by using it as wrapper script for the kiosk to pass the scans

[20] More than one would require the code to support parallel file transfer on both kiosk and front end. See: https://stackoverflow.com/questions/24058544/speed-up-rsync-with-simultaneous-concurrent-file-transfers

[21] irmacl repo: https://github.com/quarkslab/irmacl[22]

through before the front end. This wrapper can also take care of adding the identifying information to the scans via the aforementioned tags. This will make sorting scans much easier compared to how IRMA does it by default in the GUI as described later in Section 6.5.2.

Figure 4.17 is a prototype of a disk monitor that could serve as inspiration to implement a drive handler for a kiosk solution in Python3.

```
"""Drive handler protoype:"""
import pyudev

class DiskMonitor:
"""Monitors for storage disk activity."""

    def __init__(self):
    """Initiate uevent listener."""
        self.context = pyudev.Context()
        self.monitor = pyudev.Monitor.from_netlink(self.context)

    def uevent_listener(self):
    """Loop through block device events for self.monitor.
    returns dictionary with block path and file system type.
    (e.g. /dev/sda and ext4)."""
        self.monitor.filter_by(subsystem='block')

        # Iteration of MONITOR.poll = 'while: true'-loop
        for device in iter(self.monitor.poll, None):
            if device.action == 'add':
                """*Authorise user credentials*"""
                #Assign relevant data to dictionary
                dev_dict={'DEVNAME': device['DEVNAME'],'ID_FS_TYPE':
                 ↪  device['ID_FS_TYPE']}
                return dev_dict
```

Figure 4.17: Kiosk drive handler prototype

In appendix E.33 there is an incomplete prototype of disk mount helper functions written in Python3 using the subprocess module to dynamically mount inserted drives. One severe security flaw in this solution to consider is lacking input validation. Even though this code should never be called with user designated input, and should in every single case dynamically get a block's *DEVNAME* fetched from uevents, malicious injection could be inserted in these parameters by a malicious actor by editing the metadata of a drive. That, and the lack of any error handling are red-flag issues, and led to this prototype only being added as a bare-bones example as an appendix.

Transfer handling

The most important consideration of the transfer handling should be to effectively and securely transfer the files between the kiosk and the front end.

In the design, rsync was considered for handling file transfer due to its integrity preservation features with checking the delta between the local files and the remote files. Due to lack of experience and the time constraints of the project at the end of implementation, this feature was never realised, and will need to be implemented at a later stage. This, along with other kiosk recommendations are discussed later in Section 6.4.1

Kiosk work-flow

Figure 4.18 describes a proposed work-flow for the kiosk as a whole.

```
"""Kiosk process pseudo-code:
Constantly loop 'drive added'-event checker.
When event detected: try to authorise
If authorised within 3 tries:
    fork process.
    If child: mount, transfer, wait for results, display, kill self.
    If parent: go back to 'drive added'-event checker.
Else:
    Deny the scan request; go back to 'drive added'-event checker."""
while true:
    drive_added = check_udev_add_events()
    if drive_added:
        try authenticate_user_with_AD:
            if user_authorised:
                fork_process() or new_thread()
                if new_process/thread:
                    drive_path = mount_drive(drive_added)
                    # transfer_drive params: (from:, to:, id_to_tag_scan:)
                    scan_id = transfer_drive(drive_path, irma_frontend, username)
                    # Busy waiting check if scan result ready:
                    # (Potentially progress check and display)
                    while true:
                        sleep(30 second)
                        if scan_result(scan_id):
                            display_result()
                            return
        catch failed_authentication:
            # Retry authentication up to 3 times.
            # If still not successful then: deny request.
            while counter < 3
                counter += 1
                retry_try_block
            deny_request()
```

Figure 4.18: Kiosk process pseudo-code

Some specifics to note for Figure 4.18:

- Active Directory (AD) integration was never implemented, so it could potentially be a much more complicated task than the logical step might imply.
- The *transfer_drive* params are: *from: <dir-where-drive-is-mounted>*, *to: <ip-of-irma-frontend>*, *username: AD user-name*

- If an irma-cl based wrapper is used to pass the scans and assign identifying tags: *to: <ip-of-irma-wrapper>*
- The *sleep(30 second)* means the script checks if a scan result has arrived in 30 second intervals, to minimise busy waiting effect.

4.3.3 Cuckoo as a probe of IRMA

When adding Cuckoo as a probe we used the already existing VirusTotal probe along with the skeleton_plugin as references. Cuckoo's REST API is used to communicate with IRMA to create and view the results of the scan. Cuckoo was implemented following IRMA's guide[23]. To set the IP address of your existing Cuckoo infrastructure edit modules/external/cuckoo/config.ini to reflect your IP addresses and ports.

Since Cuckoo as of version 2.0.6 does not return the calculated score there is no way to determine within the probe whether or not a file is malicious or benign. As you can see from figure: 4.15 and the explanation above it, the scoring system is not very reliable to begin with. In other word deducing whether or not a file analysed with Cuckoo cannot be automated yet and will always need human interaction.

The files appropriate for Cuckoo are executables, files supporting executables or files that can contain executables e.g PE/ELF, PDF, Office files.

4.3.4 NSRL

NSRL is a reference library supported by NIST (National Institute of Standards and Technology) to promote efficient and effective forensic analysis of computers.

> "*The National Software Reference Library (NSRL) is designed to collect software from various sources and incorporate file profiles computed from this software into a Reference Data Set (RDS) of information. The RDS can be used by law enforcement, government, and industry organizations to review files on a computer by matching file profiles in the RDS. This will help alleviate much of the effort involved in determining which files are important as evidence on computers or file systems that have been seized as part of criminal investigations.*"[24]

It is important to specify that not all signatures necessarily are non-malicious the files are simply known software files, as quoted from NIST:

> "*The RDS is a collection of digital signatures of known, traceable software applications. There are application hash values in the hash set which may be considered malicious, i.e. steganography tools and hacking scripts. There are no hash values of illicit data, i.e. child abuse images.*"[25]

Because the files are profiled one can modify the detection method to use NSRL to determine whether a file most likely is safe or not. Something to be aware of if one actively uses NSRL for detection is that most alerts probably are false positives as some harmless software uses common files that also are used in hacker tools.

A way to base check for potential danger is to use the ApplicationType field in NSRL located in the NSRLProd.txt file which classifies the software based on functionality some examples are keylogger, Accounting, Money Management, Virus Scan, Drivers and Utili-

[23] Adding new IRMA probes: https://irma.readthedocs.io/en/latest/evolution/add_probe.html#for-a-probe-that-is-not-a-antivirus

[24] NIST: https://www.nist.gov/software-quality-group/national-software-reference-library-nsrl

[25] NIST: https://www.nist.gov/software-quality-group/national-software-reference-library-nsrl

ties. An issue with this is that it would probably require human intervention as there are obscene amounts of categories. One can automate it somewhat by creating a dictionary of potential unsafe or safe ApplicationTypes.

This field can be combined with the SpecialCode field which marks signatures as malicious, special or normal.

One of the main differences from NSRL and different databases (e.g. hashkeeper) is its focus, which is for use in law enforcement:

1. Provenance
2. Court worthiness
3. Scope
4. Illicit file data
5. Specificity of data

The module for NSRL in IRMA worked by using the four files provided by NSRL: NSRLProd.txt, NSRLManufacturer.txt, NSRLOS.txt, and NSRLFile.txt to build a leveldb database. The building of the database had to be initiated manually by downloading the compressed file, unpacking it, setting correct file path to the text files in the configuration and running a script. [26]

There is a command for creating the database that initially returned error specifying a missing module:

```
python -m modules.database.nsrl.nsrl create -t os NSRLOS.txt \
                                        /home/irma/leveldb/os_db
```

```
ImportError: No module named irma.common.utils.oopatterns
python -m modules.database.nsrl.nsrl create -t os NSRLOS.txt
 ↪  /home/irma/leveldb/os_db
Traceback (most recent call last):
...
TypeError: type() takes 1 or 3 arguments
```

Figure 4.19: Initial NSRL setup

It turned out to be an user error where the script was run by the wrong Python environment. It had to be run from the virtual environment installed for the current version of IRMA. When doing this the error was replaced by:

[26] NSRL FAQ: https://www.nist.gov/software-quality-group/about-nsrl/nsrl-frequently-asked-questions

```
Traceback (most recent call last):
...
  File "/home/deploy/.local/lib/python3.5/site-packages/leveldict.py", line 51, in __init__
    if isinstance(db, basestring):
NameError: name 'basestring' is not defined
```

Figure 4.20: basestring is a functionality in Python2 removed in Python3

Which is a result of running it in Python3 instead of 2 as basestring was a variable type in Python2 and was deprecated for Python3, the environment was installed as Python3 for some reason instead of Python2. Updating the codebase line by line was not an option as the library, leveldict, in use also was at fault. The issue could have been fixed simply by changing the version installed for the environment to Python2, but that would lead to issues in the future as Python2 is being deprecated in 2020. [27]

A new script to build and access the database was made instead. The new script uses the library *plyvel* to interact with *leveldb*, and most of the other code logic could be harvested with minor changes as mostly parts relying on *leveldict* was at fault. The code can be summed as a method to build the database and one to search for an entry using SHA1.

The need for manual setup for NSRL was also removed by adding an Ansible role for it which downloads NSRL and unpacks it. Adding the probe is then as easy as adding the role in the environment file. Provisioning with Ansible may be a dumb way to do this because the file size of NSRL is large. It might be better to build the files directly into the image, but that would add redundant files when not using NSRL.

4.3.5 Packer

IRMA uses Packer to build images that supports the installation, by enabling SSH if necessary, updating, upgrading, basic configuration of hardware and software, hypervisor guest additions, setting up SSH and enabling it. At this level one can change what distribution or OS is used for the images as the creation allowed Vagrant and Ansible to treat each machine as they were alike. Which is the goal of Packer, to fix one of Ansible main flaws, OS restrictions.
The Packer images created for this project can be split up into minimal and Cuckoo guests.

1. Minimal
   Only the bare minimum is left on these of both files and software, and the process of creating them is basically like what it was previously in IRMA. The main difference is that instead of installing guest additions for Virtualbox it installs VMWare tools. Additionally support for Ubuntu and Windows is added.
   The process for Windows is slightly different as it works differently. In broad strokes they are:

   1. Windows is slimmed down by removing tracking, Cortana functionality etc.
   2. Disabling Windows update (it is still run during install, but it is disabled so it

[27] Python2 deprecation: https://devguide.python.org/#status-of-python-branches

doesn't start automatically)
3. Enables Windows PowerShell Remoting by setting network connection to private [28]
4. Setting the password to Windows default
5. Installing Cygwin, adding SSH and opening ports for it

These changes are necessary for Vagrant and Ansible to use Windows machines similar to Linux machines.

2. Cuckoo guest
These are very different in that they have a variety of software pre-installed and different configuration. This is to best disguise them as vulnerable machines as some malware checks for vulnerabilities before execution. These images are Windows 10 and Windows 8.1 installations, Windows 7 should be included but wasn't added as no download source was found for it. These operating systems are good choices for Cuckoo guests as they are the usual targets, still at least one Linux machine should be included (section: 3.1.3).
The previous configuration is done with the exception of slimming down Windows, additionally the following are done:

   1. Software installed:

      a. Cuckoo agent
      b. Flash player version 14.0.0.145
      c. Firefox version 25.0
      d. Python 2.7.12
      e. PIL a Python library for imaging
      f. Ollydbg
      g. Ida free

   2. Windows is never updated
   3. Firewall is disabled

   The reason for adding vulnerable software is because some advanced malware won't execute if they do not detect vulnerabilities present on the system. The included software should probably be expanded on, but we have deemed that to be outside of our scope.
   This also counts for vulnerabilities in Operating Systems, which is why running multiple hosts with different vulnerable systems is advised.

The configuration for Windows machines have been somewhat problematic as there have been problems with the connectivity. The first issue was connecting to the machine during building after it had booted, it turned out to be an environmental problem that remains unresolved. Therefore the Windows boxes were simply built in another environment than those for Linux. Another issue is the time spent building the images. To build one Windows image somewhere between 1-2 hours were required which affected the configuration of these immensely as the testing was so time demanding. The time is used on first booting the machine, which takes about 20 minutes, then configuring it which required rebooting in-between some of the scripts. The next issue is met when

[28] Windows Powershell: http://blogs.msdn.com/b/powershell/archive/2009/04/03/setting-network-location-to-private.aspx

Ansible tries connecting to the Windows machines through SSH. Windows have different features which isn't directly documented stopping machines from directly SSHing into it, which also resulted in a struggle. This was solved with Vagrant by adding support for WinRM, but the Ansible configuration did not work with the connector. In that Ansible could not connect to the machines. As configuring this was not of central importance to the project, as Packer configuration is slightly out of scope, we decided on using the Virtualbox boxes provided by IRMA instead for the last part of the project.

5 Testing and Analysis

The goal for the testing is to find potential flaws in analyses, finding performance issues during provisioning and scanning, and checking if there are any confidentiality issues with network connectivity of the infrastructure and file sharing.

The efficiency (section 5.2), network connectivity (section: 5.5) and scan accuracy (section: 5.4) were tested through practical tests. By testing what happens and analysing the results.

The bottlenecks section (5.3) was written purely based on looking at what parts in the scanning and provisioning the process slowed down and looking into what the reason may be. Because of a lack of time we did not have time to perform more specific tests on the code base.

5.1 Development and testing hardware

Our development and testing platform was a laptop with an Intel i7-4800MQ and an SSD with 512GB (section 6.2). It was sufficient for a development environment, but not for a production environment or more than a bare-bones testing environment. When evaluating the results from testing it should be taken into consideration that these results may not reflect the relative performance of the platform in a proper production environment. Some problems with anti viruses failing, issues with provisioning of software, and bottlenecks is not necessarily reflective of the software but the hardware used for testing.

5.2 Efficiency of deployment

With the development hardware running a fresh Ubuntu 18.04.2 LTS install, the initial Vagrant provisioning using VirtualBox and with the IRMA production multi-VM setup can be expected to take about 4 minutes 15 seconds to complete. This environment contains four VMs in the test stage; *frontend.irma*, *brain.irma*, *avs-linux.irma*, *mcafee-win.irma*.

After the network has been provisioned by Vagrant, the IRMA specific setup is run. This setup connects to each IRMA machine with Ansible to provision and configure all necessary services and requirements to run the entire platform. This setup took 800 seconds on a clean install.

A series of raw logs of 17 individual setups of the entire infrastructure from Vagrant to a ready IRMA-ansible setup with 14 probes can be found in Appendix M.

The huge variations in setup times is inexplicable, as there seems to be no clear overarching correlation between setup times and apparent environmental settings.

	Avg. setup time
Vagrant	4m 25s +/- 20s
IRMA-ansible (no probes)	13m 20s
IRMA-ansible (14 probes)	25m +/- 4m

Table 5.12: Average setup times**??**

5.3 Bottlenecks

Improvements of the scan efficiency is always welcome, which is why a check for potential bottlenecks in the procedure is required. There are different methods one can employ to check for bottlenecks; using functionality in the system and looking for the obvious, another is to measure time spent and analysing results with different configuration.

- If a scan is started through the front end web GUI it is not initiated before all files are uploaded, which can result in waste of time. This is because IRMA expects all files to be present when starting scans.
    - **Quick connection**
      A quicker connection would reduce the time used for the upload.
    - **Asynchronous scanning**
      Through a combination of the kiosk and the proposed pipeline solution larger scans can be started as smaller sub-scans from the kiosk and use the pipeline to combine these. This is not implemented, but it is not a complicated feature to add. It can be implemented by splitting up scans in the kiosk and assigning each a tags which the the API can receive and use to later combine the scans in the Database. The combination of these scans can also be done using the tagging feature in IRMA.
- Celery Workers on machines scaling without a limit leads to all tasks running slowly and many scans experiencing time outs which is the cause for the large increase in errors when running larger scans.
    - **Limit amount of workers**
      IRMA has a built-in configuration for setting the max amount of active workers at a time, this is a separate option in each machine (probe, brain and front end). Configuration for setting a limit is mentioned in section 4.1.3.

Quicker provisioning through uncovering and solving bottlenecks is also a goal of the project.

- IRMA lacks some modularisation in its Ansible implementation leading to unnecessary tasks being done when only smaller tasks are needed.
    - **Modularise**
      By dividing playbooks into smaller sub tasks they can be combined to perform more exact tasks and therefore waste less time.
    - **Local mirror**
      By having a local mirror for what needs to be downloaded, the download time can be close to removed.

Other methods that should have been employed to find bottlenecks are looking through logs and testing with Python[1]. We did not have time to do this, and therefore, only have a overview of flaws found by using the system. Important logs to look at are those named *syslog* in /var/log. For the provisioning we recommend looking at tasks using

[1] IPython: https://jakevdp.github.io/PythonDataScienceHandbook/01.07-timing-and-profiling.html

larger amounts of time.

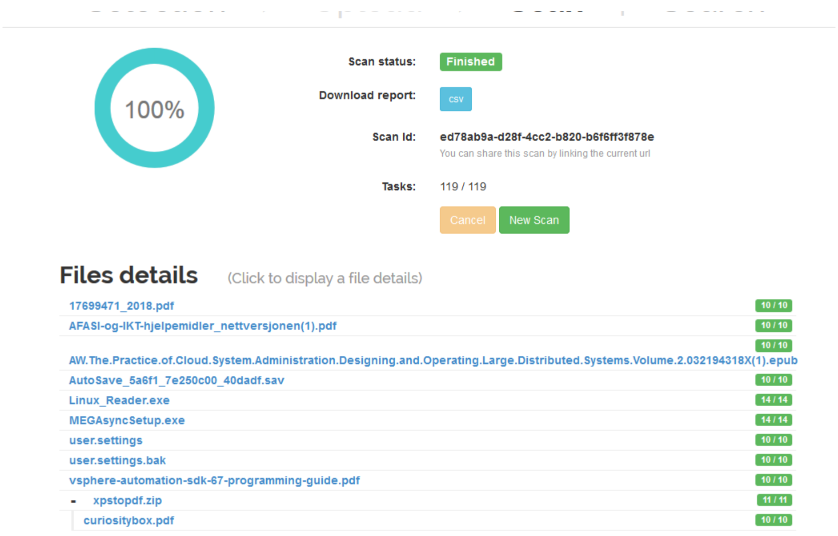
5.4 Accuracy of scans

5.4.1 Testing of benign files

The bachelor group has so far not seen any problems with false positives. False positives refers too:

> "*an error in data reporting in which a test result improperly indicates presence of a condition, when in reality it is not present*"[2]

While scanning files the bachelor group knew to be benign, the scan results always indicated the files were benign. To be sure of this the bachelor group did an analysis of 100 benign files and analysed the results. This test was done later in the development process than the scan of the malicious files in Section 5.4.2. Because of this, the benign scan was done with a slightly different set of analysers. An example scan of the result can be seen in Figure 5.21:



Scan status: Finished

Download report: csv

Scan Id: ed78ab9a-d28f-4cc2-b820-b6f6ff3f878e
You can share this scan by linking the current url

Tasks: 119 / 119

Files details (Click to display a file details)

17699471_2018.pdf	10 / 10
AFASI-og-IKT-hjelpemidler_nettversjonen(1).pdf	10 / 10
AW.The.Practice.of.Cloud.System.Administration.Designing.and.Operating.Large.Distributed.Systems.Volume.2.032194318X(1).epub	10 / 10
AutoSave_5a6f1_7e250c00_40dadf.sav	10 / 10
Linux_Reader.exe	14 / 14
MEGAsyncSetup.exe	14 / 14
user.settings	10 / 10
user.settings.bak	10 / 10
vsphere-automation-sdk-67-programming-guide.pdf	10 / 10
- xpstopdf.zip	11 / 11
curiositybox.pdf	10 / 10

Figure 5.21: Example scan of benign files

5.4.2 Testing of malicious files

The bachelor group did a statistical analysis of a selected 100 files from the open source malware repository theZoo[3] to check for false negatives. Of those 100 files, we extracted all the exe files and filled the rest of the 100 file quota with dll files. False negatives refer to:

> "*an error in which a test result improperly indicates no presence of a condition, when in reality it is present.*"[4]

We assume that all 100 files would be malicious.In total 10 antivirus scanners were used. A full list of the antiviruses used can be seen in the excel sheet provided. After the 100

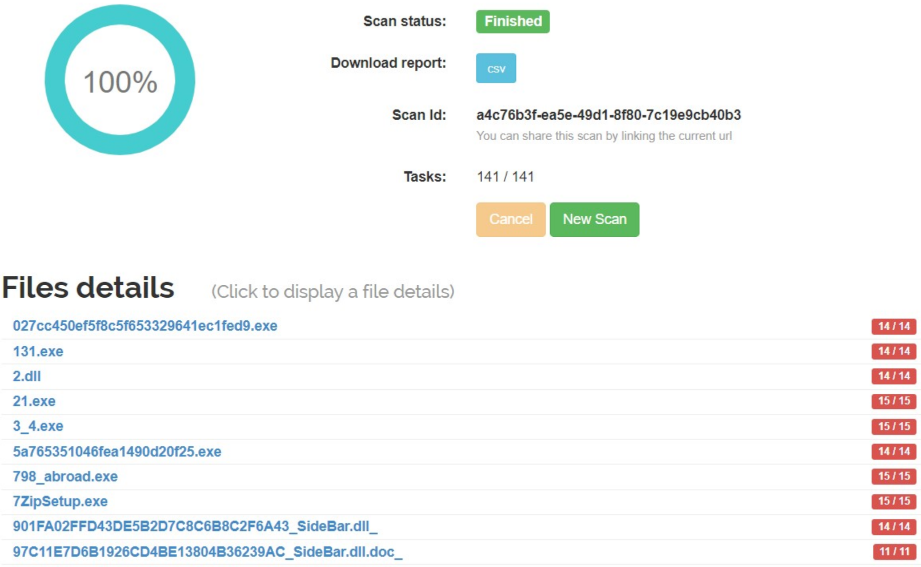
[2] False posetives/negatives wikipedia: https://en.wikipedia.org/wiki/False_positives_and_false_negatives

[3] theZoo repository: https://github.com/ytisf/theZoo

[4] False positives/negatives wikipedia: https://en.wikipedia.org/wiki/False_positives_and_false_negatives

files were scanned, their results were logged in a CSV-format. In the results a malicious file will be marked: 1, a file not detected as malicious: 0, and an error: -1. Using this the bachelor group made an excel sheet to analyse the results from the selected 100 files

The figure under shows a limited scan summary from ten of the hundred scanned files. The results look very promising.



Scan status: Finished

Download report: csv

Scan Id: a4c76b3f-ea5e-49d1-8f80-7c19e9cb40b3

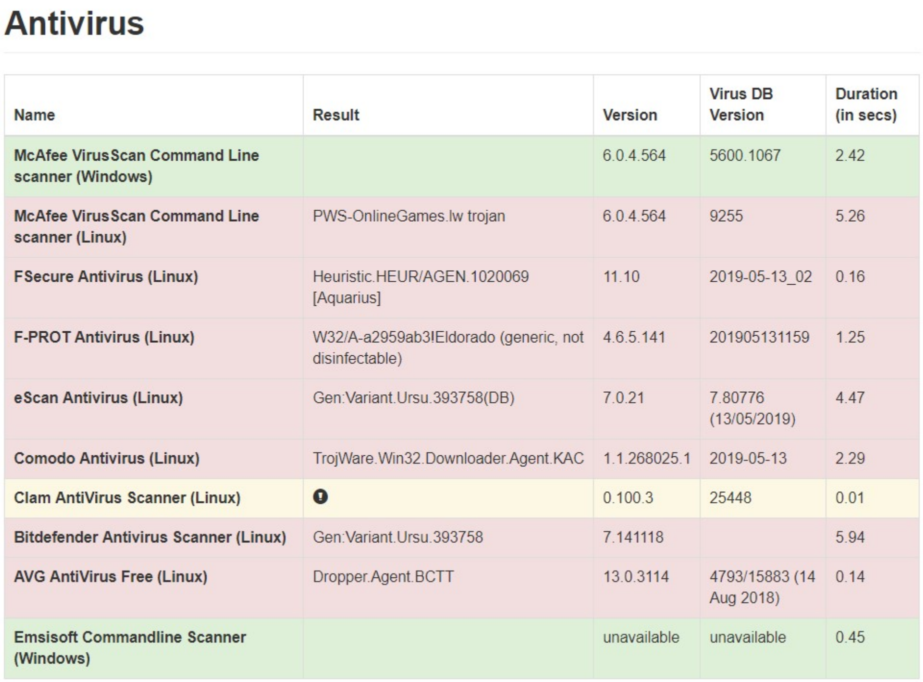
You can share this scan by linking the current url

Tasks: 141 / 141

Files details (Click to display a file details)

027cc450ef5f8c5f653329641ec1fed9.exe	14 / 14
131.exe	14 / 14
2.dll	14 / 14
21.exe	15 / 15
3_4.exe	15 / 15
5a765351046fea1490d20f25.exe	14 / 14
798_abroad.exe	15 / 15
7ZipSetup.exe	15 / 15
901FA02FFD43DE5B2D7C8C6B8C2F6A43_SideBar.dll_	14 / 14
97C11E7D6B1926CD4BE13804B36239AC_SideBar.dll.doc_	11 / 11

Figure 5.22: Summary scan of malicious files



Antivirus

Name	Result	Version	Virus DB Version	Duration (in secs)
McAfee VirusScan Command Line scanner (Windows)		6.0.4.564	5600.1067	2.42
McAfee VirusScan Command Line scanner (Linux)	PWS-OnlineGames.lw trojan	6.0.4.564	9255	5.26
FSecure Antivirus (Linux)	Heuristic.HEUR/AGEN.1020069 [Aquarius]	11.10	2019-05-13_02	0.16
F-PROT Antivirus (Linux)	W32/A-a2959ab3!Eldorado (generic, not disinfectable)	4.6.5.141	201905131159	1.25
eScan Antivirus (Linux)	Gen:Variant.Ursu.393758(DB)	7.0.21	7.80776 (13/05/2019)	4.47
Comodo Antivirus (Linux)	TrojWare.Win32.Downloader.Agent.KAC	1.1.268025.1	2019-05-13	2.29
Clam AntiVirus Scanner (Linux)	❗	0.100.3	25448	0.01
Bitdefender Antivirus Scanner (Linux)	Gen:Variant.Ursu.393758	7.141118		5.94
AVG AntiVirus Free (Linux)	Dropper.Agent.BCTT	13.0.3114	4793/15883 (14 Aug 2018)	0.14
Emsisoft Commandline Scanner (Windows)		unavailable	unavailable	0.45

Figure 5.23: Detailed scan of one malicious file

The above figure shows a detailed scan of the file 21.exe from figure 5.22. It shows 7 of the available 10 antiviruses mark the file as malicious. This can be interpreted as the file is very likely to be malicious. We see one error from ClamAV that we see quite often. This has been discussed in the implementation chapter 4.1.7, and there are also two false positives from the Windows antiviruses. This is discussed in the next chapter 6.4.3.

5.4.3 Statistics

Table 5.13 and figure 5.24 show the aggregation of results from the above mentioned scan of 100 malicious files from theZoo[5]. The exact dataset used in the scan can be seen in appendix F.

[5] theZoo repository: https://github.com/ytisf/theZoo

Antiviruses	Detection	Errors	No detection
McAfee Command Line scanner (Linux)	34	0	66
AVG AntiVirus Free (Linux)	74	0	26
eScan Antivirus (Linux))	86	0	14
FSecure Antivirus (Linux)	77	0	23
Comodo Antivirus (Linux)	67	0	33
Clam AntiVirus Scanner (Linux)	2	98	0
Emsisoft Commandline Scanner (Windows)	32	2	66
McAfee Command Line scanner (Windows)	1	0	99
F-PROT Antivirus (Linux)	68	0	32
Bitdefender Antivirus Scanner (Linux)s (Linux)	83	2	15
All antiviruses	91	0	9

Table 5.13: Table showing detections, no detections and errors

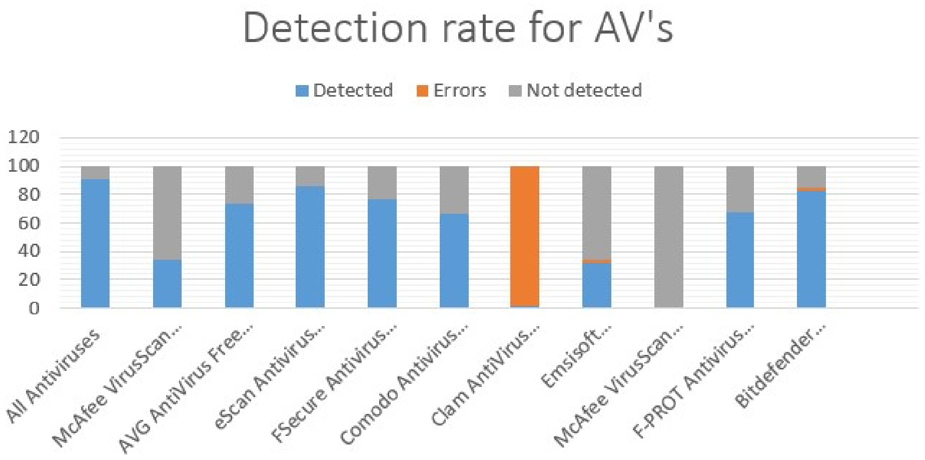


Figure 5.24: Antivirus - Detection rate

We see that in total 91 percent of the malicious files are discovered with some degree of varying discovery rate between the different antivirus products and a few products that has some problems. Even though we assume all 100 files to be malicious that is something we can not not know exactly. Some files may be bening files. Other files may be a delivery method for some malware that can only be discovered by a dynamic analyser.

5.5 What is shared?

Due to time limitations we could not do many thorough tests but we manage but we ran two Wireshark packet captures. One were with the host machine's network interface was turned off. We wanted to ensure that the analysers would still function properly and see what external addresses they tried to connect to. The other was run with complete internet access to check if any antiviruses connected to any external services and what information they possibly shared. We tested the same set of files both with and without internet access. The submitted files were a random collection of files from one of developers computer along with some malware.

We did a fairly simple check of the captured files. The host computers IP address is 10.0.0.15 and the public IP of the host machine was 88.95.173.107. To exclude this and other interal machines we used the filter in figure 5.25

```
!(ip.addr == 88.95.173.107) && !( ip.addr == 10.0.0.95) &&
 ↪  !(ip.addr == 10.0.0.125)
```

Figure 5.25: Wireshark filter

5.5.1 Without internet access

As expected every scanned completed without any problems and all the results were as expected. Most of the requests were DNS and API calls. The package captured and used for analysis is attached along with the report.

5.5.2 With internet access

Every scan obviously returned the expected results as well. A lot of the packets captured were NTP requests, DNS queris and interal ARP requests. There were also requests going to external IP's. These were of course encrypted and not possible to decrypt. Checking the IP's with an online IP lookup tool we can see who is responsible for each IP. Several of the IP's were American, one of the also belonged to Amazon (52.25.98.1). We cannot conclude anything but it is reasonable to assume that at least one of the outgoing connections were to an antivirus provider. This will of course need more research to get a correct conclusion, but with the mentioned requirements 2.1 of not sharing data it could be safe to assume that an air gapped or restricted internet access is beneficial. The package captured and used for analysis is attached along with the report.

6 Results and Discussion

This chapter sums up our general results compactly, and further discusses ideas and suggestions based off of our results. The discussions and arguments sections rely on what the overall results were by supplementing and expanding with what we, the developer team, deem logical courses of action for future work based off of our experiences and knowledge.

6.1 Results

Changes done, and therefore, the results are explained in depth in the section: 4.3.

6.1.1 Infrastructure setup

We have researched and modified the IRMA anti-malware framework, ultimately resulting in a thorough guide containing requirements, setup, configuration, pitfalls, and other supplementary information contained within this thesis (Chapter 4). Our guide uses our own slightly modified fork from the original quarkslab/irma, and is publicly available at krisshol/kmno-irma.

6.1.2 Infrastructure modifications

Antivirus troubleshooting

Not all antivirus analysers were working in the default IRMA installation. Multiple errors during provisioning were fixed and Ansible roles modified to accomodate outdated default configurations. In the end a total of 15 analysers were implemented and confirmed working, and 3 custom-made analysers were implemented; NSRL, Yara, and Cuckoo.

Pipeline support

Through a deep-dive into the scanning code base, we managed to add features supporting features to be required by a pipeline solution, albeit, we did not manage to implement a full pipeline.

These supporting features include; adding a new dynamic analysers category, functionality to retrieve probes and corresponding probe category, the method *isFinished()* for explicitly notifying when a scan is finished (Section: 4.3.1), and another function to combine individual results to take advantage of effective, smaller sub-scans and later piecing them together to the total scans. These features are implemented in the krisshol/kmno-irma fork, and will serve useful for a complete pipeline based on the discussions in Section 4.3.1.

6.1.3 Ported probes

NSRL

Originally, NSRL was a supported IRMA probe, but due to a weird abnormality with Python runtime versions, we found both implementation and the IRMA documentation to be outdated. The official implementation was written in soon-to-be deprecated Python2, but due to above mentioned abnormality the system was once run in Python3 with most functionality working, with the exception of this probe. This situation lead us to port the NSRL module to Python3, the implementation of which can be found at nikolaifa/NSRL. Unfortunately, it has not been appropriately tested in a running IRMA environment, but our modificated IRMA-fork will automatically provision this ported module using the nikolaifa/NSRL_ansible.

Cuckoo

We implemented Cuckoo using the existing ansible playbook at julianoborba/Ansible-Cuckoo as a starting point. During development to much time was spent on trying to implement a fully automatic VMware role, which in the end did not work due to feature restrictions within VMware Workstation. It also lead to us not completing an ESXi or vSphere role. We also did a complete implementation of Cuckoo as an IRMA probe.

6.1.4 Designed features

Kiosk

A kiosk was conceptually designed, and potential specific technologies available to solve the necessary use-cases were researched. The designed kiosk solution was based on GNU/Linux to access low-level functionality, but lacks any implemented solutions for handling user authentication or file transfers. The feature-limited kiosk prototype can be found at kmno-kiosk.

Pipeline

The pipeline conceptual design is complete, and specific technologies were researched for the use case. In the end, we advise using the Celery layout already in use by IRMA for enabling the required parallelisation. What lacks in the implementation is the stage/pipeline logic discussed in depth in section 4.3.1.

6.2 Hardware used in development and initial testing

HP ZBook 15 Mobile Workstation

- I7-4800MQ 2.7GHz base clock (3.7GHz boost)
- 16GB RAM
- 512GB SSD

It was stored vertically in a closed closet with minimal air circulation. The fans and air intake were facing out for optimal airflow in a non optimal environment.

6.3 Recommended Infrastructure

Because of our working environment (Section: 6.2), we have no practical basis for making the recommendations. Our only basis is some minimal guesstimates found in the official documentation for used technologies.

6.3.1 Orchestration computer/Ansible provisioner

Depending on the size of the infrastructure an average computer or server will do the job. If updates will be done manually a personal/work laptop can be used for a basic infrastructure provisioning, management and updating.

If the updates, management and provisioning should be automated, an Ansible AWX/-Tower instance might be beneficial. If the infrastructure is big, with more than 50 host and to be provisioned from a single computer, a high performance computer is recommended. Ansible uses SSH for provisioning and a high amount of concurrent connections might require a lot of resources.

6.3.2 IRMA

These recommendations are based upon general Quarkslab estimates, combined with Windows and VMWare recommendations. Quarkslab themselves specify that they do not have any specific numbers, as the system has not been tested in full scale usage. However, they give general hardware recommendations based on previous experiences and how they have hosted it in the past.

> "*For a large company, in theory, given a single high-memory machine, with 16+ cores, and SSDs, you could run IRMA platform and bear the workload load with reasonable response time.*" [1]

What exactly is meant by "large company" and "high-memory machine" is difficult to say, therefore, some estimations are made. How it is configured and how many probes are included is unknown in IRMAs estimations. The only relevant statement they make regarding hardware is:

> "*we managed to run the whole IRMA platform on a single machine by hosting it with multiple systems inside virtual machines: this setup gives fairly high throughput as long as it has reasonable IO (ideally, SSDs), and a good amount of memory (our setup was an i7 cpu with 16 GB ram on regular drives (at least 200 GB required)*" [2]

This is can be regarded as the minimum requirements for a usable system which includes every analyser. For a more efficient system the focus is increasing CPU cores and RAM. Storage space will vary depending on how often one wants to run a clean install, but using SSD storage is still important.

The system as a whole

The modules (front end, brain and probe) can either all be installed on the same machine or separately. If the different modules (front end, brain and probes) are installed on different machines the hardware resources have to be divided between them. How these are divided will depend on what is selected for each machine. It is recommended to separate the different modules so that they are easier to manage.

- CPU: 16 cores
- RAM: 32 GB
- Storage: 500+ GB

Front end

It must have resources to handle incoming requests and responses, sending/receiving tasks to/from brain, storage of files and scan results. Therefore the main focus for the front end is storage capability. As mentioned the size will vary on how often a clean install is done, but 300 GB should be enough to last more than a week depending on the

[1] IRMA recommendation: https://irma.readthedocs.io/en/latest/intro/requirements.html

[2] IRMA recommendation: https://irma.readthedocs.io/en/latest/intro/requirements.html

work load. As of now only one instance of the front end can run at a time [3] , and it is also only tested for Linux.

- CPU: 4 cores
- RAM: 4 GB
- Storage: 300 GB

Brain
It must handle requests/responses to/from the front end and the probes, store files and scan information temporarily. Therefore, it is more important for the brain to have multithreading capability. Required storage capability can be lessened through configuration for an external file server. Only one instance of the brain is supported at a time, and it is only tested for Linux.

- CPU: 6 cores
- RAM: 4 GB
- Storage: 100 GB

Probes
It must handle request/responses to/from brain and scanning tasks using present analysers. The number of CPU cores are most important as multithreading these scans are extremely important. The analysers, with the exception of Cuckoo, are not very demanding either.

- Linux (Debian) base
  - CPU: 2 cores
  - RAM: 4 GB
  - Storage: 10 GB
- Windows 10 base
  - CPU: 2 cores
  - RAM: 4 GB
  - Storage: 40 GB

The NSRL database files, which are about 4 GB, are calculated into the required base storage as it is an exception when it comes to storage required for each analyser.

Additionally one needs to add resources for each analyser installed on the machine. What the analysers actually requires varies slightly, but a general baseline makes the process easier. We recommend adding:

- CPU: 1 core for every 2 analysers
- RAM: 500 MB each analyser
- Storage: 1 GB each analyser

How one wants to split the analysers between the machines doesn't really matter as long as they have the required resources.

Lastly we recommend having the Cuckoo host running on its own probe machine as it stores files, scan results and logs locally.

[3] Multiple front ends: https://irma.readthedocs.io/en/latest/technical/brain.html#id1

6.3.3 Cuckoo

It should be noted that these recommendations are based on general Microsoft recommendations for Windows[4], VMware[5] recommendations for virtual machines and general common sense. The recommendations are based on clean Windows and Linux images with only the required programs to get Cuckoo running. It is not taken into consideration additional services and programs expected to exist in a realistic user or production environment. It is highly recommended to emulate a true to nature environment in your Cuckoo guest with anti-anti VM features to avoid more advanced malware slipping through your defences. These Cuckoo guests will most likely need more resources and that should be taken into consideration when acquiring/deciding hardware. A good starting point for estimating the necessary resources would be to check the official recommendation for the software you plan to integrate into your Cuckoo guest. You can estimate an expected load of software together with anti-anti VM features and use that as a basis to test and evaluate needed/optimal resources requirements.

The **CPU** should be of a "newer" architecture. It would be beneficial for both performance and power usage if the architecture is based on Sandy Bridge or newer for Intel processors, or the new Zen architecture for AMD processors. It is also highly recommended that the processors supports VT-d/AMD-Vi and VT-x/AMD-V. It's also likely that must be enabled in BIOS, supported by the virtualiser and per virtual machine for it to have any benefit.

Storage should preferably be SSD storage or another fast and low latency medium. A HDD storage solution with decent SSD caching would also be possible.

Recommendations per VM/Cuckoo guest

- CPU: 2 Cores (At least 1 per running VM)
- RAM: 2GB (4GB for Windows VM, 2GB for Linux)
- Storage: 60GB (anti-anti VM measures, simulating a realistic environment etc.)

Recommendations Cuckoo host

- CPU: 2 Cores + 2/1 (x concurrent VM's)
- RAM: 4GB + 2/4GB (x number of VMs)
- : Storage: 40GB + 60GB (x number of VMs)
- Support for VT-d/AMD-Vi and VT-x/AMD-V

6.3.4 Local mirroring

To improve the speed of provisioning, updating and potentially security, local hosting of required files and images should be evaluated. Some parts of the infrastructure requires downloading pre-configured images from online repositories, installer packages, virus definitions and more. These parts of the installation can potentially be sped up by hosting/caching the required files in-house. If an advanced threat is targeting the infrastructure they could abuse these external sources to plant malicious code or backdoors to get a foothold within the infrastructure. By verifying and hosting all the external files you as a host should in theory have complete control of every step of the chain. This

[4] Windows Minimum Requirements: https://support.microsoft.com/en-us/help/4028142/windows-windows-10-system-requirements

[5] vSphere Minimum Requirements: https://docs.vmware.com/en/VMware-vSphere/6.7/com.vmware.esxi.install.doc/GUID-DEB8086A-306B-4239-BF76-E354679202FC.html

can for example be solved by hosting necessary files on a in-house FTP server, VM images e.g in a datastore when using vSphere, Apt-Cacher [6] for caching required programs and installing Python packages from local packages [7]. This will of course require more infrastructure with some maintenance for the hosted files. It would also be required to edit and adapt the Ansible playbooks in varying degree, e.g for apt installed packages a simple edit of repositories are enough, while hosted files would need a configured and maintained SFTP server with proper adaption in the playbooks. This solution should fix a lot of the previously, mentioned errors in section 4.1.5, we got when provisioning IRMA. This could also solve potential future problems with downloading required files and packaged from external sources.

6.3.5 Kiosk

The kiosk is as of now only conceptual, but by definition it should not need much resources. The most resource demanding parts will be CPU and I/O, with limitations in the transfer protocol (USB, Thunderbolt, etc). By design, the kiosk should have limited input as each user needs to be authenticated before starting a scan. This means potential CPU congestion is a very unlikely, and bottlenecks will likely be in the actual transfers. Which means the most important components are I/O-ports and -protocols.

Based on this, a computer with USB3-ports, potentially thunderbolt to push for a move away from the insecure and flawed USB-protocol[8], and Gigabit Ethernet connection to handle the I/O should suffice. Depending on the actual implementation of the kiosk and how it handles files; threaded vs. single process, the CPU can either be as light as a dualcore processor or as heavy as a hyperthreaded quadcore processor.

6.4 Recommended further work

A recommended feature is a feature that will either greatly enhance the features or performance of the final product, or that will to a high degree be beneficial to evaluate and implement.

6.4.1 Additions

Recommendations for new probes

It is possible to add several more probes to IRMA, the documentation for this can be found here: [9] One possible new probe is Hashkeeper. It is very similar to the NSRL which has been written about earlier in the report here in section 4.3.4. According to the national Institute of Standards and Technology, the main difference is that the sources of NSRL are known while Hashkeeper's sources are not. The scope of Hashkeeper may potentially be much larger but the file data being used may also potentially be of illegal nature.[10]

There exists plenty of additional anti viruses that can be added on existing analyser

[6] Apt-Cacher: https://www.unix-ag.uni-kl.de/~bloch/acng/

[7] Python local packages: https://pip.pypa.io/en/stable/user_guide/#installing-from-local-packages

[8] USB flaw: https://www.schneier.com/blog/archives/2014/07/the_fundamental.html

[9] Adding probe: https://irma.readthedocs.io/en/latest/extending/add_probe.html#automatic-provisioning

[10] NIST NSRL FAQ: https://www.nist.gov/software-quality-group/about-nsrl/nsrl-frequently-asked-questions

hosts. Some examples include:

- Malwarebytes
- Tencent
- ViRobot
- With many more

Alternatives to Cuckoo, like FireEye AX and VxStream can also be considered. If one sandbox is already implemented we would not normally consider it necessary to add another, as the likelihood of a malicious file going undetected through both static and dynamic analysis is quite low, and resource requirements are substantial. If security requirements are high enough more sandbox solutions can be implemented, for example as a way to combat anti-VM techniques.

Implement kiosk

As stated multiple times, the kiosk solution was never implemented. In spite of this, specific technologies have been researched and a work-flow has been conceptualised. The specific use cases of the kiosk can be summed up to:

- Recommended Operating System:
  Any minimal GNU/Linux build. A safe and stable alternative is Debian Minimal.
- Low-level input handling:
  This is solved in our draft found at kmno-kiosk. It uses the pyudev-module, and proved through manual testing to be functional.
- High-level handling:
  Authenticating and authorising users to Active Directory. A Python3 module named easyad was found and unless future developers have experience with other LDAP solutions it looks like a recommended solution to use. It looks very intuitive, and with a low required level of entry, meaning worst case scenario is a miniscule amount of manhours spent on researching it.
- File transfer:
  Through our limited research, the only suitable tool for this use-case was rsync as it provides integrity checking and high bandwidth usage and can be recommended based on those two aspects. Unless the future developers have experience with any other secure and efficient methods to transfer files from one machine to another, looking further into using rsync should be beneficial.
- Connect scan and user identification:
  This step could be done in multiple ways. One is to adapt the receiving front end portion to accept the transfer of identifications and tag scans with their respective owner, either by modifying IRMA source code, or by placing a wrapper class on the front end to handle the kiosk output to handle repudiation of scans.
  One specific "quick and dirty" way of handling this is by making a file with a unique handle and adding the identifying user-tag on to it for the file transfer. The unique handle should in that case follow a format the front end knows to look for, and can then dynamically allocate the identifying user-tag to the respective scan as a tag.
- Kiosk front end:
  As suggested in Section 4.3.2, an ncurses-based command line user interface would be beneficial for guiding the flow of which a user interacts with the system. E.g. "In-

sert drive/disk/storage" -> "Login with acc@our.bizz.nis" -> "Iniate scan?". If yes, send files, and display progress on regular intervals. When scan finished, display result.

6.4.2 Modifications

Updating the system and air gapping

The system should periodically go through a clean install because the quality of the machines degrade over time. There is also a very small possibility of malware escaping from a sandbox VM[11] and spreading through the system. A clean install will limit the amount of time and potential for malware to do harm. Some reasons machines degrade over time are configuration drift (explained in section 3.3.3), storage space being filled up, etc.

Before a clean install is done, scan results, logs and known files can be extracted. As such, routines and documentation of what files should be extracted, how they should be extracted are necessary. Scan results are stored in the front end PostgreSQL database, which can be migrated using Alembic following the IRMA documentation.

If one wants to keep the files they are stored in the front end file system, sorted into folders by their hashes. Logs are stored differently depending on configuration, it is stored in /var/log for the file/folder belonging to the process, but can also be on a syslog server (mentioned in section 6.4.2). What exactly extracted is up to the owners of the system, but we recommend at least extracting logs and scan results.

As the system executes and deals with potentially harmful files it should be separated from the entirety of the internal KDA infrastructure. This can be dealt with mostly [12] by air gapping the infrastructure. An air gap is a network security method to isolate a secure network from an insecure one.

We recommend an infrastructure where the only server with any occasional internet access is the one hosting the local mirror. The local mirror can beneficially only be connected while updating. At any time the infrastructure is not explicitly updating, the mirror and infrastructure should be completely separated from each other. Ideally the mirror and infrastructure should be physically separated whenever updating the local mirror.

Updating the system can be done by setting up a new IRMA instance, testing if it works, and changing kiosk's end-point from the old instance to the new one. Tasks on the old instance will be completed before extracting necessary data and destroying the old machines. Orchestration and provisioning of the infrastructure should be done from a separate machine on the outside of IRMA's network, potentially using a local mirror, which is mentioned in section 6.3.4.

An important consideration to make when creating the update routine is how often the system should be wiped. For example, if its destroyed and recreated once a day, updating the antivirus databases and software in-between is pointless. If the wipe occurs once a week or rarer, it is needed to consider updating the antivirus databases and software in-between. That entails potentially daily antivirus databases updates, which means

[11] Escape from VM: https://en.wikipedia.org/wiki/Virtual_machine_escape

[12] Air Gap malware: https://arxiv.org/abs/1406.1213, https://en.wikipedia.org/wiki/Air-gap_malware

a connection must be temporarily opened so that the newer version can be acquired. The interface should not lead to the Internet but to a local mirror for the databases, and should be closed after usage.

This reduces potential exit points for an advanced malware that could potentially manage to spread across the malware analysis infrastructure.

Changes in pipeline

The pipeline development is not completed, and have features that can be added. The pipeline logic (section: 4.3.1) to decide whether a file should progress to the next stage or not should be tested and reworked for maximising efficiency while remaining the highest possible detection rate. This can be done by changing the code for the logic, testing it and iterating through different methods/logic for best possible results.

The structure of the stages (mentioned in section 4.3.1) should possibly be changed, at least when adding more analysers. E.g. if more dynamic analysers are added it would be a waste of resources to run all of these, they should rather be ran sequentially depending on necessity. If many antivirus analysers are added one should create a separate stage for these, so that resources are not wasted on scanning a file through e.g. 30 anti viruses, but instead 15 at first, then the others only when necessary.

Additionally, the files which are already scanned, and are stored on the front end and Cuckoo host can be deleted periodically. This would increase the lifespan of both machines as the storage is used up at a slower rate. Our advise would be to do this when a file scan is done, and after files deemed benign are sent to the Active Directory.

Logging

IRMA has three types of logs built into its configuration that exist separately for each module under /var/log. The logs are created independently by the processes running, the one(s) for IRMA is named syslog on each machine. IRMA have built in experimental functionality for exporting these logs to a logging server using rsyslog, (configuration is mentioned in section 4.1.3) which is an configuration default set to false. We haven't had time to test this functionality, but recommend testing it and expanding on it if necessary, and using it for formating and sending the logs to a selected logging server.

Scanning logs can be exported either through the frontend GUI where you can download a CSV report from the scan summary after the scan is completed. The report can also be retrieved with the API by sending a GET request to */results/resultId*. The result is returned in JSON format (Appendix H.38). There is also an option of doing this through irma-cli[13], but we have not researched the possibility.

Cuckoo

Cuckoo supports several modules, one of them are an Elasticsearch module that can export the log files to an Elasticsearch server. It is also possible to add AppArmor or SELinux support in Cuckoo for extended logging support. It is supported in the ansible playbook for Virtualbox, but we did not have enough time to verify its functionality within VMware and ESXi/vSphere so it was not included in the final product.

For Cuckoo a presentable report is available through the web GUI. Using the **REST API** you can do a GET request[14] to /tasks/report/ (int: id) / (str: format). To request

[13] irma-cli: https://irma.readthedocs.io/en/latest/use/cli/

[14] Cuckoo REST API documentation: https://cuckoo.sh/docs/usage/api.html?highlight=api#

your report replace the "int: id" with the id of the scan you wish to query in either HTML or JSON format by replacing "str: format" with JSON or HTML.

6.4.3 Fixes

Updating to Python3

IRMA only supports Python2 which causes a problem in the future as Python2 will be deprecated as early as 2020[15]. We hope that Quarkslab internally are working on updating IRMA to Python3, but we do not know for sure. We do think it is possible for people outside of Quarkslab to update IRMA to Python3 as the bachelor group wrote code for NSRL with Python3, but it could pose a challenge. The recommended course of action if this is deemed necessary, is to reach out to Quarkslab and collaborate efforts to update the IRMA framework as quickly as possible.

Cuckoo

Cuckoo currently only supports Python2 as of right now[16]. A lot of the external libraries supports both Python2 and 3 while some only support Python3 in newer releases. They have a long term goal to support the both Python2 and 3 but no current official timeline on release.

Fixing file upload bottleneck

As of now (section: 5.3) scans do not start at all before all files are present on front end and all tasks are defined for the scan. This means without enough resources to transfer files quickly, a few minutes are wasted at this stage. It should be possible to fix this using code added with pipeline to combine scans in the database, and through sending larger scans as sub-scans with a pre-made scan id added as a tag (section: 4.3.2). This will make the process somewhat more asynchronous and flexible to larger scan sizes with the exception of cases where the files themselves are large. This can also be fixed through a fast connection between the kiosk and the front end.

More thorough work on making scans a combination of asynchronous sub-scans for each file individually is a larger task which will increase efficiency, but the pay off compared to using a fast connection isn't very large so the work may not be worth it.

McAfee VSCL fixes

The antivirus McAfee VSCL scanner for windows almost always (99 percent of the time) return a result as benign, i.e it doesn't detect the files as malicious. We suspect this is because our version of the antivirus uses an outdated database. We use the virus database version 5600.1067. McAfee currently have two different versions of it's antivirus, v2 and v3, that is updated concurrently. The V2 version is currently on version 9257. Quite a leap from our current version on IRMA.

6.5 Nice-to-have improvements

A nice to have feature is a feature that is not critical for production value but can be a nice to addition if requirements change.

tasks-report

[15] Python2 #Update end-of-life: https://www.python.org/dev/peps/pep-0373/

[16] Cuckoo Python3 support: https://github.com/cuckoosandbox/cuckoo/issues/594

6.5.1 Additions

Threat levels

As with the pipeline, which automated some interpretation of scan results for deciding whether a file should continue to the next stage or not, some automated analysis of a whole scan result for a file is also advised. This could be done by combining viewing the values for whether a scan should continue to the next stage in the pipeline (section 4.3.1) and combining it with Cuckoo results. As it is hard to tell whether a file is potentially malicious or not we recommend basing it on numbers of detection by anti viruses, combined with a potential for harm based on file type and other metadata. Interpreting the results given by Cuckoo as a whole is probably extremely difficult, but Cuckoo handles it internally and can give us a potential which can be combined with the rest.

The representation of the levels can be as easy as 0-10 where 0 represents a non-existing to a low likelihood of it being malicious and 10 being extremely likely. With a system such as this one can automate the process for sending files to the Active Directory.

Distributed Cuckoo

If the infrastructure and load on Cuckoo increases "Distributed Cuckoo[17]" can be a beneficial advancement. This is only beneficial when there are two or more machines running a Cuckoo daemon and Cuckoo REST API. Distributed Cuckoo allows a common REST API point where samples and URLs can be submitted which will then be submitted to a connected Cuckoo node. This can allow a greater Cuckoo infrastructure without adding a lot of complexity and retaining a single point of access for you Cuckoo service.

Load balancing of probes

So far during the development process the bachelor group used two probes with a goal of using a third. Because of the development infrastructure (described in section 6.2 we used, we did not have the opportunity to use any more probes as we already experienced some problems with the current infrastructure. We recommend that KDA uses more resources and creates an infrastructure where they will have the opportunity to add more probes. Thus, the total number of analysers could be split among several probes. If IRMA can handle more than one of the same probe we speculate that it can be beneficial to run several of the same probe on different hosts, if we at the same time can load balance them.

6.5.2 Modifications

Front end GUI Improvements

As it is today, the frontend GUI is exceedingly lacking in accessibility and a positive user experience (UX). For instance, to access the results of one full, earlier scan, a user would have to; navigate to the search tab; then either know a specific file name or it's SHA256 value and search for it; or change the number of displayed files per page to circumvent a bug where the list of files will not load initially, after which the user must know a file pertaining to the desired scan to select it from an alphabetically sorted list of files; this sends the user to the report page of that specific file, and finally here the option to select "Back to the scan summary" to select a scan summary is available.

That is one of two ways to retroactively access a full scan summary. The other is to

[17] Distributed Cuckoo: https://cuckoo.readthedocs.io/en/latest/usage/dist/

already know the identifying SHA256-hash of that scan and input it directly in the URL. In other words, if the GUI is to be used at any stage it needs a big overhaul in design philosophy. This can be done as simply as basing it on the existing IRMA code and adding options to sort results by time of scan or what scan individual files pertain to. It would likely be advantageous to introduce a whole new tab for viewing and selecting past scan summaries directly from an overview of all existing scans as well. One way to accomplish sorting, could be by using the already implemented tag feature (Section 4.3.2)

Front end performance and reliability

If the platform is under heavy use and needs more input capacity to make use of abundant backend resources, adding more front ends behind a load balancer can be a good option. A load balancer approach will also provide more stability and redundancy if one or more of the front ends were to go down or fail.

As of now, IRMA has no support for multiple front end servers, but in theory it should be fairly simple to configure multiple frontends with a single brain. Support for multiple frontends is a feature that the developers want to implement, but no ETA or official plan is public.

It should be fairly simple to implement by creating new instances with the same configuration as provisioned by the playbook. Some changes to the configuration is probably needed to support multiple frontends, at least in on the brain celery configuration.

The internal SQL server on the front end can be provisioned to a separate host from the rest of the front end. This would provide an option to keep a permanent SQL record of all scans, even over multiple clean installs. If some adaptions are made, it should be possible to create a database cluster for increased performance and redundancy, and depending on the configuration load, balancing between databases should also be possible.

To improve the speed and reduce resource usage on the different frontends, a caching service could be configured and set up along with the load balancer. Such a cache should decrease load time for the end user and reduce the load on the front end. That is, as long as the cache storage is big enough to cover all of the most requested files and pages.

Ansible Playbook modularisation

Setup and maintenance of IRMA takes quite a long time, sometimes longer than necessary. This is mostly because there are a lot of components that needs to be provisioned and handled correctly. In most cases more tasks than necessary will be executed during a play because the playbooks are big and complex. A potential way of reducing the time used would be to modularise the playbooks by extracting parts of the playbook and put them in their own playbook. This can allow a user to run segments of the complete playbook instead of the full playbook on every update or configuration change. IRMA has already implemented this to a degree by separating the playbook into three pieces; provisioning.yml, updating.yml and deployment.yml. Our suggestion is a further segmentation of the playbooks into smaller parts so it won't be necessary to run tasks that are irrelevant for the given change or update.

The same could also be done with the Cuckoo playbook. The Cuckoo playbook is not nearly as complex but it could also gain on being segmented a bit. For example if testing of changes or updating should be done on only one of the roles it could save time to run one of the playbooks instead of running a full play.

Improving provisioning through snapshots

As of now provisioning of the system is done in multiple phases. First configuring and building a box using Packer, then using this box to orchestrate machines and lastly installing the system on these machines. This setup is prone to failure during the provisioning phase, note that we've experienced numerous special case errors due to the working environment VMs being so limited (working environment described in 6.2). After the system is properly configured for its environment we recommend moving as much of the provisioning into Packer or a similar tool for building snapshots/images/boxes. This allows for extensive testing of capabilities before production and moves provisioning to the testing, therefore also removing provisioning errors/faults from production, this depends on how extensive tests are. With good coverage in tests and good tests in general one can then be sure that the snapshot will work when it is started instead of having to wait about (section 5.2) 15 minutes on a potentially working build. This leads to a more reliable and scalable infrastructure which are constant goals of Infrastructure as Code.

The current Ansible implementation probably doesn't support this entirely as it also starts necessary processes on each machine after installation. This is fine as it allows for testing before the build is completed, the issue is that probably not all of these services automatically starts upon reboot. There are several methods to fix this issue, for example by adding these as services which are enabled to start on boot by systemd, or by adding local provisioning as a service run during boot which starts them.

Builds should be smoke tested individually to test whether a build of a specific module (front end, brain or probe) is working as expected. These must be written for each machine individually as they perform separate tasks. There are already functionality present for testing whether expected probes are present and works as they should (command in section 4.6).

After individual smoke testing, the system should be tested for functionality collectively by starting a machine for each image. The most important tests for this stage is whether basic functionality works. This includes web page, scanning, retrieval of results etc. Cuckoo and other dynamic analysers eventually added requires testing by themselves, which can be done through their APIs, before being added to collective tests.

The testing environment wont need quite as much resources as the production environment because speed and capacity no longer is that important.

The snapshot creation should be split up into several stages where each goes through testing and are based upon the previous stage. This modularises the creation and skips unnecessary building which in turn increases speed.

Cuckoo network configuration

With the playbook used in this project Cuckoo is configured with no internet access at all. Cuckoo has several ways of handling networking [18]

INetSim

Is a network simulation tool:

> *INetSim is a software suite for simulating common internet services in a lab envi-*

[18] Cuckoo Routing: https://cuckoo.sh/docs/installation/host/routing.html?highlight=inetsim#routing-tor:

> *ronment, e.g. for analyzing the network behaviour of unknown malware samples.* [19]

INetSim can be used for creating a better emulation of external and internal infrastructure. This can be relevant as an anti-anti-VM measure for advanced malware, or for emulating your real infrastructure for advanced custom created malware for your business.

VPN & Tor

If you don't care, or want the malware to get internet access, and want to obfuscate your real IP and location. There are two supported ways of doing this in Cuckoo. You can send the traffic through Tor [20] or through a VPN. Both are modules must be configured and enabled in Cuckoos respective configuration files, see link mentioned in the intro of this section (Cuckoo network access and handling.

[19] INetSim: https://www.inetsim.org/

[20] The Tor Project: https://www.torproject.org/about/history/

7 Conclusion

This chapter gives a short overview of the project, what is done, what we have learned, and what should be done in the future.

7.1 Project assessment

From the start this project was quite broad without very strict and confined requirements. KDA wanted us to explore and research alternatives for analysis of digital content. From one perspective this gave us a lot of flexibility on how to implement the bachelor project. Quite a lot of time was spent initially on researching different technologies, evaluating the technologies, choosing an alternative and learning about the chosen alternatives.

After the initial technologies were chosen, the most important being IRMA and Cuckoo, quite a lot of time were used figuring out how to install and configure those systems/infrastructures. For this, and the aforementioned reason, we could not implement as much of the infrastructure as we would like. Most of the pre-planned functionality was implemented, but not all of it was tested properly and some new ideas we got was only logically designed and planned.

In retrospect, when writing the report, we should have been better at documenting errors and potential bug fixes we did. During the development, individual group members would work on a problem but not always share the problem and/or the fix. As the project was so broad, different group members would work for some time on many different problems and it was easy to lose track of what was being done.

From the beginning we had planned to post daily scrum-reviews of what was being done in the bachelor groups messenger chat but it was quickly lost among general discussions and administrative decisions. In the beginning of march we separated daily scrum reviews into it's own slack channel were only scrum-reviews were posted. This increased our overview of what was being done somewhat, but we could have been more clear with what was written. We could also have tried to meet up and worked together more, but this was difficult as 3 people had different subjects at different times and one student having a part time job.

7.2 Knowledge gained

As the project was so open and broad in scope we have gained quite a lot of new knowledge surrounding different infrastructure technologies. We have become better at configuring systems/infrastructures and troubleshooting, in addition to gaining skills in project planning/management.

7.2.1 Configuration with Ansible

The configuration language Ansible was used extensively during the project, and all group members have gotten experience using it by the end of the project. Most if not all of the group members have used, edited and made either their own or other playbooks and Ansible roles.

7.2.2 Orchestration with Vagrant

Vagrant is the orchestration tool used by the group. The group has gained knowledge on how to configure Vagrantfiles to get a development and/or production environment. The group also learned the necessary commands and knowledge to configure and troubleshoot the environment.

7.2.3 Troubleshooting

The group has gotten extensive experience with troubleshooting systems/infrastructures we were introduced to. This could be simple syntax errors related to Ansible and Vagrant but it could also be more complex logical errors found during configuration of IRMA and Cuckoo. Errors during IRMA provisioning were often especially tedious as it often would take five minutes or more to check if an error were fixed.

7.2.4 Project management

At the start of the project we had a project planning phase where we chose scope, development model and made a plan for implementation among other things. During the development project we used several tools to help us with our project management. This has given us skills we can take to further projects in the future.

7.2.5 Malware detection using multiscanning

We have learned much about how an infrastructure for handling multiscanning functions, using multiple scanners, of files during the development of IRMA. This have given us a good insight into challenges that occur when working on large scale tasks such as an infrastructure.

7.3 Results

- IRMA installed, configured and modified
- Cuckoo installed and configured
- 15 antivirus analysers confirmed working
- 3 custom made analysers added
    - Cuckoo
    - NSRL
    - Yara
- Pipeline logic designed
- Kiosk designed

7.4 Future Work

- Kiosk implemented
- Update to Python3
- Pipeline implemented

Bibliography

[1] Morris, K. 2016. *Infrastructure as Code - Managing Servers in the Cloud*. O'Reilly Media.

[2] Hosmer, B. 2015. *Getting Started with SaltStack*. https://leanpub.com/gettingstartedwithsaltstack.

A Installation Guide

A.1 IRMA

Clone the repositories:

```
git clone --recursive https://github.com/krisshol/kmno-irma
```

The official version is at: https://github.com/quarkslab/irma

Install the requirements, exact versions and specification can be installed through:

```
pip install -r requirements.txt
```

The requirements file is located in the ansible folder.
For using Vagrant in combination with VMWare one needs a plugin, this needs to be licensed. The license file can be acquired in contact with Hashicorp at https://www.vagrantup.com/vmware/index.html. The plugin can then be installed and licensed using:

```
vagrant plugin install vagrant-vmware-desktop
vagrant plugin license vagrant-vmware-desktop ~/license.lic
```

After the plugin is activated, and the hypervisor installed one can setup the machines with the wanted configuration using (from the ansible folder):

```
export VM_ENV="name of environment file" # optional
vagrant up
```

The environment files are located in the ansible/environments folder. To specify a provider (e.g. VMWare) for Vagrant to use the flag –provision can be added.
The machines can then be provisioned using from the ansible folder using:

```
python irma-ansible.py environments/prod.yml playbooks/playbooks.yml
```

This will start an about 20 minutes long provisioning of necessary files and software for the system.
After the web GUI should be available at:

```
http://<the IP specified in the environment file for front end>
```

And the API documentation will be available under:

```
http://<the IP specified in the environment file for front end>/swagger/
```

Which probes that are added is specified in the environment file selected. And available probes are listed in provisioning.yml.

A.2 Cuckoo

Requirements/Preparations

Ansible Provisioner

On the Ansible provisioner, ansible and the git repository with the playbook are obvious requirements. It is also necessary to have all the correct information regarding the systems Cuckoo will be installed on, and potential guest systems as well as root access.

Installation

Step 1 - Clone the Ansible playbook repository

```
git clone https://github.com/knaku/Ansible-Cuckoo.git \\
```

To install Ansible, run the install_ansible.sh in the root folder of the newly cloned git repository, or copy and run the commands below.

```
apt update
apt -y install apt-transport-https software-properties-common
apt-add-repository ppa:ansible/ansible -y
apt update
apt -y install ansible sshpass
```

If you don't already have git installed run:

```
apt update
apt -y install apt-transport-https software-properties-common
apt install git
```

Step 2 - Prepare Cuckoo host

Ubuntu 18.04 server/machine, 16.04 and 17.04 should work as well but have not been tested.

Ansible requires Python and a SSH-server to run. If you use a regular desktop Ubuntu there is a good chance that both are already installed.

Run these command in you terminal to install the needed requirements:

```
apt install -y openssh-server python
```

If you want Ansible to connect with the user name and password of the account, use those. If you want Ansible to connect with SSH-keys, simply add your public key to the Cuckoo host.

Step 3 - Cuckoo guest image

You need a pre-configured Windows, Linux or macOS fulfilling the requirements given by Cuckoo [1]. You can see a quick summary of Windows requirements below:

[1] Cuckoo software requirements: https://cuckoo.sh/docs/installation/guest/requirements.html

- (Optional) All anti-anti VM counter measure configured and installed
- (Optional) All user programs (Microsoft Office, Adobe Acrobat/Reader etc.), and other requirements to emulate a normal environment.
- Installed python.
- agent.py that starts automatically, preferably with administrative rights. Use .pyw instead of .py if you want the terminal window to be minimised.
- Turn off the Windows firewall.
- Turn off Windows updating.
- Set a static IP. Cuckoo cannot handle DHCP yet. If you leave everything as default after cloning set it to 192.168.56.111 with a default gateway of 192.168.56.1.
- Snapshots:
-
    - You can create a snapshot manually while the VM is running with requirements above
    - You can let Ansible handle it automatically (mostly if using VMware Workstation). Agent.py is required to start on startup for this to work.
    -
        - With Virtualbox you can let the playbook handle everything automatically.
        - With VMWare Workstation you must manually start and stop the VM during the play if a snapshot does not already exists. If a snapshot exists it might create a bit of issue with Cuckoo and VMWare.

Step 4 - Installing and running Cuckoo

Replace the information in ***cuckoo-playbook/inventories/production/hosts*** with the correct one for your Cuckoo guest.

- HOST is the IP address of the server to install Cuckoo to
- ADMIN is a user with sudo privileges on the server
- PASSWORD is the user ADMIN password

Now run this command in the terminal of your ansible-provisioner to start the play, look below for explanation for a description of the different variables after –extra-vars:

```
ansible-playbook -i inventories/production site.yml --extra-vars "distribution=bionic nic=ens32 vmwareNetworkAdapter=1 license=xxxxx-xxxxx-xxxxx-xxxxx-xxxxx"
```

- "distribution" is the Ubuntu distribution
- "nic" is the nic on the cuckoo host to use in the routing.conf
- "vmwareNetworkAdapter" is vmnet suffix used with the cuckoo guest.
- "license" is the license for VMWare Workstation and is only to add when using VMWare workstation.

You might want to look over and verify or change the different variables in some of the configuration files since they are hard coded and might not reflect the values you have in you environment. Directories and files to check are "cuckoo-playbook/roles/[name of role]/files/*.conf" The most relevant roles to check are Cuckoo and VMWare, the others might be relevant depending on your problem but have not been changed by us. Look at the bottom of the git repository [2], under "Things that need manual changing if you change the default" for most relevant variables that might need changing.

[2] Ansible Playbook for Cuckoo: https://github.com/knaku/ansible-cuckoo

B Project Agreement

Prosjektavtale

mellom NTNU Fakultet for informasjonsteknologi og elektroteknikk (IE) på Gjøvik (utdanningsinstitusjon), og

Kongsberg Defence & Aerospace (oppdragsgiver), og

Kristian Sigtbakken Holm, Nikolai Fauskrud, Martin Kvalvåg og Olav Henrik Hoggen (student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra **01.01.19** til **20.05.19**.

Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU IE på Gjøvik yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra NTNU å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon/fax, reiser og nødvendig overnatting på steder langt fra NTNU på Gjøvik. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. NTNU IE på Gjøvik står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Alle bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv hvis de har skriftlig karakter A, B eller C.

Tilgjengeliggjøring i det åpne arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og instituttleder/fagenhetsleder om de i løpet av prosjektet endrer syn på slik offentliggjøring.

Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av NTNU til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved NTNU og/eller studenter har interesser.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i NTNUs elektroniske eksamenssystem. I tillegg leveres ett eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, IE er det instituttleder/faggruppeleder som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og NTNU som regulerer nærmere forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene. Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale med oppdragsgiver, skjer dette uten NTNU som partner.

10. Når NTNU også opptrer som oppdragsgiver, trer NTNU inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene imellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): **Basel Katt**

Oppdragsgivers kontaktperson (navn): **David Lee Andersen**

Student(er) (signatur): Olav H. Hoegegen dato 25/1/2019
Nikolai Fauskrud dato 25/01/19
Kristian S. [illegible] dato 25/01/19
Martin Kvalvåg dato 25/01/19

Oppdragsgiver (signatur): Thomas R. Andersen dato 13/2-19

Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.
Godkjennes digitalt av instituttleder/faggruppeleder.

Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.
Plass for evt sign:

Instituttleder/faggruppeleder (signatur): ______________________ dato ________

C Development Process

C.1 Daily Scrum logs

D IRMA dataflow models

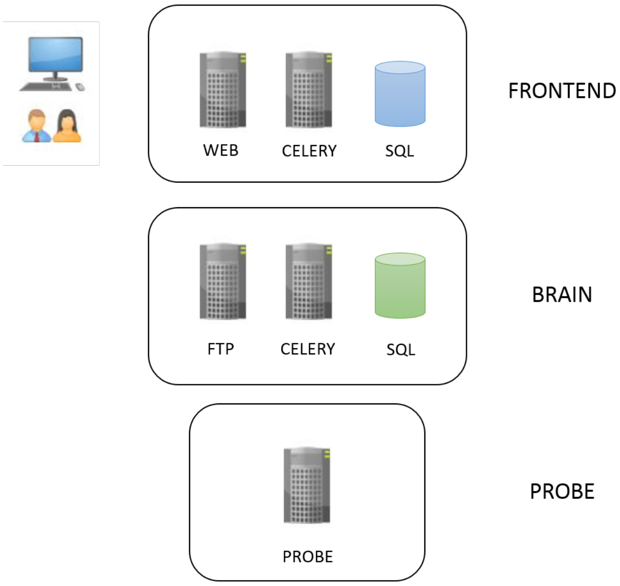


Figure D.26: IRMA Figure 1

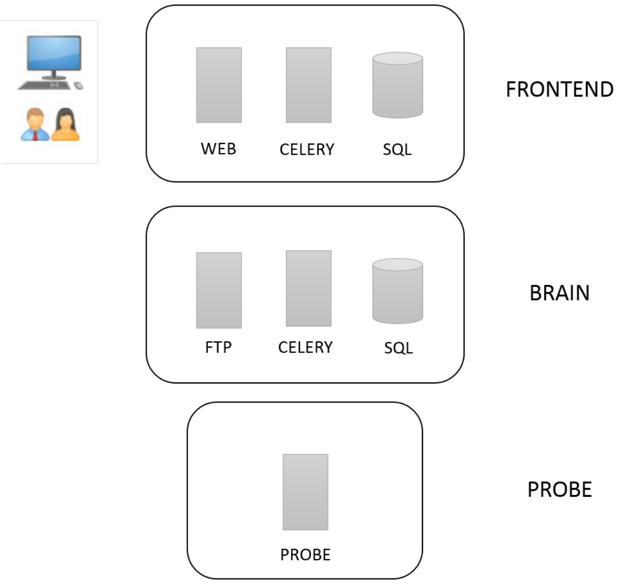


Figure D.27: IRMA Figure 2

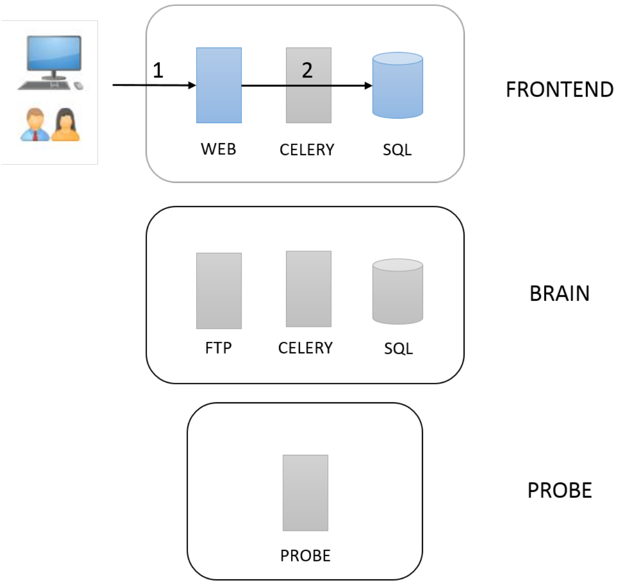


Figure D.28: IRMA Figure 3

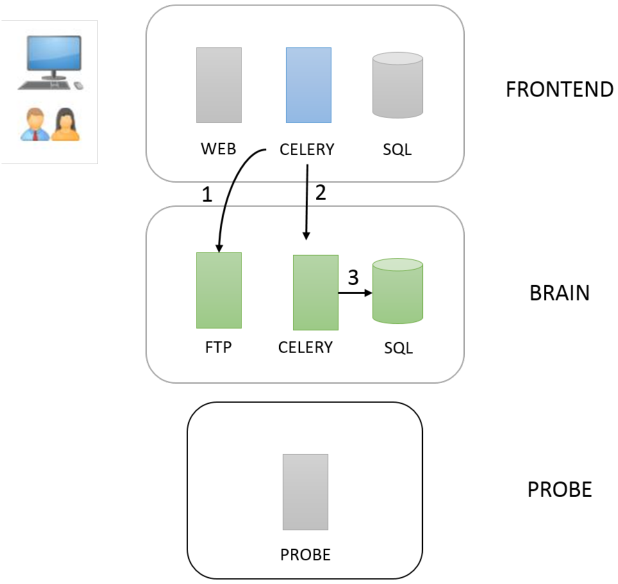


Figure D.29: IRMA Figure 4

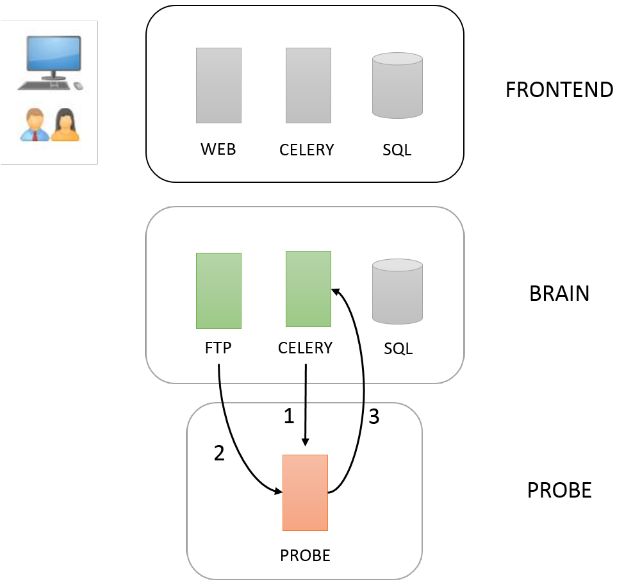


Figure D.30: IRMA Figure 5

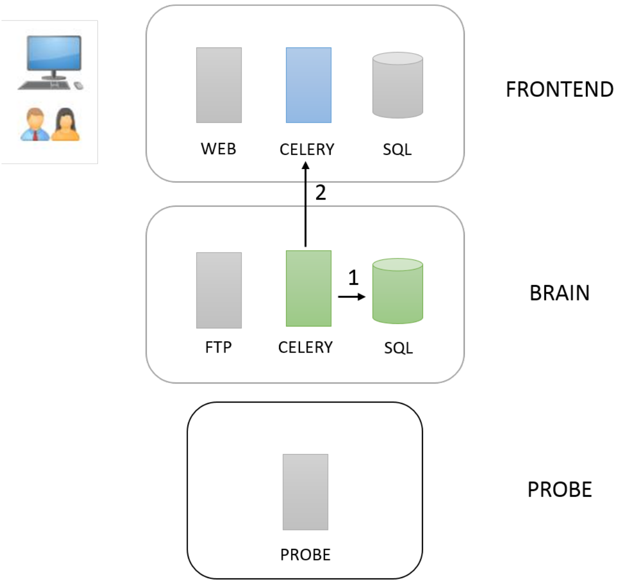


Figure D.31: IRMA Figure 6

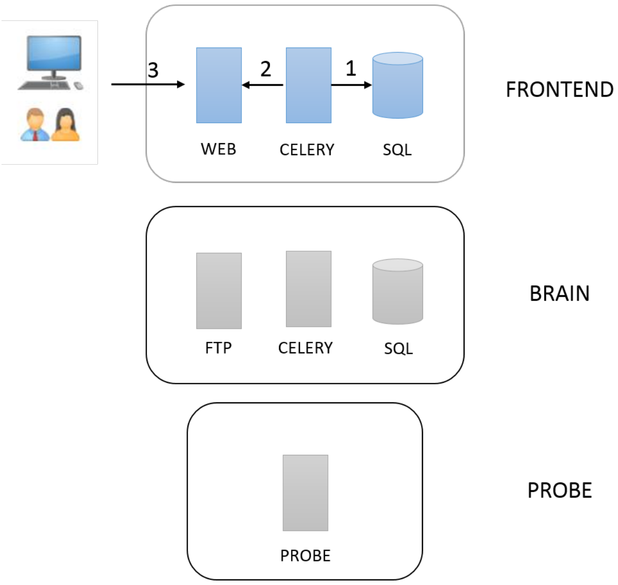


Figure D.32: IRMA Figure 7

E Kiosk appendix

E.1 Kiosk mount helper prototypes

```
"""Mount drive helpers:"""
import subprocess

def temp_mkdir(target):
"""Dynamically make temp dir '/tmp/<target>'."""
    command = "mkdir -p /tmp/{}".format(target)
    print(command.split(" "))
    output = subprocess.check_output(command.split(" "))

def temp_rmdir(target)
"""Remove temp dir <target>."""
    command = "rmdir -p /tmp/{}".format(target)
    output = subprocess.check_output(command.split(" "))

def readonly_mount(source, fs):
"""Mount <source> as read only at /tmp/<source>
with file system <fs>."""
    temp_mkdir(source)
    target = "/tmp/{}".format(source)
    command = "mount -o ro,noload -t {2} {0} {1}".format(source,
     ↪  target, fs)
    output = subprocess.check_output(command.split(" "))

def unmount(target):
"""Unmount a block from <target>"""
    command = "umount {}".format(target)
    output = subprocess.check_output(command.split(" "))
    temp_rmdir(target)
```

Figure E.33: Kiosk mount helpers protoype

E.1.1 Simplified kiosk work-flow

- Detect new drive added.
- Ask for user authentication.
- Authenticate with AD-structure.
- If authorised:
    - Fork process/start new thread.
    - Mount drive.
    - Rsync all folders and files directly to the front end.
    - The rest of the platform handles analysis.

- Else:
    - Deny input request.

F Testing Appendix

The dataset used in the accuracy of scans section can be seen under.

Linux time			Linux time			
Date	SHA256Sum	Filename	First seen	Last seen	Size	Status
1557755177	ddf2542dc5ac74	7ZipSetup.exe	1557581705	1557755384	162032	1
1557755177	e9cfb6eb3a77cd	5a765351046fea1490d20f25.exe	1557581704	1557755385	386048	1
1557755177	036e4f452041f9	3_4.exe	1557581704	1557755395	60928	1
1557755177	ef32516eb5658	2.dll	1557581591	1557755408	41988096	1
1557755177	3144079c68ba0	901FA02FFD43DE5B2D7C8C6B8C2F6A43_SideBar.dll_	1557752795	1557755394	41984	1
1557755177	2fd5b075ab9dff	798_abroad.exe	1557581707	1557755426	1638624	1
1557755177	2ecc525177ed52	131.exe	1557581706	1557755421	2415104	1
1557755177	a13161d957ef1b	97C11E7D6B1926CD4BE13804B36239AC_SideBar.dll.doc_	1557752793	1557755427	124732	1
1557755177	027cc450ef5f8c5	027cc450ef5f8c5f653329641ec1fed9.exe	1557237901	1557755429	362360	1
1557755177	8abb47ca7c0c48	21.exe	1557581705	1557755429	56224	1
1557755939	e226dc651390b	_setup.lib	1557582038	1557756068	247905	0
1557755939	961537d5fd688	counter.exe	1557581711	1557755961	32256	1
1557755939	5291232b297df	1002.exe	1557581707	1557755978	257024	1
1557755939	89a1bbe42cde0	ch.dll	1557581710	1557756051	13312	1
1557755939	e67834d1e8b38	cerber.exe	1557581710	1557756034	619008	1
1557755939	b02c56d294476	cam.dll	1557581710	1557756051	65024	1
1557755939	84d06e7541baf	Build.exe	1557581710	1557756061	401408	1
1557755939	96ca097b0daff9	BOTBINARY.EXE	1557581710	1557756062	77824	1
1557755939	ef1503f018dc86	Acrobat PDF Writer 3.exe	?	1557581709	1557756066	1
1557755939	3a93d0b434590	abba_-_happy_new_year_zaycev_net.exe	1557581708	1557756069	194968	1
1557755939	8cf50ae247445c	1003.exe	1557581707	1557756068	261120	1
1557757059	89c2d370bfa36f	dumped2.exe	1557581746	1557757135	1965568	1
1557757059	17b7ad3434a9c	dumped.dll	1557581745	1557757117	73728	1
1557757059	db8c0fc8427546	dump1.exe	1557581745	1557757158	274432	1
1557757059	32f66e8f05d39b	DELLXT.dll	1557581745	1557757212	21495808	1
1557757059	a15c351b94004	decrypted_inj_services_x64.dll	1557581712	1557757197	28665	1
1557757059	099ad10b55e74	decrypted_inj_services_Win32.dll	1557581711	1557757211	61440	1
1557757059	b71a4a57d2174	Fake Intel (1).exe	1557581749	1557757215	1220213	1
1557757059	4122054927442	F897A65B.exe	1557581748	1557757219	79620	1
1557757059	ed01ebfbc9eb5b	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5	1557576585	1557757219	3514368	1
1557757059	ae79d6e52e9eb	DW20.dll	1557581746	1557757219	44032	1

1557757298	d23b4a30f6b1f0	gchrome.exe	1557581751	1557757463	2930176	1
1557757298	e98dccc92e1733	f-mydoom.exe	1557581750	1557757432	113664	1
1557757298	f132324acf09c05	fm.dll	1557581750	1557757475	13312	1
1557757298	fc3e0bee121475	FLASH829.EXE	1557576586	1557757526	22528	1
1557757298	3f5a6d8334f31a	FIX_NIMDA.exe	1557581750	1557757527	40960	0
1557757298	4265111455953	file_4571518150a8181b403df4ae7ad54ce8b16ded0c.exe	1557581750	1557757527	873472	1
1557757298	10745182ac1b73	fax_390392029_072514.exe	1557581749	1557757527	283136	1
1557757298	ef32516eb5658c	hV46VA.dll	1557581591	1557757532	41988096	1
1557757298	6f201afc797370	hostr.exe	1557581752	1557757528	107520	1
1557757298	87aaec18d56d29	GREEN.EXE	1557581752	1557757533	77767	1
1557757798	c1f6f5c2cb8ee4c	malware3.exe	1557753384	1557757829	158464	1
1557757798	4265111455953	malware2.exe	1557581750	1557757877	873472	1
1557757798	89c2d370bfa36f	malware1.exe	1557581746	1557757883	1965568	1
1557757798	3373baa1681c89	malware.exe	1557581869	1557757882	269312	1
1557757798	b10eeea84d9bf0	loader_00400000.Embedded01.DLL	1557581869	1557757917	32768	1
1557757798	59979d3bc3d645	jpeg1x32.dll_C2BA81C0DE01038A54703DE26B18E9EE	1557581868	1557757895	31744	1
1557757798	834d1dbfab8330	InstallBC201401.exe	1557578095	1557757917	13370880	0
1557757798	c22b5d38c0de8f	malware6.exe	1557753385	1557757915	428168	1
1557757798	efc94fdac875345	malware5.exe	1557753384	1557757922	20480	1
1557757798	9781784910935	malware4.exe	1557753384	1557757928	98158	1
1557758135	fd624aa2055175	njRAT.exe	1557581870	1557758206	982016	1
1557758135	cd9709bf1c7396	NAudio.dll	1557581870	1557758160	391168	0
1557758135	cdadc26c09f869	Mono.Cecil.dll	1557581870	1557758225	312320	1
1557758135	c8d699c35d3070	MiniConfigBuilder.exe	1557581869	1557758222	13312	1
1557758135	fde583027a692c	Mic.dll	1557581869	1557758242	417280	1
1557758135	0f79e9a13bd268	Q30097~1.EXE	1557581871	1557758241	237280	0
1557758135	a3e4bee1b6944a	pw.dll	1557581871	1557758245	39936	1
1557758135	66bca3f92841b7	PlugX_3C74A85C2CF883BD9D4B9F8B9746030F_DW20.dll_	1557753387	1557758246	233472	1
1557758135	40050153dceec2	PDFXCview.exe	1557581871	1557758253	431884	1
1557758135	e3e057465bb3a	payload.dll	1557581871	1557758251	10752	1
1557758283	5e77eee9704e6	scanslam.exe	1557581873	1557758307	61440	0
1557758283	d3dcb25f9004f6	sc2.dll	1557581873	1557758346	10752	1
1557758283	ebc324308ee010	sample.exe	1557581873	1557758362	57344	1

1557758283	517ac5506a5488	Ransomware.Unnamed_0.exe	1557581873	1557758355	924160	1
1557758283	d24d79011d003	raffle.exe	1557581872	1557758371	1787208	1
1557758283	aedf60c10f8cc74	Q300972I.EXE	1557581871	1557758378	242832	0
1557758283	8e2bdcaee8dfef	slide.exe	1557581874	1557758391	102400	0
1557758283	4c2efe2f1253b9	signed.exe	1557581874	1557758394	91888	1
1557758283	39aedd6cd6df61	shmgr.dll_AD6590E0DF575228911852B1E401D46E	1557581874	1557758400	161280	1
1557758283	82379c6a61d040	SCHDPL32.exe	1557581873	1557758401	166400	1
1557758461	dce2d575bef073	Win32.DarkTequila.exe	1557237828	1557758488	877568	1
1557758461	699ec052ecc898	Win32.AgentTesla.exe	1557581877	1557758502	460800	1
1557758461	9dfde6e3e10615	W32_Swen@MM.exe	1557581876	1557758526	106496	1
1557758461	0fa128bd5e54cc	W32.Elkern.4926.exe	1557581876	1557758547	397312	1
1557758461	c7dc529d8aae76	Vcffipzmnipbxzdl.exe	1557581875	1557758563	846848	1
1557758461	b831f61d4e2a43	UpdateCheck.exe	1557581875	1557758551	8192	1
1557758461	58e61318aad78	TOKYO_1258.EXE	1557581875	1557758571	736	1
1557758461	e5c643f1d8ecc0	svchost.exe	1557581875	1557758568	720896	1
1557758461	fff0ccf5feaf5d46	strip-girl-2.0bdcom_patches.exe	1557581874	1557758577	22528	1
1557758461	9d88425e266b3	win32.exe	1557581466	1557758582	24960	1
1557758649	2c7b1c5c51f695	wirelesskeyview.exe	1557581469	1557758794	177568	1
1557758649	4eabb1adc035f0	win33.exe	1557581469	1557758803	68096	1
1557758649	6528633916983	Win32_klez.exe	1557581469	1557758860	253952	0
1557758649	653bc2b16b1624	Win32.WannaPeace.exe	1557581469	1557758867	674816	1
1557758649	55504677f82981	Win32.Wannacry.exe	1557237910	1557758879	5267459	1
1557758649	ff808d0a12676b	Win32.SofacyCarberp.exe	1557581466	1557758875	133632	1
1557758649	1c0ea462f0bbd7	Win32.GravityRAT.exe	1557581466	1557758881	660480	1
1557758649	ef32516eb5658c	ydrHrp_One.dll	1557581591	1557758887	41988096	1
1557758649	bc73990f2d0435	YAUNCH.EXE	1557581469	1557758886	8681	1
1557758649	a6ff8dfe654da70	wmighost.dll	1557581469	1557758886	20480	1
1557759030	cee4a7208ec314	ZHR2970.EXE	1557581592	1557759076	5290	1
1557759030	8463ac9eec3013	ZHR1958.EXE	1557581592	1557759052	4534	1
1557759030	71b38f041b4a4a	ZeroAccess_xxx-porn-movie.avi.exe_	1557581592	1557759131	163840	1
1557759030	0e4cdcde772cf0	Y-TP46.EXE	1557581592	1557759077	29825	1
1557759030	d080159cff102b	Y-L3052.EXE	1557581592	1557759139	4092	1
1557759030	c2581af6d4ff858	yfoye_dump.exe	1557581592	1557759131	36864	1

1557759030	91fa185f353b79	YESMILE.EXE	1557581592	1557759148	4946	1
1557759030	e57367f1a537b8	YEKE1204.EXE	1557581592	1557759149	21684	1
1557759030	a15c351b94004	decrypted_inj_services_x64.dll	1557581712	1557759150	28665	1
1557759030	099ad10b55e74	decrypted_inj_services_Win32.dll	1557581711	1557759151	61440	1

G Code Appendix

```yaml
---

servers:
  - name: frontend.irma
    ip: 172.16.1.30
    ansible_groups: [frontend, sql-server]
    box: quarkslab/debian-9.0.0-amd64
    cpus: 2
    memory: 2048
  - name: brain.irma
    ip: 172.16.1.31
    ansible_groups: [brain]
    box: quarkslab/debian-9.0.0-amd64
    cpus: 2
    memory: 2048
  - name: avs-linux.irma
    ip: 172.16.1.32
    ansible_groups: [avast, avg, bitdefender, clamav, comodo, escan]
    box: quarkslab/debian-9.0.0-amd64
    cpus: 2
    memory: 2048
  - name: mcafee-win.irma
    ip: 172.16.1.33
    ansible_groups: [mcafee-win]
    box: eval-win10x64-enterprise
    cpus: 2
    memory: 2048
    windows: true

libvirt_config:
  driver: kvm
  # connect_via_ssh: true
  # host:
  # username:
  # storage_pool_name:
  # id_ssh_key_file:

ansible_vars:
  irma_environment: production
  vagrant: true
irma_code_archive_generation: False
```

Figure G.34: prod.yml

```
---

- hosts: 127.0.0.1
  connection: local
  tasks:
  - name: Install lief
    command: pip install -r ~/kmno-irma/probe/modules/metadata/lief/requirements.txt


  - name: Install static analyzer
    command: pip install -r ~/kmno-irma/probe/modules/metadata/pe_analyzer/requirements.txt


  - name: Install peid
    command: pip install -r ~/kmno-irma/probe/modules/metadata/peid/requirements.txt
```

Figure G.35: metadataProvisioning.yml

```
---

- hosts: all
  tasks:
  - name: Install clamAV
    become: true
    become_method: sudo
    command: sudo apt-get install clamav-daemon -y

  - name: Find locked process
    shell:  ps aux | grep clamav | awk '  ' '{print$2}'
    register: pid

  - name: Kill locked process
    become: true
    become_method: sudo
    shell: kill "{{item}}"
    with_items: "{{pid.stdout_lines }}"

  - name: Freshclam
    become: true
    become_method: sudo
    command: freshclam
    register: test
    failed_when: false
```

Figure G.36: clamAvFix.yml

H Format of database and scans

H.1 Scan API

The format for a request to create a scan object:

```
    {
      files: [fileext1, fileext2...]
      options:
         probes: list of probes or None for all available,
         force: boolean (default False),
         mimetype_filtering: boolean (default True),
         resubmit_files: boolean (default True),
    }
```

Figure H.37: scan query

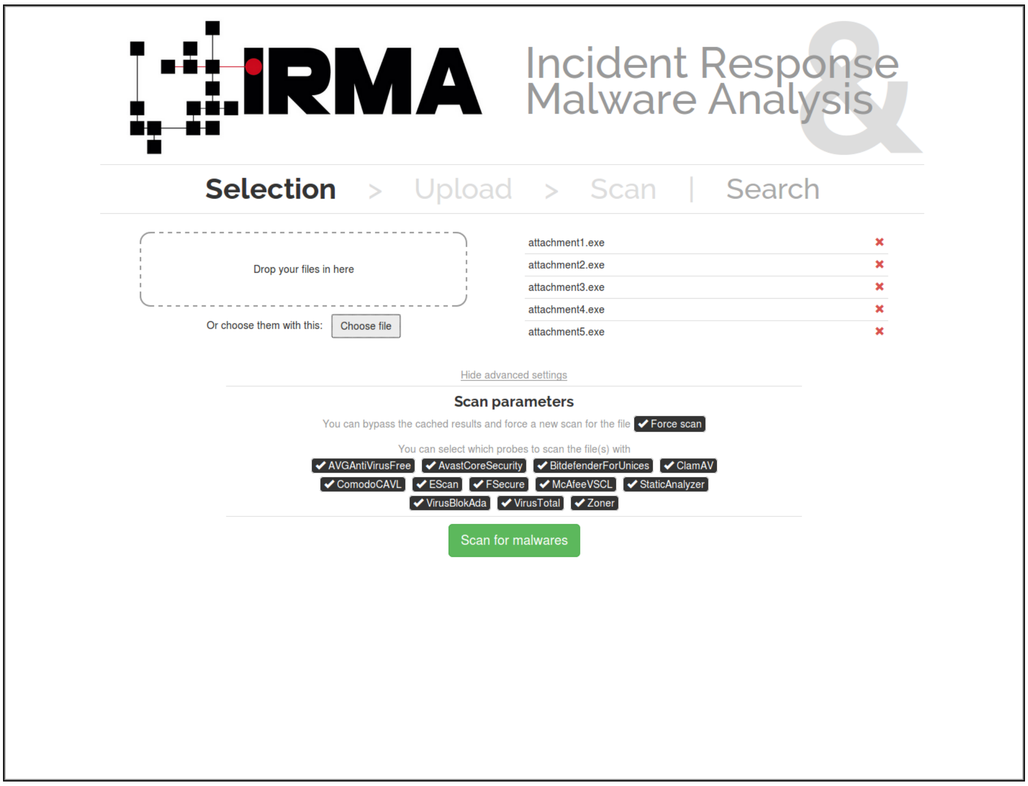
H.2 Result API

The format returned when requesting scan result:

```
[
  {
    "result_id": 0,
    "name": "string",
    "path": "string",
    "file_sha256": "string",
    "parent_file_sha256": "string",
    "scan_id": "string",
    "file_infos": {
      "sha256": "string",
      "sha1": "string",
      "md5": "string",
      "mimetype": "string",
      "timestamp_first_scan": "string",
      "timestamp_last_scan": "string",
      "size": 0,
      "tags": [
        {
          "text": "string"
        }
      ]
    },
    "probe_results": [
      {
        "status": 0,
        "name": "string",
        "type": "external",
        "results": "string",
        "version": "string"
      }
    ],
    "probes_total": 0,
    "probes_finished": 0,
    "status": 0
  }
]
```

Figure H.38: response on result requests

I images


IRMA
Incident Response & Malware Analysis
Selection > Upload > Scan | Search
Choose file
Hide advanced settings
Scan for malwares

Drop your files in here

Or choose them with this: Choose file

- attachment1.exe ✖
- attachment2.exe ✖
- attachment3.exe ✖
- attachment4.exe ✖
- attachment5.exe ✖

Scan parameters

You can bypass the cached results and force a new scan for the file ☑ Force scan

You can select which probes to scan the file(s) with
☑ AVGAntiVirusFree ☑ AvastCoreSecurity ☑ BitdefenderForUnices ☑ ClamAV
☑ ComodoCAVL ☑ EScan ☑ FSecure ☑ McAfeeVSCL ☑ StaticAnalyzer
☑ VirusBlokAda ☑ VirusTotal ☑ Zoner

Figure I.39: IRMA web-gui

J Meeting Logs

J.1 Record of meetings

J.1.1 Dato: 2019/01/11

Laget Trello Board med oppgaver.
Laget forslag til scrum arbeidsmetodikk.
Laget rutiner og regler.

/Olav Henrik Hoggen

J.1.2 19-01-09 week-01

Author: kristian
- Møte Fredag 11.jan kl 10:15-12:00 rom: S213

- Møte Fredag 18.jan kl 10:15-14:15 rom: S312

- Kar.mål: B/A
- Arb.metode diskutert: heller mot Scrum men kan ikke nok til å ta et valg enda

- Trello board e.l. for å dele opp og ut arb.oppg

- Alle fører egne timer arb. med Toggl e.l. Må diskutere om det er rel. hvordan det grupperes

- Rullerer hvem som skriver referat

- Oppgaver:

- Repetere arbeidsmetodikker.

- Utgjøre tanker om oppgaven så kan forstå den og begrense der nødvendig sammen med kongsb/veileder.
- - Gjøre opp tanker rundt forarbeidet og planleggingen
- - Olav: begynner Cuckoo-fikling

- Martin opptatt torsdager, Olav opptatt tirsdager, Nikolai?

J.1.3 19-01-18 week-02

Author: kristian

Meeting w/Basel:
- "task was .. rather large?" - Nikolai
- Input: set of files
- Process: run files into programs
- Output: results from scan

- Keep thinking what we do in regards to task B)

- Start large, then narrow down. "Follow your interest"
- Ask Kongsberg about a case-testing hdd
- Run Cuckoo directly on the openstack as own vm?

J.1.4 28.02.19

Diskuterte fobedringspunkter for utvikling av produktet, som f. Eks
mer organsiert felles arbeid
bedre strukturering ved fordeling av oppgaver.
Mer disiplin

Laget bedre organisert backlog
Planlagt neste sprint
Forbedret møte med oppdragsgiver
Diskutert prioriteringsrekkefølge på implementasjoner

/Martin K

J.1.5 Møte med Kongsberg 1/3/2019

Lisens for VMware ingen problem, send link på mail
De vil ha oppskriften til hvordan alt settes opp
Kan lage opptak fra VMware som demo.
Anbefales å se på og forstå Packer
Fint å legge ned i rapporten hva Packer er, Vagrant, Terraform etc
Tror at Packer kan legge inn lisensnøkkel senere?
Vi kan kjøre hybrid løsning av type 1 og 2 hvis det er nødvendig. (vmware workstation kan imitere esxi)
kan søke på sommerjobb, send mail før start av neste uke
si ifra om vi føler møte kan være nyttig om 2 uker

J.1.6 Sprint 18/03/2019 - 31/03/2019

Sprint planning mål:
Cuckoo som faktisk fungerer, som en probe

Hvordan de ulike antivirus probene fungerer (statisk, dynamisk, header, søking), og hvilke flere vi kan ta i bruk.
Skrive ferdig introduksjonskapittelet.
Skrive kravspek (figurer, tekst, mer)
Skrive om teknologier brukt og teknologier vurdert.
Utforske/inkorporere tags med IRMA
Utforske utgående data etter og under en scan

K Time logged

The log is not accurate for some members for the month of january and february as some of the group did not start to use Toggl until march.

Detailed report

2019-01-01 - 2019-12-31

Total 911 h 00 min

Date	Description	Duration	User
01-09	**planlegge bachelor** Kgb - Bachelor	**2:00:00** 11:00-13:00	Makval
01-11	**planlegge bachelor** Kgb - Bachelor	**1:45:00** 11:30-13:15	Makval
01-11	**(no description)** Kgb - Bachelor	**1:45:00** 13:15-15:00	Makval
01-16	**bachelor arbeid** Kgb - Bachelor	**4:45:00** 10:00-14:45	Makval
01-23	**forprosjekt** Kgb - Bachelor	**4:20:00** 11:35-15:55	Makval
01-24	**forprosjekt** Kgb - Bachelor	**3:50:00** 10:45-14:35	Makval
01-30	**forprosjekt** Kgb - Bachelor	**4:00:00** 12:35-16:35	Makval
01-31	**forprosjekt** Kgb - Bachelor	**3:00:00** 14:35-17:35	Makval
02-01	**(no description)** Kgb - Bachelor	**2:00:00** 07:00-09:00	Nikolai Fau
02-01	**(no description)** Kgb - Bachelor	**2:00:00** 11:00-13:00	Nikolai Fau
02-12	**notepad logging** Kgb - Bachelor	**1:30:00** 12:13-13:43	Makval
02-15	**(no description)** Kgb - Bachelor	**7:03:21** 10:00-17:04	Nikolai Fau
02-15	**(no description)** Kgb - Bachelor	**7:00:00** 10:20-17:20	Makval
02-15	**(no description)** Kgb - Bachelor	**1:59:35** 17:29-19:28	Nikolai Fau
02-15	**(no description)** Kgb - Bachelor	**1:23:24** 19:59-21:23	Nikolai Fau
02-15	**(no description)** Kgb - Bachelor	**0:20:13** 22:15-22:35	Nikolai Fau
02-15	**(no description)** Kgb - Bachelor	**0:14:33** 23:13-23:27	Nikolai Fau
02-18	**(no description)** Kgb - Bachelor	**8:05:00** 10:40-18:45	Makval
02-18	**(no description)** Kgb - Bachelor	**0:08:23** 20:23-20:31	Nikolai Fau

02-18	**(no description)** Kgb - Bachelor	**0:28:56** 21:40-22:09	Nikolai Fau
02-19	**(no description)** Kgb - Bachelor	**0:36:28** 13:55-14:32	Nikolai Fau
02-19	**(no description)** Kgb - Bachelor	**0:47:04** 16:51-17:38	Nikolai Fau
02-19	**(no description)** Kgb - Bachelor	**1:00:00** 17:30-18:30	Nikolai Fau
02-20	**(no description)** Kgb - Bachelor	**5:00:00** 12:00-17:00	Makval
02-21	**(no description)** Kgb - Bachelor	**2:00:00** 09:30-11:30	Makval
02-21	**(no description)** Kgb - Bachelor	**7:25:00** 13:00-20:25	Nikolai Fau
02-22	**(no description)** Kgb - Bachelor	**4:00:00** 09:00-13:00	Makval
02-23	**(no description)** Kgb - Bachelor	**3:00:00** 10:00-13:00	Makval
02-24	**(no description)** Kgb - Bachelor	**3:00:00** 10:00-13:00	Makval
02-24	**(no description)** Kgb - Bachelor	**3:00:00** 14:00-17:00	Makval
02-24	**(no description)** Kgb - Bachelor	**3:25:00** 14:25-17:50	Nikolai Fau
02-25	**RIP Laptop, prøvde å sette opp cuckoo** Kgb - Bachelor	**2:30:00** 22:00-00:30	Makval
02-27	**Scrum planlegging** Kgb - Bachelor	**2:28:00** 14:32-17:00	Nikolai Fau
02-28	**Research** Kgb - Bachelor - [meeting]	**0:00:06** 14:32-14:32	Krisshol
02-28	**Research** Kgb - Bachelor - [meeting]	**0:00:10** 14:33-14:33	Krisshol
02-28	**Research** Kgb - Bachelor - [meeting]	**1:39:49** 14:33-16:13	Krisshol
02-28	**Research** Kgb - Bachelor - [meeting]	**0:40:33** 16:21-17:02	Krisshol
02-28	**Research** Kgb - Bachelor - [meeting]	**0:00:00** 16:21-16:21	Krisshol
03-01	**Ansible/Vagrant troubleshooting** Kgb - Bachelor	**0:30:00** 10:30-11:00	Ohho1588
03-01	**Møter + Packer** Kgb - Bachelor - [meeting, Packer]	**7:00:00** 11:00-18:00	Nikolai Fau
03-01	**Møte med Basel** Kgb - Bachelor	**0:15:00** 11:00-11:15	Ohho1588

03-01	**Ansible/Vagrant troubleshooting** Kgb - Bachelor	**1:15:00** 11:15-12:30	Ohho1588
03-01	**Ansible/Vagrant troubleshooting** Kgb - Bachelor	**0:15:00** 12:30-12:45	Ohho1588
03-01	**Ansible cuckoo** Kgb - Bachelor	**3:30:00** 12:30-16:00	Makval
03-01	**Research** Kgb - Bachelor - [meeting]	**2:23:03** 12:57-15:20	Krisshol
03-01	**Møte med KGB** Kgb - Bachelor	**0:45:00** 13:00-13:45	Ohho1588
03-01	**Gått gjennom IRMA dokumentasjon, kjørt IRMA** Kgb - Bachelor	**1:00:00** 13:45-14:45	Ohho1588
03-01	**RabbitMQ troubleshooting** Kgb - Bachelor	**1:50:00** 14:45-16:35	Ohho1588
03-01	**Research** Kgb - Bachelor - [meeting]	**2:32:13** 15:21-17:53	Krisshol
03-01	**Ansible cuckoo** Kgb - Bachelor	**2:15:00** 16:30-18:45	Makval
03-01	**Research** Kgb - Bachelor - [meeting]	**1:28:53** 17:53-19:22	Krisshol
03-01	**Ansible cuckoo på stasjonær** Kgb - Bachelor	**5:00:00** 20:00-01:00	Makval
03-03	**Packer research** Kgb - Bachelor - [Packer]	**0:03:01** 20:40-20:43	Nikolai Fau
03-03	**Packer research** Kgb - Bachelor - [Packer]	**0:48:16** 20:44-21:32	Nikolai Fau
03-03	**Packer Cuckoo** Kgb - Bachelor - [Packer]	**0:27:48** 21:37-22:05	Nikolai Fau
03-03	**Packer** Kgb - Bachelor - [Packer]	**1:09:40** 22:22-23:32	Nikolai Fau
03-03	**IRMA feilsøking** Kgb - Bachelor - [irma]	**0:30:09** 23:54-00:24	Nikolai Fau
03-04	**IRMA feilsøking** Kgb - Bachelor - [irma]	**0:10:27** 00:32-00:42	Nikolai Fau
03-04	**Packer** Kgb - Bachelor - [Packer]	**0:21:12** 01:24-01:45	Nikolai Fau
03-04	**IRMA feilsøking** Kgb - Bachelor - [irma]	**1:04:56** 13:00-14:05	Nikolai Fau
03-04	**Ubuntu + drivere + min stasjonere != sant** Kgb - Bachelor	**1:00:00** 18:00-19:00	Makval
03-05	**Cuckoo ansible oppsett er teit, fikk ikke utrettet så mye men kom nærmere en løsning** Kgb - Bachelor	**5:00:00** 11:30-16:30	Makval

03-05	**irmacml testing** Kgb - Bachelor	**3:00:00** 11:40-14:40	Ohho1588
03-05	**Packer Cuckoo** Kgb - Bachelor - [Packer]	**2:39:02** 13:16-15:55	Nikolai Fau
03-05	**Research** Kgb - Bachelor - [meeting]	**2:23:26** 13:43-16:07	Krisshol
03-05	**irmacml feilsøking** Kgb - Bachelor	**1:00:00** 14:40-15:40	Ohho1588
03-05	**irmacml feilsøking** Kgb - Bachelor	**1:00:00** 16:00-17:00	Ohho1588
03-05	**Research** Kgb - Bachelor - [meeting]	**1:49:58** 16:24-18:14	Krisshol
03-05	**Docker er teit** Kgb - Bachelor	**1:00:00** 17:10-18:10	Makval
03-06	**Packer Cuckoo** Kgb - Bachelor - [Packer]	**0:30:00** 00:00-00:30	Nikolai Fau
03-06	**lynkurs rapport** Kgb - Bachelor	**1:00:00** 12:05-13:05	Ohho1588
03-06	**Research** Kgb - Bachelor - [crash course, report]	**0:59:05** 12:05-13:05	Krisshol
03-06	**Research** Kgb - Bachelor - [meeting]	**4:29:43** 14:03-18:33	Krisshol
03-06	**IRMA VMWare** Kgb - Bachelor - [irma]	**0:41:28** 14:03-14:45	Nikolai Fau
03-06	**manuell testing av datasett** Kgb - Bachelor	**1:00:00** 14:30-15:30	Ohho1588
03-06	**Prøver å finne ut hvorfor setup scriptet gir 2 failures** Kgb - Bachelor	**1:00:00** 15:30-16:30	Ohho1588
03-06	**Prøver å finne ut hvorfor setup scriptet gir 2 failures** Kgb - Bachelor	**1:00:00** 16:30-17:30	Ohho1588
03-06	**Prøver å finne ut hvorfor setup scriptet gir 2 failures** Kgb - Bachelor	**0:30:00** 17:30-18:00	Ohho1588
03-07	**Packer Cuckoo** Kgb - Bachelor - [Packer]	**0:09:52** 01:19-01:28	Nikolai Fau
03-07	**Packer Cuckoo** Kgb - Bachelor - [Packer]	**0:43:00** 14:15-14:58	Nikolai Fau
03-07	**Cuckoo ansible playbook** Kgb - Bachelor	**2:00:00** 15:15-17:15	Makval
03-08	**Research** Kgb - Bachelor - [irma]	**5:19:04** 11:28-16:48	Krisshol
03-08	**Cuckoo ansible playbook** Kgb - Bachelor	**5:15:00** 11:35-16:50	Makval

03-08	**Packer Cuckoo** Kgb - Bachelor - [Packer]	**4:38:10** 12:08-16:46	Nikolai Fau
03-11	**Cuckoo ansible playbook** Kgb - Bachelor	**6:55:00** 11:30-18:25	Makval
03-11	**Research** Kgb - Bachelor - [irma]	**1:11:37** 14:30-15:42	Krisshol
03-11	**Packer Cuckoo** Kgb - Bachelor - [Packer]	**0:34:00** 19:16-19:50	Nikolai Fau
03-12	**cuckoo windows vm manuel** Kgb - Bachelor	**2:00:00** 11:00-13:00	Makval
03-12	**Packer Cuckoo** Kgb - Bachelor - [Packer]	**1:29:58** 12:30-14:00	Nikolai Fau
03-12	**implement vmware handling in playbook** Kgb - Bachelor	**5:30:00** 13:00-18:30	Makval
03-12	**Packer Cuckoo** Kgb - Bachelor - [Packer]	**1:43:00** 14:17-16:00	Nikolai Fau
03-12	**Research** Kgb - Bachelor - [irma]	**1:11:14** 14:37-15:49	Krisshol
03-12	**Research** Kgb - Bachelor - [irma]	**5:05:53** 16:43-21:48	Krisshol
03-13	**implement vmware handling in playbook** Kgb - Bachelor	**7:41:00** 11:10-18:51	Makval
03-13	**Packer Cuckoo** Kgb - Bachelor - [Packer]	**2:45:00** 13:45-16:30	Nikolai Fau
03-13	**Packer Cuckoo** Kgb - Bachelor - [Packer]	**2:09:18** 19:15-21:25	Nikolai Fau
03-13	**Packer Cuckoo** Kgb - Bachelor - [Packer]	**1:16:00** 21:29-22:45	Nikolai Fau
03-14	**Antivirus** Kgb - Bachelor	**3:00:00** 12:30-15:30	Ohho1588
03-14	**vmware rest api for nettverks adapter håndtering** Kgb - Bachelor	**3:05:00** 15:00-18:05	Makval
03-15	**vmware rest api for nettverks adapter håndtering** Kgb - Bachelor	**4:50:00** 14:10-19:00	Makval
03-15	**Test making** Kgb - Bachelor - [irma]	**1:10:26** 16:40-17:50	Krisshol
03-15	**Test making** Kgb - Bachelor - [irma]	**3:07:48** 18:08-21:16	Krisshol
03-16	**fikler med packer** Kgb - Bachelor - [Packer]	**1:49:31** 15:44-17:34	Nikolai Fau
03-16	**Packer Cuckoo** Kgb - Bachelor - [Packer]	**0:24:32** 18:08-18:33	Nikolai Fau

03-16	**Packer Cuckoo** Kgb - Bachelor - [Packer]	**0:34:13** 20:15-20:49	Nikolai Fau
03-16	**Packer Progress** Kgb - Bachelor - [Packer]	**1:18:55** 21:58-23:17	Nikolai Fau
03-17	**Packer ssh** Kgb - Bachelor - [Packer]	**0:56:43** 13:40-14:36	Nikolai Fau
03-18	**teste cuckoo med packer images fra bunn av** Kgb - Bachelor	**2:56:00** 10:14-13:10	Makval
03-18	**rapport research/skriving** Kgb - Bachelor	**1:00:00** 11:00-12:00	Ohho1588
03-18	**scrum møte** Kgb - Bachelor - [meeting]	**0:45:00** 14:00-14:45	Nikolai Fau
03-18	**sprint planning møte** Kgb - Bachelor	**1:00:00** 14:00-15:00	Ohho1588
03-18	**Scrum Planning** Kgb - Bachelor - [irma]	**0:49:05** 14:05-14:54	Krisshol
03-18	**teste cuckoo med packer images fra bunn av** Kgb - Bachelor	**3:45:00** 14:10-17:55	Makval
03-18	**rapport skriving intro + req** Kgb - Bachelor	**1:00:00** 15:00-16:00	Ohho1588
03-18	**Cuckoo-ansible** Kgb - Bachelor - [pair programming]	**1:14:49** 15:30-16:45	Krisshol
03-18	**rapport skriving req + tek** Kgb - Bachelor	**1:00:00** 16:00-17:00	Ohho1588
03-18	**General research** Kgb - Bachelor - [pair programming]	**0:46:43** 16:45-17:31	Krisshol
03-18	**rapport skriving tek** Kgb - Bachelor	**1:00:00** 17:00-18:00	Ohho1588
03-18	**General research** Kgb - Bachelor - [pair programming]	**1:27:48** 18:07-19:35	Krisshol
03-19	**scrum møte** Kgb - Bachelor - [meeting]	**0:25:00** 00:30-00:55	Nikolai Fau
03-19	**fikset bitdefender failure + litt rapportskriving** Kgb - Bachelor	**1:00:00** 10:00-11:00	Ohho1588
03-19	**teste cuckoo med packer images fra bunn av** Kgb - Bachelor	**3:25:00** 10:25-13:50	Makval
03-19	**Antivirus** Kgb - Bachelor	**1:00:00** 11:00-12:00	Ohho1588
03-19	**teste cuckoo med packer images fra bunn av** Kgb - Bachelor	**2:55:00** 15:10-18:05	Makval
03-19	**Packer ssh** Kgb - Bachelor - [Packer]	**1:01:10** 22:07-23:08	Nikolai Fau

03-19	**prøvde å fikse vm start med ansible igjen, funka ikke:(** Kgb - Bachelor	**1:00:00** 23:32-00:32	Makval
03-20	**Packer ssh** Kgb - Bachelor - [Packer]	**0:10:00** 00:30-00:40	Nikolai Fau
03-20	**cuckoo start vm i ansible playbook, still** Kgb - Bachelor	**3:23:00** 10:57-14:20	Makval
03-20	**malware research** Kgb - Bachelor - [cuckoo, Packer, research]	**2:00:10** 12:30-14:30	Nikolai Fau
03-20	**cuckoo start vm i ansible playbook** Kgb - Bachelor	**4:20:00** 14:40-19:00	Makval
03-20	**Thesis writing** Kgb - Bachelor - [report]	**0:19:32** 16:51-17:10	Krisshol
03-20	**Reset prep** Kgb - Bachelor	**1:07:14** 17:34-18:41	Krisshol
03-20	**kgb reinstallere +** Kgb - Bachelor	**3:35:00** 21:10-00:45	Makval
03-21	**antivirus og feilsøking** Kgb - Bachelor	**2:30:00** 10:00-12:30	Ohho1588
03-21	**irma packer** Kgb - Bachelor - [irma, Packer]	**0:39:41** 15:00-15:40	Nikolai Fau
03-22	**antivirus og irma feilsøking** (no project)	**1:45:00** 10:15-12:00	Ohho1588
03-22	**møte, og planlegging og snakking og stuff** Kgb - Bachelor	**5:20:00** 11:20-16:40	Makval
03-22	**Møte + Yara** Kgb - Bachelor - [irma, meeting, research]	**4:13:50** 11:30-15:44	Nikolai Fau
03-22	**planlegging** Kgb - Bachelor	**2:00:00** 12:00-14:00	Ohho1588
03-22	**Plan meeting** Kgb - Bachelor	**1:50:30** 12:30-14:20	Krisshol
03-22	**Antivirus** Kgb - Bachelor	**2:30:00** 14:00-16:30	Ohho1588
03-22	**Kiosk design** Kgb - Bachelor	**0:51:22** 14:37-15:29	Krisshol
03-22	**Kiosk design** Kgb - Bachelor	**0:45:28** 15:42-16:27	Krisshol
03-22	**Kiosk design** Kgb - Bachelor	**1:29:24** 16:50-18:19	Krisshol
03-22	**Packer and cleanup in code** Kgb - Bachelor - [cuckoo, Packer]	**3:25:53** 22:30-01:56	Nikolai Fau
03-23	**vmware er bæsj** Kgb - Bachelor	**2:02:00** 00:30-02:32	Makval

03-25	**Antivirus** Kgb - Bachelor	**3:00:00** 09:00-12:00	Ohho1588
03-25	**irma feilsøling** Kgb - Bachelor	**2:00:00** 12:00-14:00	Ohho1588
03-25	**kmno-irma i Git** Kgb - Bachelor	**1:00:00** 15:00-16:00	Krisshol
03-26	**Vm start..** Kgb - Bachelor	**1:00:00** 09:30-10:30	Makval
03-26	**møte** Kgb - Bachelor	**1:00:00** 11:00-12:00	Ohho1588
03-26	**Møte** Kgb - Bachelor - [irma, meeting, research]	**2:00:00** 11:00-13:00	Nikolai Fau
03-26	**Møte + AVsammen med olav** Kgb - Bachelor	**2:00:00** 11:00-13:00	Makval
03-26	**kgb møte** Kgb - Bachelor	**0:55:00** 11:05-12:00	Krisshol
03-26	**antivirus og rapportskriving** Kgb - Bachelor	**4:30:00** 12:00-16:30	Ohho1588
03-26	**Skrive rapport + misc** Kgb - Bachelor	**1:50:00** 14:30-16:20	Makval
03-26	**prøvde å fikse eset, comodo og clam er humørsyke** Kgb - Bachelor	**1:25:00** 16:50-18:15	Makval
03-26	**Yara** Kgb - Bachelor - [irma, research]	**1:57:22** 20:00-21:57	Nikolai Fau
03-27	**prøvde vm start igjen, gitt opp** Kgb - Bachelor	**3:50:00** 11:00-14:50	Makval
03-27	**antivirus og rapportskriving** Kgb - Bachelor	**2:00:00** 12:00-14:00	Ohho1588
03-27	**Yara** Kgb - Bachelor	**1:45:00** 13:00-14:45	Nikolai Fau
03-27	**kmno-irma i Git** Kgb - Bachelor	**1:26:08** 14:00-15:26	Krisshol
03-27	**sjekker at cuckoo fungerer som den skal** Kgb - Bachelor	**3:55:00** 14:50-18:45	Makval
03-27	**packer windows** Kgb - Bachelor - [Packer]	**0:43:00** 21:30-22:13	Nikolai Fau
03-28	**packer windows + yara** Kgb - Bachelor - [Packer]	**1:53:10** 00:06-02:00	Nikolai Fau
03-28	**Yara** Kgb - Bachelor	**1:05:40** 02:15-03:20	Nikolai Fau
03-28	**skrev på rapport** Kgb - Bachelor	**1:00:00** 10:00-11:00	Makval
03-28	**irma** Kgb - Bachelor - [irma]	**1:00:00** 11:20-12:20	Nikolai Fau

03-28	**yara + irma**	**3:35:00**	Nikolai Fau
	Kgb - Bachelor - [irma]	14:40-18:15	
03-28	**kmno-irma i Git**	**1:32:23**	Krisshol
	Kgb - Bachelor	15:55-17:28	
03-28	**kmno-irma i Git**	**0:41:11**	Krisshol
	Kgb - Bachelor	17:36-18:17	
03-28	**få cuckoo til å kjøre**	**2:45:00**	Makval
	Kgb - Bachelor	21:15-00:00	
03-28	**av-win og yara finish upper**	**1:43:30**	Nikolai Fau
	Kgb - Bachelor - [irma]	21:25-23:08	
03-28	**packer windows**	**0:26:00**	Nikolai Fau
	Kgb - Bachelor - [Packer]	23:40-00:06	
03-29	**rapportskriving**	**2:25:00**	Ohho1588
	Kgb - Bachelor	11:00-13:25	
03-29	**Thesis writing**	**0:58:11**	Krisshol
	Kgb - Bachelor - [report]	18:00-18:59	
03-31	**packer windows**	**0:40:00**	Nikolai Fau
	Kgb - Bachelor - [Packer]	12:00-12:40	
03-31	**irma production bugs**	**4:00:00**	Nikolai Fau
	Kgb - Bachelor - [irma]	16:22-20:22	
03-31	**irma production bugs**	**1:01:33**	Nikolai Fau
	Kgb - Bachelor - [irma]	20:23-21:24	
04-01	**rapportskriving**	**2:00:00**	Ohho1588
	Kgb - Bachelor	10:00-12:00	
04-01	**få cuckoo til å kjøre**	**5:15:00**	Makval
	Kgb - Bachelor	11:00-16:15	
04-01	**sett på wireshark**	**1:30:00**	Ohho1588
	Kgb - Bachelor	12:00-13:30	
04-01	**tcpdump av scan**	**1:00:00**	Ohho1588
	Kgb - Bachelor	14:00-15:00	
04-02	**Yara test**	**1:31:50**	Nikolai Fau
	Kgb - Bachelor - [irma]	00:00-01:32	
04-02	**Klargjøre for mer cuckoo playbook testing**	**0:35:00**	Makval
	Kgb - Bachelor	09:15-09:50	
04-02	**Forbedre cuckoo playbook**	**4:05:00**	Makval
	Kgb - Bachelor	10:25-14:30	
04-02	**diskusjon**	**0:30:00**	Ohho1588
	Kgb - Bachelor	12:00-12:30	
04-02	**windows + pipelining + rapport**	**1:55:00**	Nikolai Fau
	Kgb - Bachelor - [Packer, report, research]	12:05-14:00	
04-02	**ansible script antivirus**	**2:00:00**	Ohho1588
	Kgb - Bachelor	12:30-14:30	

04-02	**Forbedre cuckoo playbook** Kgb - Bachelor	**2:50:00** 14:40-17:30	Makval
04-02	**ansible script antivirus** Kgb - Bachelor	**2:35:00** 15:00-17:35	Ohho1588
04-02	**kmno-irma i Git (redo)** Kgb - Bachelor	**2:19:07** 15:00-17:19	Krisshol
04-02	**windows + pipelining + rapport** Kgb - Bachelor - [Packer, report, research]	**2:15:00** 15:15-17:30	Nikolai Fau
04-03	**cuckoo** Kgb - Bachelor	**0:30:00** 10:15-10:45	Makval
04-03	**ansible script antivirus** Kgb - Bachelor	**1:00:00** 14:00-15:00	Ohho1588
04-03	**packer windows + pipeline irma scans** Kgb - Bachelor - [irma, Packer, research]	**2:33:44** 21:00-23:33	Nikolai Fau
04-04	**pipelining provisioning** Kgb - Bachelor - [irma, research]	**1:25:20** 02:07-03:32	Nikolai Fau
04-04	**ansible script antivirus** Kgb - Bachelor	**3:00:00** 11:00-14:00	Ohho1588
04-04	**pipelining provisioning** Kgb - Bachelor - [irma, research]	**1:20:01** 13:30-14:50	Nikolai Fau
04-04	**ansible script antivirus** Kgb - Bachelor	**1:40:00** 14:30-16:10	Ohho1588
04-04	**pipelining + scalability** Kgb - Bachelor - [irma, research]	**2:52:33** 14:50-17:42	Nikolai Fau
04-04	**vmware forskjellig ip samme base image** Kgb - Bachelor	**4:00:00** 15:10-19:10	Makval
04-04	**pipelining + scalability** Kgb - Bachelor - [irma, research]	**0:02:14** 17:50-17:52	Nikolai Fau
04-04	**cuckoo test** Kgb - Bachelor	**3:45:00** 21:25-01:10	Makval
04-04	**packer windows** Kgb - Bachelor - [Packer]	**0:04:41** 23:50-23:55	Nikolai Fau
04-05	**cuckoo test, dokumentasjon og rapport** Kgb - Bachelor	**1:00:00** 11:00-12:00	Makval
04-05	**ansible script antivirus feilsøking** Kgb - Bachelor	**4:00:00** 11:30-15:30	Ohho1588
04-05	**Cuckoo doku, pluss gr samtaler** Kgb - Bachelor	**4:25:00** 12:40-17:05	Makval
04-05	**Diskusjon** Kgb - Bachelor - [meeting]	**1:00:00** 13:00-14:00	Nikolai Fau
04-05	**kgb møte** Kgb - Bachelor	**0:20:00** 14:45-15:05	Krisshol

04-05	**Kiosk design** Kgb - Bachelor	**3:53:33** 15:25-19:19	Krisshol
04-08	**cuckoo dokumentasjon og testing av playbook.** Kgb - Bachelor	**8:10:00** 11:50-20:00	Makval
04-08	**clamAV fungerer nå** Kgb - Bachelor	**2:00:00** 12:00-14:00	Ohho1588
04-08	**Sliter plutselig med noen andre AV-er** Kgb - Bachelor	**2:00:00** 14:00-16:00	Ohho1588
04-09	**ansible script antivirus** Kgb - Bachelor	**2:00:00** 11:00-13:00	Ohho1588
04-09	**installere esx** Kgb - Bachelor	**1:20:00** 11:50-13:10	Makval
04-09	**Sett på rapporter** Kgb - Bachelor	**1:00:00** 13:00-14:00	Ohho1588
04-09	**prøve å innstalere esx på nytt, riktig versjon kanskje...** Kgb - Bachelor	**1:15:00** 17:45-19:00	Makval
04-09	**Kiosk design** Kgb - Bachelor	**1:30:00** 18:30-20:00	Krisshol
04-10	**winrm fix + starting av tester** Kgb - Bachelor - [irma, Packer]	**0:30:00** 00:30-01:00	Nikolai Fau
04-10	**reading IRMA code** Kgb - Bachelor - [irma, research]	**1:27:00** 01:00-02:27	Nikolai Fau
04-10	**api research, NSRL, known good/bad** Kgb - Bachelor - [irma, Packer, research]	**1:21:27** 12:03-13:25	Nikolai Fau
04-10	**AV-debugging** Kgb - Bachelor	**3:00:00** 13:00-16:00	Ohho1588
04-10	**Testing og oppsett** Kgb - Bachelor - [irma, Packer, research]	**0:10:00** 13:24-13:34	Nikolai Fau
04-10	**Rapport + speed tests** Kgb - Bachelor - [irma, report, research]	**1:00:00** 13:44-14:44	Nikolai Fau
04-10	**Kiosk design** Kgb - Bachelor	**0:30:15** 17:45-18:16	Krisshol
04-10	**NSRL** Kgb - Bachelor - [irma, research]	**1:29:35** 20:32-22:01	Nikolai Fau
04-10	**NSRL** Kgb - Bachelor - [irma, research]	**1:39:00** 23:21-01:00	Nikolai Fau
04-11	**AV-debugging** Kgb - Bachelor	**3:00:00** 12:00-15:00	Ohho1588
04-11	**NSRL debug** Kgb - Bachelor - [irma, Packer, report, research]	**0:42:00** 15:25-16:07	Nikolai Fau
04-11	**NSRL debug** Kgb - Bachelor	**0:27:00** 16:15-16:42	Nikolai Fau

04-11	**Kiosk design** Kgb - Bachelor	**2:00:00** 21:00-23:00	Krisshol
04-12	**NSRL debug** Kgb - Bachelor	**1:00:02** 01:30-02:30	Nikolai Fau
04-12	**Sett på output av scans og windows provisjonering** Kgb - Bachelor	**3:00:00** 13:00-16:00	Ohho1588
04-12	**NSRL debug + Pipeline planning + IRMA research** Kgb - Bachelor - [irma, meeting, pair programming, research]	**2:58:12** 13:10-16:09	Nikolai Fau
04-12	**Cuckoo som irma probe** Kgb - Bachelor	**1:30:00** 13:30-15:00	Makval
04-12	**Cuckoo som irma probe** Kgb - Bachelor	**1:35:00** 16:00-17:35	Makval
04-12	**NSRL debug** Kgb - Bachelor	**0:00:02** 16:04-16:04	Nikolai Fau
04-12	**kode forståelse** Kgb - Bachelor - [irma, research]	**0:30:00** 16:55-17:25	Nikolai Fau
04-12	**kode forståelse** Kgb - Bachelor - [irma, research]	**1:05:00** 21:10-22:15	Nikolai Fau
04-13	**NSRL kode** Kgb - Bachelor - [irma, research]	**0:45:00** 01:10-01:55	Nikolai Fau
04-14	**NSRL kode** Kgb - Bachelor - [irma, research]	**1:00:00** 01:00-02:00	Nikolai Fau
04-14	**NSRL koding** Kgb - Bachelor - [irma]	**4:00:00** 14:07-18:07	Nikolai Fau
04-14	**NSRL koding** Kgb - Bachelor - [irma]	**0:32:21** 18:47-19:20	Nikolai Fau
04-14	**kode research** Kgb - Bachelor - [irma, research]	**0:51:00** 19:50-20:41	Nikolai Fau
04-17	**rapport skriving** Kgb - Bachelor	**1:15:00** 15:15-16:30	Makval
04-17	**Kiosk design** Kgb - Bachelor	**1:20:35** 23:06-00:27	Krisshol
04-18	**Kiosk design** Kgb - Bachelor	**0:45:02** 22:40-23:25	Krisshol
04-18	**Kiosk design** Kgb - Bachelor	**1:31:19** 23:50-01:22	Krisshol
04-21	**windows provisjonering** Kgb - Bachelor	**1:00:00** 11:00-12:00	Ohho1588
04-21	**linux AV roller** Kgb - Bachelor	**3:00:00** 12:00-15:00	Ohho1588
04-22	**linux AV roller** Kgb - Bachelor	**3:00:00** 13:00-16:00	Ohho1588

04-22	**rapport m.m** Kgb - Bachelor	**0:15:00** 21:20-21:35	Makval
04-23	**oversikt over hvilke AVer som feiler og hvor de gjør det** Kgb - Bachelor	**2:00:00** 09:00-11:00	Ohho1588
04-23	**rapport** Kgb - Bachelor	**0:36:00** 11:00-11:36	Ohho1588
04-23	**ferdig av linux roller** Kgb - Bachelor	**2:24:00** 11:36-14:00	Ohho1588
04-23	**Sjekket om noen av-errors har en enkel fix** Kgb - Bachelor	**1:47:00** 14:00-15:47	Ohho1588
04-23	**rapport m.m** Kgb - Bachelor	**0:15:00** 18:45-19:00	Makval
04-23	**rapport m.m** Kgb - Bachelor	**0:50:00** 19:10-20:00	Makval
04-24	**Kiosk design** Kgb - Bachelor	**3:17:47** 18:42-22:00	Krisshol
04-24	**Kiosk design** Kgb - Bachelor	**0:56:49** 23:20-00:17	Krisshol
04-26	**Kiosk design** Kgb - Bachelor	**1:02:28** 14:47-15:49	Krisshol
04-26	**Kiosk design** Kgb - Bachelor	**1:36:49** 20:42-22:19	Krisshol
04-26	**Kiosk design** Kgb - Bachelor	**1:15:56** 22:55-00:11	Krisshol
04-30	**rapportskriving** Kgb - Bachelor	**2:00:00** 12:00-14:00	Ohho1588
04-30	**(no description)** Kgb - Bachelor	**5:00:00** 18:00-23:00	Nikolai Fau
04-30	**Kiosk design** Kgb - Bachelor	**0:36:09** 19:00-19:36	Krisshol
04-30	**Kiosk design** Kgb - Bachelor	**3:37:00** 21:30-01:07	Krisshol
05-01	**rapportskriving** Kgb - Bachelor	**0:20:00** 13:40-14:00	Ohho1588
05-01	**møte om rapportstruktur og rapportskriving** Kgb - Bachelor	**2:00:00** 14:00-16:00	Ohho1588
05-01	**Møte** Kgb - Bachelor	**1:05:00** 14:40-15:45	Krisshol
05-01	**Kiosk design** Kgb - Bachelor	**1:34:32** 15:56-17:30	Krisshol
05-01	**use case** Kgb - Bachelor	**1:00:00** 16:00-17:00	Ohho1588
05-02	**studert win av roller (to måter å implementere avhengig om installasjonsscript brukes)** Kgb - Bachelor	**2:30:00** 08:00-10:30	Ohho1588

05-02	**win av rolle implementasjon** Kgb - Bachelor	**1:30:00** 10:30-12:00	Ohho1588
05-02	**rapportskriving - kravspek** Kgb - Bachelor	**3:10:00** 12:00-15:10	Ohho1588
05-03	**Fixing introduction** Kgb - Bachelor	**3:03:31** 01:32-04:36	Krisshol
05-03	**rapportskriving kravspek** Kgb - Bachelor	**1:30:00** 12:00-13:30	Ohho1588
05-03	**rapportskriving use case & sekvensdiagram** Kgb - Bachelor	**4:00:00** 14:00-18:00	Ohho1588
05-03	**rapport + møte** Kgb - Bachelor	**3:00:00** 14:15-17:15	Makval
05-03	**Fixing introduction** Kgb - Bachelor	**1:42:03** 14:59-16:41	Krisshol
05-03	**(no description)** Kgb - Bachelor	**2:00:00** 15:00-17:00	Nikolai Fau
05-04	**(no description)** Kgb - Bachelor	**4:00:00** 14:00-18:00	Nikolai Fau
05-04	**rapport** Kgb - Bachelor	**1:30:00** 15:30-17:00	Makval
05-05	**rapportskriving malware & malware detection** Kgb - Bachelor	**6:00:00** 09:00-15:00	Ohho1588
05-05	**(no description)** Kgb - Bachelor	**5:00:00** 12:00-17:00	Nikolai Fau
05-06	**Fixing introduction** Kgb - Bachelor	**4:03:07** 00:00-04:03	Krisshol
05-06	**rapportskriving** Kgb - Bachelor	**2:00:00** 09:00-11:00	Ohho1588
05-06	**rapportskriving** Kgb - Bachelor	**3:00:00** 12:00-15:00	Ohho1588
05-06	**rapport** Kgb - Bachelor	**2:45:00** 13:45-16:30	Makval
05-06	**rapportskriving** Kgb - Bachelor	**2:30:00** 15:30-18:00	Ohho1588
05-06	**rapport** Kgb - Bachelor	**0:25:00** 17:10-17:35	Makval
05-07	**rapportskriving** Kgb - Bachelor	**7:15:00** 10:00-17:15	Ohho1588
05-07	**rapport** Kgb - Bachelor	**5:15:00** 12:45-18:00	Makval
05-07	**(no description)** Kgb - Bachelor	**9:00:00** 15:00-00:00	Nikolai Fau
05-08	**rapportskriving** Kgb - Bachelor	**1:30:00** 08:30-10:00	Ohho1588

05-08	**rapport** Kgb - Bachelor	**5:00:00** 12:10-17:10	Makval
05-08	**(no description)** Kgb - Bachelor	**2:00:00** 15:00-17:00	Nikolai Fau
05-08	**(no description)** Kgb - Bachelor	**3:00:00** 18:00-21:00	Nikolai Fau
05-08	**lese og kommentere rapport** Kgb - Bachelor	**2:10:00** 21:15-23:25	Makval
05-08	**Fixing requirements** Kgb - Bachelor	**1:49:16** 23:26-01:15	Krisshol
05-09	**Fixing theory** Kgb - Bachelor	**3:08:10** 01:39-04:47	Krisshol
05-09	**(no description)** Kgb - Bachelor	**6:00:00** 12:00-18:00	Nikolai Fau
05-09	**Fixing theory** Kgb - Bachelor	**1:30:01** 21:20-22:50	Krisshol
05-10	**Fixing theory** Kgb - Bachelor	**4:27:23** 00:04-04:31	Krisshol
05-10	**rapportskriving** Kgb - Bachelor	**1:00:00** 10:00-11:00	Ohho1588
05-10	**Møte** Kgb - Bachelor	**2:40:00** 11:00-13:40	Krisshol
05-10	**møte** Kgb - Bachelor	**3:00:00** 11:00-14:00	Makval
05-10	**møter** Kgb - Bachelor	**3:00:00** 11:00-14:00	Ohho1588
05-10	**Fixing theory** Kgb - Bachelor	**0:48:44** 15:30-16:18	Krisshol
05-11	**rapportskriving - testing og analyse** Kgb - Bachelor	**7:15:00** 12:00-19:15	Ohho1588
05-11	**rapport** Kgb - Bachelor	**7:15:00** 12:00-19:15	Makval
05-11	**(no description)** Kgb - Bachelor	**5:00:00** 13:00-18:00	Nikolai Fau
05-11	**Fixing theory** Kgb - Bachelor	**1:00:00** 18:30-19:30	Krisshol
05-11	**Fixing theory** Kgb - Bachelor	**2:31:20** 22:00-00:31	Krisshol
05-12	**(no description)** Kgb - Bachelor	**4:00:00** 12:00-16:00	Nikolai Fau
05-12	**Fixing theory** Kgb - Bachelor	**0:20:00** 18:00-18:20	Krisshol
05-12	**Adding theory - IRMA** Kgb - Bachelor	**0:30:00** 20:00-20:30	Krisshol

05-12	**Adding theory - IRMA** Kgb - Bachelor	**3:19:53** 21:35-00:54	Krisshol
05-12	**så på esxi støtte** Kgb - Bachelor	**1:10:00** 22:30-23:40	Makval
05-13	**Adding theory - IRMA** Kgb - Bachelor	**2:50:29** 01:08-03:59	Krisshol
05-13	**(no description)** Kgb - Bachelor	**10:00:00** 07:00-17:00	Nikolai Fau
05-13	**legge till Cuckoo som probe i IRMA** Kgb - Bachelor	**5:00:00** 11:00-16:00	Makval
05-13	**rapportskriving** Kgb - Bachelor	**5:00:00** 11:00-16:00	Ohho1588
05-14	**(no description)** Kgb - Bachelor	**10:00:00** 09:00-19:00	Nikolai Fau
05-14	**rapportskriving** Kgb - Bachelor	**8:10:00** 10:00-18:10	Ohho1588
05-14	**rapport, cuckoo probe, drøfting av rapport** Kgb - Bachelor	**8:15:00** 10:00-18:15	Makval
05-14	**reinstallere + fikse irma som jeg sletta** Kgb - Bachelor	**0:30:00** 18:45-19:15	Makval
05-14	**reinstallere + fikse irma som jeg sletta** Kgb - Bachelor	**0:45:00** 20:00-20:45	Makval
05-14	**reinstallere + fikse irma som jeg sletta** Kgb - Bachelor	**5:30:00** 20:00-01:30	Makval
05-15	**rapportskriving + testing** Kgb - Bachelor	**5:00:00** 10:00-15:00	Ohho1588
05-15	**rapport skriving, cuckoo probe** Kgb - Bachelor	**4:40:00** 10:20-15:00	Makval
05-15	**(no description)** Kgb - Bachelor - [report]	**10:00:00** 11:00-21:00	Nikolai Fau
05-15	**General report writing - kiosk** Kgb - Bachelor	**8:48:49** 13:30-22:18	Krisshol
05-15	**rapport skriving, cuckoo probe** Kgb - Bachelor	**6:20:00** 15:30-21:50	Makval
05-15	**rapportskriving + testing** Kgb - Bachelor	**6:30:00** 15:30-22:00	Ohho1588
05-15	**rapport skriving, cuckoo probe** Kgb - Bachelor	**1:10:00** 22:50-00:00	Makval
05-15	**General report writing - kiosk** Kgb - Bachelor	**0:44:21** 22:57-23:41	Krisshol
05-15	**rapportskriving + testing** Kgb - Bachelor	**1:00:00** 23:00-00:00	Ohho1588

05-16	**rapport skriving, cuckoo probe** Kgb - Bachelor	**6:00:00** 10:05-16:05	Makval
05-16	**(no description)** Kgb - Bachelor - [report]	**12:00:00** 11:00-23:00	Nikolai Fau
05-16	**rapportskriving + testing** Kgb - Bachelor	**6:00:00** 12:00-18:00	Ohho1588
05-16	**General report writing - kiosk** Kgb - Bachelor	**0:49:12** 15:13-16:02	Krisshol
05-16	**rapport skriving, cuckoo probe** Kgb - Bachelor	**4:10:00** 16:50-21:00	Makval
05-16	**General report writing - kiosk** Kgb - Bachelor	**1:52:48** 18:00-19:53	Krisshol
05-16	**cuckoo probe** Kgb - Bachelor	**1:35:00** 21:35-23:10	Makval
05-16	**cuckoo probe** Kgb - Bachelor	**1:30:00** 23:35-01:05	Makval
05-17	**rapportskriving + testing** Kgb - Bachelor	**3:00:00** 09:00-12:00	Ohho1588
05-17	**cuckoo probe debugging** Kgb - Bachelor	**3:00:00** 23:00-02:00	Makval
05-18	**rapportskriving** Kgb - Bachelor	**12:00:00** 10:00-22:00	Ohho1588
05-18	**rapport lesing** Kgb - Bachelor	**11:00:00** 11:15-22:15	Makval
05-18	**Fixing stuff** Kgb - Bachelor	**4:14:08** 11:45-15:59	Krisshol
05-18	**(no description)** Kgb - Bachelor - [report]	**7:06:00** 13:00-20:06	Nikolai Fau
05-18	**Fixing stuff** Kgb - Bachelor	**3:26:45** 16:13-19:40	Krisshol
05-18	**Fixing stuff** Kgb - Bachelor	**2:37:57** 19:58-22:36	Krisshol
05-18	**cuckoo probe** Kgb - Bachelor	**0:15:00** 23:00-23:15	Makval
05-19	**cuckoo probe** Kgb - Bachelor	**2:00:00** 00:40-02:40	Makval
05-19	**irma** Kgb - Bachelor	**1:00:00** 09:40-10:40	Makval
05-19	**(no description)** Kgb - Bachelor - [report]	**13:00:00** 10:00-23:00	Nikolai Fau
05-19	**rapportskriving** Kgb - Bachelor	**13:00:00** 10:00-23:00	Ohho1588
05-19	**report** Kgb - Bachelor	**6:35:00** 11:50-18:25	Makval

05-19	**report + cuckoo** Kgb - Bachelor	**4:35:00** 18:55-23:30	Makval

Created with toggl.com

L Slack communication logs

```
[
    {
        "client_msg_id": "dcc77fa2-b27c-458e-92f8-89d82bb629e8",
        "type": "message",
        "text": "Hva jeg har gjort:\nPrivate github fork av IRMA, forsøkt
         ↪ videre med rabbitmq remote denied bugen.\n\nHva jeg
         ↪ planlegger:\nTeste setup av IRMA på egen maskin, sjekke om
         ↪ rabbitmq bugen skjer der også.\nHvis feilen skjer: Fortsette
         ↪ med rabbitmq, finne ut hvor env variablen faktisk må
         ↪ settes.\nOm den går opp: manuelt sende inn og teste dataset.
         ↪ \nEvt: Se på ssh auth mellom kgb og github for pull av
         ↪ repository.\n\nPotensielle hindringer:\nRabbitmq modulen
         ↪ utdatert og må erstattes eller oppdateres\/skrives om. ",
        "user": "UGMTT84F9",
        "ts": "1551498669.013400"
    }
][
    {
        "client_msg_id": "45276472-b498-407b-9afc-2f3cbc6c8ce6",
        "type": "message",
        "text": "Hva jeg har gjort:\nSett gjennom irma dokumentasjon,
         ↪ lastet ned Irma command line API og prøvd å få den til å
         ↪ fungere. Ingen fremgang i dag.\n\nHva jeg
         ↪ planlegger:\nFortsette med irmacml feilsøking, evt begynne på
         ↪ rapport",
        "user": "UGMNQ83QS",
        "ts": "1551802870.018700"
    },
    {
        "user": "UGQTTS29M",
        "type": "message",
        "subtype": "channel_join",
        "ts": "1551811677.019000",
        "text": "<@UGQTTS29M> has joined the channel"
    },
    {
        "client_msg_id": "b4ce090b-8515-47d0-97b0-d4603d033ef4",
        "type": "message",
        "text": "Hva har jeg gjort:\nJeg prøvd å få ansible playbook til
         ↪ å fungere, ikke helt i mål enda. Slåss med docker for å kjøre
         ↪ virtualbox inne i docker, fikk det ikke helt til men virker
         ↪ greit nok.\nHva jeg planlegger:\nFullføre overnevnte og få
         ↪ teste cuckoo ordentlig\nPotensielle hindringer\nAnsible er
         ↪ teit, cuckoo image er knot.",
        "user": "UGQTTS29M",
        "ts": "1551814570.020400"
    },
    {
        "client_msg_id": "a9c8ff54-bc4e-4cb0-966e-2e8d1978336c",
        "type": "message",
        "text": "Hva har jeg gjort:\nPrøvd å lage et image for en cuckoo
         ↪  guest (win10), fikk ikke sett om det var suksessfult ettersom
         ↪  tmux sesjon var borte når jeg kom tilbake, antar at pc ble
         ↪  restarta",
        "user": "UGL9Y07FB",
        "ts": "1551828944.022300"
    },
    {
        "client_msg_id": "ebd2510d-24fa-4cf9-a6ea-abbfec439af1",
        "type": "message",
        "text": "Hva jeg har gjort:\nPort forwardet IRMA frontenden,
         ↪  kranglet med CLI-apiet til IRMA\n\nHva jeg
         ↪  planlegger:\nTroubleshoote problematisk IRMA der filer ikke
         ↪  vil scannes, så får aldri resultat\n\nPotensielle
         ↪  hindringer:\nIRMA ikke ordentlig, ikke kontakt mellom
         ↪  delene?",
        "user": "UGMTT84F9",
        "ts": "1551838422.023600",
        "edited": {
            "user": "UGMTT84F9",
            "ts": "1551838582.000000"
        }
    }
][
    {
        "client_msg_id": "7a7c8fe3-abc7-4b5a-8b3c-7b4215ff16d2",
        "type": "message",
        "text": "Hva jeg har gjort:\nVært på lynkurs, manuell testing av
         ↪  datasett, prøvd å fikse failures fra setup.py test\n\nHva jeg
         ↪  planlegger:\nFortsette å se på setup.py, legge til flere
         ↪  antivirus",
        "user": "UGMNQ83QS",
        "ts": "1551892375.027300"
    },
    {
        "client_msg_id": "0757c036-ced2-4ba0-a11a-2c7239c579b9",
        "type": "message",
        "text": "Hva jeg har gjort:\nIRMA-cli oppe, testing går. Lagd
         ↪  script til å fore den med NTNU datasettene. Sett at VMware
         ↪  backbone fikser problemene med IRMA ikke ville scanne.\n\nHva
         ↪  jeg planlegger:\nForstå logger fra IRMA scans, prøve å finne
         ↪  en halvoversiktlig måte å bruke dem+se dem\n\nPotensielle
         ↪  hindringer:\nForferdelig view av scanne resultater, hvis man
         ↪  kan kalle det i det hele tatt. Må forbedres\/skrotes og lages
         ↪  bedre",
        "user": "UGMTT84F9",
        "ts": "1551892962.031200"
    },
    {
        "client_msg_id": "7dc5ff9c-509c-403a-bc1c-38b258d7b986",
        "type": "message",
        "text": "Hva jeg har gjort:\nFikk ikke gjort stort mer, men fiksa
        ↪  så irma bruker VMWare, fikk testa packer image generering av
        ↪  windows cuckoo guest.\nHva jeg planlegger:\nSe på scriptet
        ↪  som skal aktivere winRM på windows maskina\nEventuelt sette
        ↪  opp cuckoo guest for linux",
        "user": "UGL9Y07FB",
        "ts": "1551908248.033000"
    }
][
    {
        "client_msg_id": "ac29dc0d-6fa7-4601-a859-319129cf7afe",
        "type": "message",
        "text": "Hva jeg har gjort:\nTatt meg veldig god tid til testing
        ↪  av antivirus, sett på rapport og dokumentasjon (og reddit)
        ↪  mens testingen foregikk. Alle antivirusene untatt 5 gir
        ↪  errors\/blir skipped.\nHva jeg planlegger:\nSkaffe en trial
        ↪  versjon lisensnøkkel for et antivirus som gir error\/skipped
        ↪  for å sjekke hva som skjer da.",
        "user": "UGMNQ83QS",
        "ts": "1552060548.037400"
    },
    {
        "client_msg_id": "4d0b292c-93c5-44d6-be0d-0844d78ec618",
        "type": "message",
        "text": "Hva jeg har gjort (torsdag 07.03)\nVurderte juju4 cuckoo
        ↪  playbook og prøvde å få den til å fungere igjen. Ga opp etter
        ↪  en stund og postet en issue der jeg ba om hjelp. \n\nHva jeg
        ↪  planlegger:\nEnten få det til å fungere eller bytte til annen
        ↪  playbook\n\n\n------\n\nHva har jeg gjort (fredag 08.03):\nGa
        ↪  opp juju4 playbook. Har begynt å teste julianbo.... et eller
        ↪  annet playbook, har flere features \"innebygd\". Har mått
        ↪  tilpasse til ansible 2.7 før jeg innså at det kanskje er
        ↪  greier å bruke en eldre versjon av ansible. Fullførte aldri
        ↪  helt da noen facts ikke er samme i 2.7 som i
        ↪  playbooken.\n\nHva planelgger jeg:\nFullføre og
        ↪  forhåpentligvis ende med at denne er det beste valget for
        ↪  playbook for cuckoo. Legge til vmware role. Teste å sørge for
        ↪  at den fungere ordentlig med irma. Gjøre ytterligere
        ↪  utbedringer å legge til flere features.\n\nPotendielle
        ↪  hindringer:\n\nDenne fungere heller ikke og jeg failer vilt.
        ↪  Må gjøre mer playbook arbeid fra bunn av.",
        "user": "UGQTTS29M",
        "ts": "1552060880.046200"
    }
][
    {
        "client_msg_id": "aa27ad54-f8f0-42d2-aab4-29f2e47d04b5",
        "type": "message",
        "text": "Hva jeg har gjort:\nEndret python scriptet til å sende
        ↪  mindre biter av datasettet til api testing om
        ↪  gangen.\nDiskutert potensielle frontend løsninger der hvor
        ↪  scan results blir mer oversiktlig.\n\nHva jeg planlegger:\nSe
        ↪  på trello om jeg finner noe å gripe tak i. Evt se på Windows
        ↪  packer til Nikolai om han ikke får fikset det.
        ↪  \n\nPotensielle hindringer:\nIRMA er VELDIG tregt til å
        ↪  scanne virker det som. Må testes ordentlig
        ↪  etterhvert.\nIgjen, helt forferdelig gui, og også
        ↪  frustrerende api, må utforskes. ",
        "user": "UGMTT84F9",
        "ts": "1552141486.053800"
    }
][
    {
        "client_msg_id": "e2415a1c-e010-40c6-accf-ac5bbc036173",
        "type": "message",
        "text": "Hva jeg har gjort:\nKort økt, ryddet delvis i trello, så
        ↪  over litt IRMA kode og bittelitt om hvordan implementere en
        ↪  probe.\n\nHva jeg planlegger:\nFørst få igang python script
        ↪  til å sammenligne forventet res vs faktisk res, så begynne å
        ↪  se på loggemuligheter siden det er et stort krav fra
        ↪  kgb\n\nPotensielle hindringer:\nDårlig form, ellers ingenting
        ↪  programvare relatert i dag.",
        "user": "UGMTT84F9",
        "ts": "1552315302.058000"
    },
    {
        "client_msg_id": "9cb968fd-877e-402a-9b50-d3907e73080d",
        "type": "message",
        "text": "Hva har jeg gjort:\nEndelig slåss meg ferdig med den
        ↪  tredje ansible cuckoo playbooken. Denne virker lovende og
        ↪  fungerer ganske bra, har også flere prossecing moduler
        ↪  allerede inkludert f. eks surricata. Fjernet også et par vmer
        ↪  som ikke fungerte somm har kjørt i litt over 4 dager uten å
        ↪  egentlig eksistere. Det var det som laget så mye bråk.\n\nHva
        ↪  jeg planlegger:\nEnten benytte meg av packer image eller evt
        ↪  laget et manuelt fåreløpig samt implementere automatisk
        ↪  håndtering av vmware og image i playbooken sånn at et
        ↪  ordentlig oppsett av cuckoo kan endelig komme til live. Burde
        ↪  også få til en automatisk håndtering av medsendt conf fil for
        ↪  moloch.\n\nPotensielle hindringer:\nProssecing modulen
        ↪  mitmproxy fungerer ikke med python2, kun med python3. Hvert
        ↪  fall den siste versjonen. Foreløpig bare bort kommentert. Så
        ↪  langt virker dette ganske så lovende og burde egentlig
        ↪  fungere relativt greit.",
        "user": "UGQTTS29M",
        "ts": "1552324578.063300",
        "edited": {
            "user": "UGQTTS29M",
            "ts": "1552324640.000000"
        }
    },
    {
        "client_msg_id": "668fb043-5458-4245-bc5e-e35dee43fbbf",
        "type": "message",
        "text": "Hva jeg har gjort:\nFortsetter å jobbe med packer for
         ↪  windows, virker som det er et problem med brannmuren.
         ↪  Problemet med tilkobling av winrm\/ssh vedvarer. Problemet er
         ↪  ikke vmware specific.\nDen klarer ikke sette opp TCP
         ↪  tilkobling til ip:port\nHva jeg planlegger:\nFå til et
         ↪  fungerende oppsett",
        "user": "UGL9Y07FB",
        "ts": "1552331715.064200",
        "edited": {
            "user": "UGL9Y07FB",
            "ts": "1552340006.000000"
        }
    }
][
    {
        "client_msg_id": "e2f7b324-4604-4867-a057-ae1da9a05485",
        "type": "message",
        "text": "Hva har jeg gjort:\nLaget en windows vm manullet som
         ↪  burde fungere.\nHar også begynt å implementere vmware
         ↪  handling i playbooken. Installerer uten problemer. Så langt
         ↪  har jeg ikke funnet et cli som fungerer ordentlig.\n\nHva jeg
         ↪  planlegger:\nFinne en løsning som fungerer og spinne opp
         ↪  cuckoo på kgb og koble til irma.\n\nPotensielle
         ↪  hindringer:\nvmrun ved config av network adapters fungerer
         ↪  ikke. PowerCLI burde være en løsning men virker som om det
         ↪  kan være kronglete å implementere med ansible.",
        "user": "UGQTTS29M",
        "ts": "1552411925.068500"
    },
    {
        "client_msg_id": "0c18d242-3a61-4510-b8d0-2363b3043709",
        "type": "message",
        "text": "Hva jeg har gjort:\nFortsatt å jobbe med packer +
         ↪  windows. Nå ser det ut til at problemet skyldes at maskinen
         ↪  skrur seg av\nHva jeg planlegger:\nFinne ut eksakt hvorfor
         ↪  dette skjer, og hindre det fremover\nPotensielle
         ↪  hindringer:\nLogs er veldig spredd, og dermed også vanskelige
         ↪  å feilsøke",
        "user": "UGL9Y07FB",
        "ts": "1552419318.070100"
    },
    {
        "client_msg_id": "071fe7a3-8d12-41ac-9c00-830fa71e15e7",
        "type": "message",
        "text": "Hva jeg har gjort:\nHar sett på samme Windows
         ↪  Packer-filer som Nikolai, null resultat her heller.\nVet ikke
         ↪  hva konkret feilen er, så vanskelig å fikse det.\nHverken
         ↪  vanlig Makefile eller Makefile.cuckoo ender noe
         ↪  annerledes.\nPotensielt en annen win-iso vil kunne
         ↪  funke?\n\nHva jeg planlegger:\nStange hodet mot dette til
         ↪  sprinten er over om ikke en av oss løser det.\nEllers rolig
         ↪  pause-underholdning med IRMA-APIet og scripte tests
         ↪  der.\n\nPotensielle hindringer:\nTydelig at WinRM ikke er
         ↪  oppe, om det skyldes at win10 skrur seg av eller motsatt, vet
         ↪  jeg ikke.\nSom Nikolai skrev er logs spredd, og viser seg å
         ↪  være utfordrende å få innsyn i VMen på hva som går galt.",
        "user": "UGMTT84F9",
        "ts": "1552423358.076800"
    }
][
    {
        "client_msg_id": "588447f2-f984-4f80-a071-740453972c0a",
        "type": "message",
        "text": "Hva jeg har gjort:\nKommet meg forbi et problem med
         ↪  packer + windows, nådd neste. Nåværende problem er at VMet
         ↪  ikke har nettverkstilgang og at det derfor ikke kan laste ned
         ↪  nødvendig programvare. Fungerer fint i virtualbox,
         ↪  nettverksproblemer i VMWare skyldes konfigurasjonen som
         ↪  gjøres generelt i vmx_data. Virker som VMWare feilen varierer
         ↪  etter miljø (fungerer på egen laptop, men ikke stasjonær)
         ↪  (laptop bruker vmware 14.1 mens stasjonær og kgb bruker
         ↪  15)\nHva jeg planlegger:\nDette problemet tror jeg skal være
         ↪  ganske rett fram, så planen er fortsatt å løse dette og får
         ↪  skvipet til å funke.",
        "user": "UGL9Y07FB",
        "ts": "1552493194.079300",
        "edited": {
            "user": "UGL9Y07FB",
            "ts": "1552512109.000000"
        }
    },
    {
        "client_msg_id": "3dd0b993-3d8e-4b9a-bf39-11a2027e1ccb",
        "type": "message",
        "text": "Hva jeg har gjort:\nFortsatt på implementasjon av vmware
         ↪  handling i playbooken. Nesten ferdig, mangler kun håndtering
         ↪  av nettverk adapterene\nsom jeg tror jeg har funnet løsningen
         ↪  på, med litt hint fra David. REST apiet funker, bare jeg som
         ↪  hadde misforstått litt.\nHar også fått testet cuckoo og hele
         ↪  playbooken i en vm.\n\nHva jeg planlegger å gjøre:\nFullføre
         ↪  implementasjon og få REST apiet til å fungere. Samt teste
         ↪  hele playbooken på nytt på clean image for å sørge for at alt
         ↪  fungere 100%\n\nPotensielle hindringer:\nOppsett av REST api
         ↪  gjennom ansible er vanskelig eller ikke mulig. Eller lett.
         ↪  Hvem vet. Må også finne ut hvorfor\nå ta eierskap gjennom
         ↪  chown ikke fungerer og er nok for vmware, må fortsatt ta
         ↪  eierskap gjennom vmware gui.\nKan ha noe med at vm filer er
         ↪  flyttet, noe som også må fikses gjennom ansible.",
        "user": "UGQTTS29M",
        "ts": "1552498263.079500",
        "edited": {
            "user": "UGQTTS29M",
            "ts": "1552498434.000000"
        }
    }
][
    {
        "client_msg_id": "546480db-e77f-4991-a0e9-702c6770e8d2",
        "type": "message",
        "text": "Hva har jeg gjort:\nSosa mye, prøvd å fått wifi på
        ↪ laptopen til å fungere. Great success, jeg fikk det til.
        ↪ Skjønt hvordan jeg skal få satt opp vmrest gjennom ansible...
        ↪ kopiere conf fil.. Eller tror jeg det blir ish grei skuring å
        ↪ få gjort resten.\n\nHva planlegger jeg:\nFullføre ^. Muligens
        ↪ også implementere et par comfort features som står nevnt på
        ↪ trello. Evt kjøre full test igjen fra clean image?",
        "user": "UGQTTS29M",
        "ts": "1552584088.085300"
    }
][
    {
        "client_msg_id": "13331b2b-3d11-45bb-8d43-f65ba07584d7",
        "type": "message",
        "text": "Hva har jeg gjort:\nFikset cuckoo netverks adapter
        ↪ håndtering. Yay:tada: Trenger fortsatt litt fiksing da.
        ↪ Trenger også bedre håndtering av vm image og sånt, men den
        ↪ fungerer i det minste helt automatisk nå, foruten vm filene
        ↪ da.\n\nHva planlegger jeg:\nFix it ^. Kanskje et par comfort
        ↪ features? mest sannsylig ikke. Kjøre igjennom en full test
        ↪ igjen og sjekke at cuckoo faktisk fungerer som den
        ↪ skal.\n\nHva har jeg lært:\nHusk \" og {{ }} når du holder på
        ↪ med ansible. Me dumdum",
        "user": "UGQTTS29M",
        "ts": "1552672628.088300"
    },
    {
        "client_msg_id": "b45571da-66c1-4e79-9c46-31d5b2151fca",
        "type": "message",
        "text": "Hva jeg har gjort:\nFikset python-test script til å
        ↪ fungere og printe dersom known false positives eller false
        ↪ negatives oppstår.\n\nHva jeg planlegger:\nSe på å sette opp
        ↪ så det ikke blocker og da ikke potensielt stopper alt fra å
        ↪ få kjøre til testen er ferdig.\n\nPotensielle
        ↪ hindringer:\nIRMA virker forferdelig tregt. Ved å kjøre
        ↪ scriptet og bare sende med 2 av hver type fil (4
        ↪ totalt),\nbrukte IRMA 37 sekunder på å returnere resultat.
        ↪ Ved 10 av hver tar det mellom 1 min og 1 min 30.",
        "user": "UGMTT84F9",
        "ts": "1552673243.090100",
        "edited": {
            "user": "UGMTT84F9",
            "ts": "1552673413.000000"
        }
    }
][
    {
        "client_msg_id": "866600e6-c127-4442-8190-a767ce3f69a2",
        "type": "message",
        "text": "Hva jeg har gjort:\nKommet med videre med packer + cukoo
         ↪  + windows skvipet, neste er at winrm sliter med å koble seg
         ↪  til.\nHva jeg planlegger:\nHar fått det til å fungere med ssh
         ↪  i en annen variant, noe som virker en del lettere. Så tenker
         ↪  å konvertere til ssh",
        "user": "UGL9Y07FB",
        "ts": "1552774701.091200",
        "edited": {
            "user": "UGL9Y07FB",
            "ts": "1552774732.000000"
        }
    }
][
    {
        "client_msg_id": "b7384704-d985-4153-a36a-54c1030c0287",
        "type": "message",
        "text": "Hva jeg har gjort:\n14\/03:\nSå på antivirus og hvordan
         ↪  vi kunne legge til flere. Fant en ansible fil der man kunne
         ↪  legge til lisensnøkler for visse antiviruser. Så om det var
         ↪  mulig å skaffe trial versjoner av de nevnt i filen, men ingen
         ↪  av dem kom med en lisensnøkkel.\nI dag:\nJobbet med rapport i
         ↪  dag, sett litt på struktur av andre bacheloroppgaver, lagt
         ↪  til stoff til rapporten på introduksjonskapittelet, mye tatt
         ↪  fra forprosjektet. Har lagt til noe på arbeidsmetoder om
         ↪  verktøy vi har brukt (Trello, Toggl), sikkert mye mer vi kan
         ↪  ha her. Tror at eneste vi mangler på intro kapittelet er
         ↪  øvrige roller og rapportstruktur. Lagt til Kongsbergs krav og
         ↪  kladdet litt på teknologier.",
        "user": "UGMNQ83QS",
        "ts": "1552928052.091600",
        "edited": {
            "user": "UGMNQ83QS",
            "ts": "1552928401.000000"
        }
    },
    {
        "client_msg_id": "1c52e74e-a344-4805-a225-67ff12f6d3e7",
        "type": "message",
        "text": "Hva har jeg gjort: \ntestet cuckoo fra bunn av, ble ikke
         ↪  ferdig for vmware er teit og ansible er, og jeg er
         ↪  teit.\n\nGjøre videre: \nFix it",
        "user": "UGQTTS29M",
        "ts": "1552928540.095100"
    },
    {
        "client_msg_id": "e2c26401-3163-480a-8bf6-0a31b8e3bd1b",
    "type": "message",
    "text": "Hva jeg har gjort:\nSå på rapport oppsettet for
    ↪ forståelsens skyld, og prøvde å aktivere en ekstra antivirus
    ↪ for IRMA (Bit Defender)\n\nHva jeg planlegger:\nSette opp
    ↪ flere evaluation\/trials for antivirus og teste, prøve å leke
    ↪ med tags for å markere om en fil er suspekt, trygg, eller
    ↪ farlig.\n\nPotensielle hindringer:\nDersom APIet og videre
    ↪ funksjonaliteten i IRMA er så begrenset som jeg mistenker, så
    ↪ kan det kreve veldig mye å få implementert noe form for
    ↪ automatisert tagging.",
    "user": "UGMTT84F9",
    "ts": "1552954087.100100"
  }
][
  {
    "client_msg_id": "2fe42ca7-abd2-4e4f-8b8b-055b321befd5",
    "type": "message",
    "text": "Hva har jeg gjort:\nDouble quotes er sykt teit, og jeg
    ↪ har møka med å få ansible til å godta \" i en command i hele
    ↪ dag. Sykt tett. Ikke salty i det hele tatt. Postet på reddit
    ↪ for hjelp, om det ikke er noen løsning blir det å kopiere
    ↪ over hele kommandoen og kjøre lokalt på cuckoo maskinen som
    ↪ et script. Noe som er teit.\n\nHva jeg planlegger:\nFikse
    ↪ det. Så verifisere at ansible playbooken fungerer.
    ↪ Igjen.\n\nPotensielle hindringer:\nAnsible",
    "user": "UGQTTS29M",
    "ts": "1553015127.103300",
    "reactions": [
      {
        "name": "joy",
        "users": [
          "UGL9Y07FB"
        ],
        "count": 1
      }
    ]
  },
  {
    "user": "UGMTT84F9",
    "type": "message",
    "subtype": "channel_purpose",
    "ts": "1553015262.104100",
    "text": "<@UGMTT84F9> set the channel purpose: Legg scrum daily
    ↪ review her:\nHva jeg har gjort\nHva jeg
    ↪ planlegger?\nPotensielle hindringer.\n\nKun daily reviews
    ↪ skal postes her. Alt annet vil bli slettet. Memes, tull,
    ↪ seriøs diskusjon skal til sine respektive channels.",
    "purpose": "Legg scrum daily review her:\nHva jeg har gjort\nHva
    ↪ jeg planlegger?\nPotensielle hindringer.\n\nKun daily reviews
    ↪ skal postes her. Alt annet vil bli slettet. Memes, tull,
    ↪ seriøs diskusjon skal til sine respektive channels.",
    "reactions": [
      {
```

```
                "name": "triumph",
                "users": [
                    "UGL9Y07FB"
                ],
                "count": 1
            }
        ]
    },
    {
        "client_msg_id": "91f27293-eb68-40dc-a243-68847180b7cb",
        "type": "message",
        "text": "Hva jeg har gjort:\ndokumentert eksakt hva som er
         ↪  installert i packer boxes, og sett godt over hva som blir
         ↪  gjort.\nHva jeg planlegger?\nLese fler artikler jeg har
         ↪  funnet om dynamisk analyse for bl.a hvilke software software
         ↪  vi burde ha med eller andre generelle tips\nPotensielle
         ↪  hindringer:\nPacker problemet jeg feilsøker på sida er
         ↪  irriterende, ellers går det nok fint",
        "user": "UGL9Y07FB",
        "ts": "1553041076.105700",
        "edited": {
            "user": "UGL9Y07FB",
            "ts": "1553041174.000000"
        }
    }
][
    {
        "client_msg_id": "ee66603e-5894-498c-afce-a0f5fc92ba55",
        "type": "message",
        "text": "Hva jeg har gjort:\nSkrevet om en hel setning i
         ↪  rapporten, gjort backup av et helt script.\n\nHva jeg
         ↪  planlegger:\nHar ikke snøring atm\n\nPotensielle
         ↪  hindringer:\nPersonlig helse :sunglasses:",
        "user": "UGMTT84F9",
        "ts": "1553103972.107400",
        "edited": {
            "user": "UGMTT84F9",
            "ts": "1553103992.000000"
        }
    },
    {
        "client_msg_id": "9f365af0-1eef-4ba3-986a-e47aa3b37765",
        "type": "message",
        "text": "Hva har jeg gjort:\nPrøvd å få start vm til å fungere,
         ↪  ikke fått det til. Tror jeg må få lit feedback og ideer fra
         ↪  dere eller noe andre. Er kind of lost.\n\nHva jeg
         ↪  planlegger:\nFix\n\nPotensielle hindringer:\nsfljsf, teit.
         ↪  møk",
        "user": "UGQTTS29M",
        "ts": "1553104804.109100"
    }
][
    {
        "client_msg_id": "dec1d867-235f-4c33-90b8-fc8461e75576",
        "type": "message",
        "text": "Hva jeg har gjort:\nFikset Bitdefender failures\nLagt
        ↪  til litt mer på teknologidelen av rapporten\nPrøvd å legge
        ↪  til flere antivirus\n\nHva jeg planlegger:\nSe mer på
        ↪  antivirus",
        "user": "UGMNQ83QS",
        "ts": "1553184496.110200",
        "reactions": [
            {
                "name": "+1",
                "users": [
                    "UGMTT84F9"
                ],
                "count": 1
            }
        ]
    }
][
    {
        "client_msg_id": "b388f3c5-63ce-42fc-bbda-6e43f1b425dd",
        "type": "message",
        "text": "Hva jeg har gjort:\nBegynt design av kiosk
        ↪  m\/interaksjon med resten av systemet.\nWrapper rundt api
        ↪  design-tanker, kan kontrollere statisk+dynamisk analyse flow,
        ↪  pluss sette tags på hensiktsmessig måte\n\nHva jeg
        ↪  planlegger:\nSe mer på designet, gå mer low level i
        ↪  kiosk-krav.\n\nPotensielle hindringer:\nHvordan få til en
        ↪  hensiktmessig flow fra disk settes inn til resultat er klart,
        ↪  og hvordan håndtere det så bruker får nytte av det.\n-
        ↪  Systemet skal være isolert; hvor isolert -&gt; Skal det være
        ↪  exit points for scannede filer, iSåfall hvor skal det så
        ↪  være?\n- Hvis legge opp til mulig AD: ca hvordan er
        ↪  AD-strukturen til kgb -&gt; hva slags identifiers er
        ↪  nødvendig for å gjøre filoverføring av trygge filer
        ↪  brukervennlig?",
        "user": "UGMTT84F9",
        "ts": "1553274767.118700"
    },
    {
        "client_msg_id": "537c47e9-a827-4c50-81d8-289172ddfd82",
        "type": "message",
        "text": "Hva har jeg \ntalked'n stuff, planned'n stuff. Vmware er
        ↪  teit.\n\nhva jeg planlegger:\nskrive rapport.
        ↪  Vmware.\n\nPotesielle hindringer:\nVmware",
        "user": "UGQTTS29M",
        "ts": "1553305009.119700"
    }
][
    {
        "client_msg_id": "d0343a2b-4df0-4c0f-91b8-317e37af53e4",
        "type": "message",
        "text": "Fredag 22\/3:\nHva jeg gjorde:\nLa til F-prot
        ↪ antivirus\nPrøvde å legge til Sophos men var utdaterte
        ↪ instrukser på IRMA og Sophos var teit",
        "user": "UGMNQ83QS",
        "ts": "1553501881.121000"
    }
][
    {
        "client_msg_id": "1c8307f4-fd61-4110-ac97-baf4329b99c1",
        "type": "message",
        "text": "Fredag 22\/3\nHva jeg har gjort:\nMasse snakking og litt
        ↪ rapport",
        "user": "UGQTTS29M",
        "ts": "1553613556.125900"
    },
    {
        "client_msg_id": "b79a71c3-9c2a-4479-905f-f94c7a7fff1e",
        "type": "message",
        "text": "Hva jeg har gjort:\nAktivert ClamAV, comodo, LIEF, PEID,
        ↪ StaticAnalyzer og Virustotal med god hjelp fra Martin der jeg
        ↪ fikk error. Prøvd å aktivere Eset, NSRL og AVG. Lagt til et
        ↪ avsnitt på rapporten.\n\nHva jeg planlegger:\nHar gått
        ↪ gjennom alle antivirusene IRMA har dokumentasjon på. Vil
        ↪ fortsette å gå gjennom de som IRMA støtter men ikke har
        ↪ dokumentasjon på. Føler meg ferdig med Vagrant på rapporten
        ↪ så går over til å skrive om antivirus snart.\n\nPotensielle
        ↪ hindringer:\nSom skrevet over så har IRMA en del flere
        ↪ antivirus de støtter men som de har ingen dokumentasjon på,
        ↪ som kan bli vanskelig å aktivere. I tillegg gir clamAV error
        ↪ på hver eneste scan hittil og PEID, LIEF, staticanalyzer
        ↪ vises som aktivert på IRMA men gjør ingen scanning.
        ↪ Bitdefender har også plutselig begynt å gi errors.  Verdt å
        ↪ se på senere.",
        "user": "UGMNQ83QS",
        "ts": "1553613772.131400"
    },
    {
        "client_msg_id": "97c6e672-ecad-4b59-91aa-c1da29dffff2",
        "type": "message",
        "text": "Hva jeg har gjort:\nPrøvde ny måte å løse VM start på.
        ↪ Fungerte ikke. Har tenkt på potensiell ny løsning. Om det
        ↪ ikke fungerer tror jeg gir opp, for vmware workstation.
        ↪ Muligens fikse for esx da det tilslutt blir brukt. \nHoldt på
        ↪ med å legge inn AV sammen med Olav. Skrev noe på
        ↪ rapport\n\nVidere: \nFortsette cuckoo, fortsette rapport.",
        "user": "UGQTTS29M",
        "ts": "1553613793.132000"
    }
][
    {
        "client_msg_id": "15c170c6-a4eb-4bb8-ac48-1faad63a4837",
        "type": "message",
        "text": "Hva jeg har gjort:\nFunnet en løsning på git sub-module
         ↪  problemet. En alternativ løsning er å fjerne alle spor av
         ↪  sub-modules og bare legge til den ene fila manuelt i
         ↪  repoet.\nStartet smått på rapportseksjonen IRMA.\n\nHva jeg
         ↪  planlegger:\nVisualisere hvordan IRMA seksjonen i rapporten
         ↪  bør bygges opp.\nPrøve å tenke ut en måte og samspille
         ↪  rapporten til en minimumsgrad så det blir mindre å skrive om
         ↪  senere+mindre unødvendig skriving fra alle for å dekke andres
         ↪  stoff.\n\nPotensielle hindringer:\nLite coherency mellom
         ↪  gruppemedlemmer så mye informasjon i rapporten fort blir
         ↪  gjengitt redundantly i flere seksjoner på rad (mye plass til
         ↪  å beskrive IRMA var både under IRMA og Cuckoo som var neste
         ↪  seksjon. Rotete.)",
        "user": "UGMTT84F9",
        "ts": "1553708380.137500"
    },
    {
        "client_msg_id": "d66aec4e-b4ef-4fdf-8eaa-cba489767a9d",
        "type": "message",
        "text": "Hva har jeg gjort:\nGitt opp autostart av vm med vmware.
         ↪  Testet cuckoo uten autostart. Liten bug med lasting av vm når
         ↪  cuckoo kjører, tester nå burde være i orden nå snart.\n\nHva
         ↪  jeg planlegger:\nVerifisere at cuckoo fungerer. Skriver
         ↪  ferdig cuckoo delen i rapporten + muligens litt
         ↪  virtualisering. Så muligens se på ESX?\n\nPotensielle
         ↪  hindringer:\nCuckoo vm issue var mer enn liten bug?",
        "user": "UGQTTS29M",
        "ts": "1553708638.140100"
    }
][
    {
        "client_msg_id": "25151eec-cfba-478f-8829-fa4e7cd27851",
        "type": "message",
        "text": "Hva jeg har gjort:\nDrevet med mer AV testing i går og i
         ↪  dag. Tenker jeg gir meg med å prøve å aktivere flere. Er
         ↪  4(Zoner, Virusblokada, McAfeeVSCL, Kaspersky\t) til som ikke
         ↪  er prøvd aktivert men nettsidene deres var vanskelig å
         ↪  navigere og fant ikke en linux versjon. 11 AVer er velykket
         ↪  aktivert i IRMA men noen er veldig ustabile og er \"skrudd
         ↪  av\" per nå. Noen ganger virker de, andre ganger ikke. Noen
         ↪  som er aktivert gir ofte error state i IRMA, spesielt
         ↪  Bitdefender og Comodo.\n\nHva jeg planlegger:\nIkke helt
         ↪  sikker hva jeg skal gjøre praktisk akkurat nå. På rapporten
         ↪  er det mer som kan skrives og dokumentasjon kan muligens
         ↪  utbedres.\n\nPotensielle hindringer:\nUstabile AV-er, ustabil
         ↪  IRMA.",
        "user": "UGMNQ83QS",
        "ts": "1553783788.004600"
    },
    {
        "client_msg_id": "27c8b45f-869e-4025-8ea6-98bf0e710f36",
        "type": "message",
        "text": "Hva har jeg gjort:\nSkrevet mer i rapporten. Mangler
         ↪  fortsatt en del. Fått testet at cuckoo fungerer som den skal.
         ↪  Kjører fortsatt playbook siden jeg glemte å gi mer enn 512
         ↪  mb.\n\nHva planlegger jeg:\nSkrive rapport. Få cuckoo
         ↪  ordentlig inn i IRMA. Streamlinet håndtering av cuckoo i
         ↪  playbooken og evt vurdere hva som kan fjernes i vårt
         ↪  tilfelle.\n\nPotensielle hindringer:\nPlaybooken trenger mere
         ↪  fixes.",
        "user": "UGQTTS29M",
        "ts": "1553808283.006900"
    }
][
    {
        "client_msg_id": "70342846-1314-47a2-9962-259251e2bdca",
        "type": "message",
        "text": "Hva jeg har gjort:\nSkrevet ish litt over en halv side
         ↪  om antivirus.\n\nHva jeg planlegger:\nHar ikke skrevet noe
         ↪  spesifikt om antivirusene vi har nå eller implementasjon av
         ↪  dem.\n\nPotensielle hindringer:\nSnart helg",
        "user": "UGMNQ83QS",
        "ts": "1553862098.008300",
        "edited": {
            "user": "UGMNQ83QS",
            "ts": "1553862129.000000"
        }
    },
    {
        "client_msg_id": "bfdfa7e0-31e4-4c59-b2da-d8272517788a",
        "type": "message",
        "text": "Hva jeg har gjort:\nSkrevet om IRMA seksjon til å
         ↪  omhandle Malware Analyse Platform teknologier, begynt å
         ↪  diskutere OPSWAT vs IRMA vs VirusTotal der og utdype mer om
         ↪  alternativ+krav\n\nHva jeg planlegger:\nImportere gamle
         ↪  notater videre\nFortsette kiosk+nødvendige infrastruktur
         ↪  tanker videre\n\nPotensielle hindringer:\nPersonlige
         ↪  problemer, lite klarhet i hvordan rapporten konkret bør se ut
         ↪  (bør være intuitivt nok til å ikke bli altfor stort
         ↪  problem)",
        "user": "UGMTT84F9",
        "ts": "1553882626.012300"
    }
][
    {
        "client_msg_id": "20470966-a842-4865-8dac-7c1c30f8da40",
        "type": "message",
        "text": "Hva jeg har gjort:\nSkrevet ned hvordan man enabler
         ↪  antivirusene vi har fått til hittil i rapporten, men muligens
         ↪  overflødig hvis vi legger alt i ansible script? Også tatt en
         ↪  tcpdump mens en scan foregår.\n\nHva jeg
         ↪  planlegger:\nUsikker\n\nPotensielle hindringer:\nOgså litt
         ↪  usikker på rapportstruktur, har lagt til det jeg skrev i
         ↪  implementation seksjonen av rapporten.",
        "user": "UGMNQ83QS",
        "ts": "1554125374.014500",
        "thread_ts": "1554125374.014500",
        "reply_count": 1,
        "reply_users_count": 1,
        "latest_reply": "1554152922.016200",
        "reply_users": [
            "UGL9Y07FB"
        ],
        "replies": [
            {
                "user": "UGL9Y07FB",
                "ts": "1554152922.016200"
            }
        ]
    },
    {
        "client_msg_id": "86bba503-09bc-4e71-9c66-5e8c585a36b3",
        "type": "message",
        "text": "Hva har jeg gjort:\nFå cuckoo til å fungere.\n\nHva
         ↪  planlegger jeg:\nFikse på playbook sånn at så mye som mulig
         ↪  går av seg selv. Så evt fikse på configer også sånt. Evt
         ↪  vurdere hva som er overflødig og kan fjernes fra
         ↪  sluttprodukt. Skrive mer i rapporten.",
        "user": "UGQTTS29M",
        "ts": "1554127625.016100"
    },
    {
        "client_msg_id": "28279047-9aa6-4c84-af02-e04074743dd7",
        "type": "message",
        "text": "Veldig greit å ha gjort det manuelt før man skriver det
         ↪  om til ansible :smile:",
        "user": "UGL9Y07FB",
        "ts": "1554152922.016200",
        "thread_ts": "1554125374.014500",
        "parent_user_id": "UGMNQ83QS"
    }
][
    {
        "client_msg_id": "e0f4abda-b2db-4829-a846-3e467a4dcd5f",
        "type": "message",
        "text": "Hva jeg har gjort:\nBegynt med automatisering av AV-er.
         ↪  Lagt til peid, static analyzer og lief. Ble akkurat ferdig
         ↪  med AVG, Avast og Comodo men fikk ikke testet før vi måtte
         ↪  dra.\n\nHva jeg planlegger:\nFortsette med ansible scriptet.
         ↪  Forbedre det jeg har skrevet i rapporten.\n\nPotensielle
         ↪  hindringer:\nHar ikke fått lagt til Trid og skjønner ikke
         ↪  hvordan akkurat nå. Jeg trodde det skulle gjøres på samme
         ↪  måte som peid, static analyzer og lief men det går tydeligvis
         ↪  ikke? Har vært lagt til i Irma før...",
        "user": "UGMNQ83QS",
        "ts": "1554221052.019200"
    },
    {
        "client_msg_id": "441ce02e-ea31-4c25-8a60-11c541ff6af4",
        "type": "message",
        "text": "Hva har jeg gjort:\nCuckoo\n\nHva planlegger jeg:\nSamme
         ↪  som forrige",
        "user": "UGQTTS29M",
        "ts": "1554226102.020000"
    }
][
    {
        "client_msg_id": "b0147612-96d2-4358-a0a2-c9dfd57abea7",
        "type": "message",
        "text": "Hva jeg har gjort:\nFått fikset problemene til AVG,
         ↪  Avast og Comodo. Men har sletet med ClamAV resten av
         ↪  tiden.\n\nHva jeg planlegger:\nFortsette med scriptet",
        "user": "UGMNQ83QS",
        "ts": "1554387073.021000"
    },
    {
        "client_msg_id": "b396a06c-3cf1-4fef-96a3-063e7d7afb29",
        "type": "message",
        "text": "Hva jeg har gjort:\nJobbet med pipeline, sett på
         ↪  muligheter. F.eks ved å bruke
         ↪  <https:\/\/github.com\/airbus-seclab\/rebus> (har en
         ↪  fungerende provisjoneringsløsning om denne er best). Et
         ↪  alternativ er å ha et eget api entrypoint som pipeliner scans
         ↪  ved å kommunisere med API'en lokalt. Funnet et par løsninger
         ↪  for å raskere sette opp irma, og potensielt noen for raskere
         ↪  skans. Delvis planlagt designet for pipeline i tillegg.\nLagt
         ↪  til Yara støtte og skiftet over til prod i nye irma repo\nHva
         ↪  jeg planlegger:\n IDAG: Fikse windows winrm problemer
         ↪  ettersom det automatisk brukes av ansible og eventuelle
         ↪  nettverksproblemer som kommer av at vi bruker vmware\n
         ↪  SENERE: Diskutere pipeline med grp, og teste
         ↪  effektivitetsløsninger. Oppdatere Kongsberg om fremgang",
        "user": "UGL9Y07FB",
        "ts": "1554393974.025300",
        "edited": {
            "user": "UGL9Y07FB",
            "ts": "1554394606.000000"
        },
        "attachments": [
            {
                "service_name": "GitHub",
                "title": "airbus-seclab\/rebus",
                "title_link":
                 ↪  "https:\/\/github.com\/airbus-seclab\/rebus",
                "text": "REbus facilitates the coupling of existing tools
                 ↪  that perform specific tasks, where one's output will
                 ↪  be used as the input of others. -
                 ↪  airbus-seclab\/rebus",
                "fallback": "GitHub: airbus-seclab\/rebus",
                "from_url": "https:\/\/github.com\/airbus-seclab\/rebus",
                "thumb_url":
                 ↪  "https:\/\/avatars0.githubusercontent.com\/u\/25298315?s=400&v=4",
                "thumb_width": 250,
                "thumb_height": 250,
                "service_icon":
                 ↪  "https:\/\/a.slack-edge.com\/bfaba\/img\/unfurl_icons\/github.png",
                "id": 1,
                "original_url":
                 ↪  "https:\/\/github.com\/airbus-seclab\/rebus"
            }
        ]
    },
    {
        "client_msg_id": "eb51f3be-20df-4501-84c9-24171495edd3",
        "type": "message",
        "text": "Hva har jeg gjort:\nFikset ip problem med vm som er fra
         ↪  samme image. Sjekket at playbook fungerer som planlagt på
         ↪  nytt. Det gjorde den. Trenger litt kjærlighet på noe config
         ↪  men ellers greit.\n\nHva planlegger jeg:\nSkrive i rapporten
         ↪  og fikse dokumentasjon + README til å reflektere endringer og
         ↪  generelt oppdatere den. Muligens også utbedre noe
         ↪  funksjonalitet.",
        "user": "UGQTTS29M",
        "ts": "1554419518.027100"
    }
][
    {
        "client_msg_id": "1e9d314d-56a1-42ff-b388-bfa11f4c73e4",
        "type": "message",
        "text": "Hva har jeg gjort:\nLagt inn feil ip i vmen igjen?
         ↪  Skrevet om Readme og installasjonsguide i git repo. Satt opp
         ↪  cuckoo igjen. Nå skal den være mer eller mindre i
         ↪  orden.\n\nHva planlegger jeg:\nSkrive ferdig doku. Skrive mer
         ↪  i rapporten. Legge til ESX støtte.\n\nPotensielle
         ↪  hindringer:\nEsx er værre en workstation.",
        "user": "UGQTTS29M",
        "ts": "1554476794.031900",
        "reactions": [
            {
                "name": "cry",
                "users": [
                    "UGMTT84F9"
                ],
                "count": 1
            },
            {
                "name": "+1",
                "users": [
                    "UGL9Y07FB"
                ],
                "count": 1
            }
        ]
    },
    {
        "client_msg_id": "cca37045-803b-499c-bdb5-27002b4788fa",
        "type": "message",
        "text": "Hva jeg har gjort:\nKioskløsning, funnet noen konkrete
         ↪  alternativer til å oppdage tilkoblede enheter i
         ↪  kiosken\nEnder antageligvis opp med listener for å finne
         ↪  \"ACTION\"=\"add\" og uevents\nSatt opp igjen så irma faktisk
         ↪  starter lokalt til bruk for api-testing+api-docs\n\nHva jeg
         ↪  planlegger:\nImplementere kiosk demo\n\nPotensielle
         ↪  hindringer:\nVi har redigert default environment-filer
         ↪  (prod.yml referer til vmware istedenfor for å bare lage
         ↪  vmware_prod.yml til samme formål)\nDette er fantastisk
         ↪  irriterende når man kaster bort tid på å prøve og sette opp
         ↪  default env og det ikke funker.\nVet ca hva <@UGL9Y07FB>
         ↪  planlegger, men aner ikke hvordan det blir utført. Føler jeg
         ↪  sitter i tåka på hvordan connections mellom kiosk og irma
         ↪  skal kodes.",
        "user": "UGMTT84F9",
        "ts": "1554484332.039500",
        "edited": {
            "user": "UGMTT84F9",
            "ts": "1554484405.000000"
        },
        "thread_ts": "1554484332.039500",
        "reply_count": 1,
        "reply_users_count": 1,
        "latest_reply": "1554485861.039900",
        "reply_users": [
            "UGL9Y07FB"
        ],
        "replies": [
            {
                "user": "UGL9Y07FB",
                "ts": "1554485861.039900"
            }
        ]
    },
    {
        "client_msg_id": "520276e4-a86b-42f0-8db5-16d9613964a6",
        "type": "message",
        "text": "Per nå brukes orginal prod slik at vi har fungerende
         ↪  windows maskin for avs. I første omgang holder det med at
         ↪  kiosken starter en skann hos IRMA i likheten av det du har
         ↪  kodet før for å teste irma. Etter det er gjort kan man utvide
         ↪  ved å legge til tags på filer tilhørende den returnerte skann
         ↪  id'en",
        "user": "UGL9Y07FB",
        "ts": "1554485861.039900",
        "edited": {
            "user": "UGL9Y07FB",
            "ts": "1554485916.000000"
        },
        "thread_ts": "1554484332.039500",
        "parent_user_id": "UGMTT84F9"
    }
][
    {
        "client_msg_id": "6624afce-3a83-4ae6-a691-4d30f08f4d28",
        "type": "message",
        "text": "Hva har jeg gjort:\nFikset \"ferdig\" ansible playbooken
        ↪   for cuckoo og pushet til github\n\nHva jeg
        ↪   planlegger:\nSkrive mer om det i rapporten. Legge til
        ↪   esx.\n\nPotensielle hindringer:\nEsx er verre en
        ↪   workstation",
        "user": "UGQTTS29M",
        "ts": "1554746114.041700"
    },
    {
        "client_msg_id": "6882b7c7-d6a8-4d9a-a8c3-0aa3d41088aa",
        "type": "message",
        "text": "Hva jeg har gjort:\nGjort klart windows på hjemmepc til
        ↪   å jobbe i framtida. Ellers opptatt med ikke-skolefaglige ting
        ↪   i dag.\n\nHva jeg planlegger:\nFortsette arbeid fra fredag på
        ↪   skolen.\n\nPotensielle hindringer:",
        "user": "UGMTT84F9",
        "ts": "1554752932.043900"
    }
][
    {
        "client_msg_id": "3809e065-dfb4-49e4-aedf-564262db006a",
        "type": "message",
        "text": "Hva jeg har gjort:\nI går og i dag. Fikset clamAV
        ↪   problemet men fikk problemer med de AV-ene som henter ting
        ↪   fra nettet. Har ikke klart å fikse de men scriptet er i
        ↪   prinsippet fungerende.\n\nHva jeg planlegger:\nBegynne å
        ↪   debugge probes, se på rapport og kanskje windows AV-er.",
        "user": "UGMNQ83QS",
        "ts": "1554816704.045600"
    },
    {
        "client_msg_id": "0ae60fb4-e1e0-4fe5-be94-77b86502dbeb",
        "type": "message",
        "text": "Hva har jeg gjort:\nPrøvd å innstallere esx, fikk
        ↪   installert gratis versjonen. Trenger ikke gratis versjonen.
        ↪   Prøvde det, fikk ikke prøve lisens. Sia ga med beskjed om at
        ↪   den ikke fantes...\n\nHva planlegger jeg:\nPrøve igjen
        ↪   da..\n\nPotensielle hindringer:\nvmware",
        "user": "UGQTTS29M",
        "ts": "1554829192.047200",
        "reactions": [
            {
                "name": "heart",
                "users": [
                    "UGMTT84F9"
                ],
                "count": 1
            }
        ]
    },
    {
        "client_msg_id": "f26fa495-f897-4dd7-a5a2-928fa9dcc9ed",
        "type": "message",
        "text": "Hva jeg har gjort:\nFunnet en teori om at livet ville
         ↪  vært enklere med ritalin eller lignende\n\nHva jeg
         ↪  planlegger:\nHva som helst at this point\n\nPotensielle
         ↪  problemer:",
        "user": "UGMTT84F9",
        "ts": "1554837784.050100"
    }
][
    {
        "client_msg_id": "2c2aaf8a-8a92-45d6-aa11-c547432c9bc5",
        "type": "message",
        "text": "*Hva jeg har gjort:*\nTesta metoder for å gjøre vagrant
         ↪  provisjonering fortere, og funnet flere.\nBegynt på NSRL
         ↪  møter på problemet når jeg skal \"bygge\" dbene (tror det
         ↪  bare vil si at jeg skriver om formatet på fila til
         ↪  key=value)\nPlanlagt og skrevet litt for rapporten, og
         ↪  utforsket multiscanner. Jobbet med metodikk for pipeline, må
         ↪  først teste ut API'en for å finne ut mer eksakt hvordan den
         ↪  funker\n*Hva jeg planlegger:*\nFikse NSRL db bygging, er i
         ↪  kommunikasjon med i IRC for dette + videre feilsøking og
         ↪  forståelse av kode.\nTeste IRMA API  og begynne med kode for
         ↪  pipeline (bl.a se på hva som returneres i hvert stage av
         ↪  pipeline for å planlegge logikken)\nFå sendt en demo\nTeste
         ↪  parallellisering av IRMA scans og skrive mer
         ↪  notater\n*Potensielle problemer:*\nKan oppstå vanskeligheter
         ↪  med å få løst NSRL problemet, koden er såpass komplisert så
         ↪  forståelse er tidkrevende",
        "user": "UGL9Y07FB",
        "ts": "1554936171.050300",
        "edited": {
            "user": "UGL9Y07FB",
            "ts": "1554936624.000000"
        }
    }
][
    {
        "client_msg_id": "66423e9b-653e-4e55-83da-3eea5a3378b6",
        "type": "message",
        "text": "Hva jeg har gjort:\nSett på om noen AV-er trengs å bli
         ↪  debugget. Bitdefender brukte å faile ofte før men etter 300+
         ↪  scans har den bare failet 2 ganger. Har prøvd å gjøre debug
         ↪  utifra den her:
         ↪  <https:\/\/irma.readthedocs.io\/en\/latest\/troubleshooting\/debug.html>.
         ↪  Og ifølge den var det ikke noe i veien med Bitdefender, la ut
         ↪  output i google drive under docs. Ingen ande AV-er har gitt
         ↪  errors, men ser ut som mcAffe aldri reagerer på noe.",
        "user": "UGMNQ83QS",
        "ts": "1554993838.053600",
    "reactions": [
      {
        "name": "heart",
        "users": [
          "UGL9Y07FB",
          "UGMTT84F9"
        ],
        "count": 2
      }
    ]
  },
  {
    "client_msg_id": "3098f03d-4961-42c4-8850-afa400d10ac3",
    "type": "message",
    "text": "Hva har jeg gjort:\nEndelig fått til å sette noen
    ↪ skeleton filer til livet for kiosken.\nTygd gjennom en del
    ↪ dokumentasjon på hvordan man kan parse events fra kernel i
    ↪ python;\nSlik kan man oppdage nye enheter plugget inn.\n\nHva
    ↪ jeg planlegger:\nFå istand enkel uevent lesing for å dynamisk
    ↪ oppdage nye enheter.\n\nPotensielle
    ↪ problemer:\nDokumentasjonen er ikke særlig selvforklarende og
    ↪ vanskelig å forstå seg på.",
    "user": "UGMTT84F9",
    "ts": "1555018625.058400"
  },
  {
    "client_msg_id": "56560c03-066e-4fdc-87e6-afef5b1531f4",
    "type": "message",
    "text": "*Hva jeg har gjort:*\nKommet videre med NSRL, metoden
    ↪ for å bygge bygger på utdatert et utdatert bibliotek (støttes
    ↪ ikke lenger i python3, som venv'et kjøres i)\nSkrevet for
    ↪ rapport\n*Hva jeg planlegger:*\nFikse NSRL db bygging +
    ↪ videre feilsøking og forståelse av kode. Eventuelt finne en
    ↪ løsning for å automatisk sette opp venvet i python2 eller se
    ↪ på andre alternativer for dette\nTeste IRMA API  og begynne
    ↪ med kode for pipeline (bl.a se på hva som returneres i hvert
    ↪ stage av pipeline for å planlegge logikken)\nFå sendt en
    ↪ demo\nTeste parallellisering av IRMA scans og skrive mer
    ↪ notater\n*Potensielle problemer:*\nKan oppstå vanskeligheter
    ↪ med å få løst NSRL problemet, mye utdatert",
    "user": "UGL9Y07FB",
    "ts": "1555028866.058700",
    "edited": {
      "user": "UGL9Y07FB",
      "ts": "1555029002.000000"
    }
  }
][
  {
    "client_msg_id": "e715d2d6-0378-4f27-a4dc-b6f2dfa7c5a3",
    "type": "message",
        "text": "*Hva jeg har gjort:*\nNSRL er ca ferdig, burde forbedre
         ↪  noe kode, men vil teste med IRMA først\nFunnet ut hvor jeg
         ↪  kan slenge inn et kall mot pipelinen i IRMA koden\n*Hva jeg
         ↪  planlegger:*\nTeste og evt fikse resterende av NSRL og
         ↪  begynne å skrive kode for pipeline\nFå sendt en
         ↪  demo\n*Potensielle problemer:*",
        "user": "UGL9Y07FB",
        "ts": "1555270828.001600"
    }
][
    {
        "client_msg_id": "a61899ca-d1d5-448b-9f12-93e78339f984",
        "type": "message",
        "text": "Hva jeg har gjort:\nNye ledetråder funnet for udev og
         ↪  tilhørende python wrapper-bibliotek, so thats nice\nForstår
         ↪  pyudev wrapperen, mangler udev bibliotek kjennskap\n\nHva jeg
         ↪  planlegger:\nDypdykk i udev biblioteket for kommunikasjon
         ↪  mellom kernel og user space\n\nPotensielle
         ↪  hindringer:\nKomplisert dokumentasjon å lese gjennom og
         ↪  forstå",
        "user": "UGMTT84F9",
        "ts": "1555630291.003100"
    }
][
    {
        "client_msg_id": "af47ce2f-81a3-485e-adf8-1368608401dc",
        "type": "message",
        "text": "Hva jeg har gjort:\nSett på windows AV provisjonering og
         ↪  implementert 1. IRMA støtter flere men da må vi lage rollene
         ↪  selv. Begynt å implementere AV scriptet mitt som flere roller
         ↪  men har ikke kommet så langt der ennå for har fokusert på en
         ↪  type error.\n\nHva jeg planlegger:\nFortsette med å lage
         ↪  flere roller både får linux og windows.",
        "user": "UGMNQ83QS",
        "ts": "1555851558.002400",
        "edited": {
            "user": "UGMNQ83QS",
            "ts": "1555852032.000000"
        }
    }
][
    {
        "client_msg_id": "a8e7c58f-3dd4-4b89-b65e-910549ecaaef",
        "type": "message",
        "text": "Hva jeg har gjort:\nBlitt ferdig med linux av roller, de
         ↪  gir ihvertfall ingen errors nå men skulle gjerne gjort en
         ↪  restart av IRMA for å faktisk se om de virker men det kan tas
         ↪  senere. Fant ut i dag at noe av det originale av scriptet
         ↪  mitt var overflødig for det hadde allerede blitt gjort i noen
         ↪  av rollene :disappointed: Men var fortsatt noe jeg kunne
         ↪  legge til. Begynte å lage en oversikt over hvilke av-er som
         ↪  ikke virket og prøvde å sjekke om noen hadde noen raske
         ↪  fixes, fikk lagt til et par flere samtidig.\n\nHva jeg
         ↪  planlegger:\nPrøve å lage noen windows
         ↪  av-roller\n\nPotensielle hindringer:\nTar ish 5 min å sjekke
         ↪  om en spesifikk rolle fungerer så tar fort en del tid når du
         ↪  gjør flere syntax feil på rad xD",
        "user": "UGMNQ83QS",
        "ts": "1556027806.007200"
    }
][
    {
        "client_msg_id": "061d6351-5957-42b7-88db-828923d4b45e",
        "type": "message",
        "text": "Hva jeg har gjort:\nFunnet ut hvordan håndtere udev til
         ↪  å finne ut når en device settes inn i maskin, og noen
         ↪  potensielle attributes som ser ut til kunne fungere til å
         ↪  identifisere alle normale filsystem og formaterte
         ↪  drives.\n(ID_FS_TYPE og subsystem=\"block\")\n\nHva jeg
         ↪  planlegger:\nImplementere koden som skal ligge over det low
         ↪  level nivået; mount drives, sende til ekstern maskin (IRMA),
         ↪  \n\"long term\": kreve autentisering for å sende til scan
         ↪  (Active Directory login?).\n\nPotensielle problemer:\nUSB
         ↪  attacks og hvordan ikke motarbeide countermeasures til det i
         ↪  kodingen.\nAutentisering er viktig, men også det jeg antar
         ↪  blir mest problematisk å implementere.\nFortsatt ikke helt
         ↪  sikker på hvordan overføre hele disker, men tror det går
         ↪  greit med http(s) kopiering av mounted drives rett og
         ↪  slett.\nAccountance buddies er ubrukelig..",
        "user": "UGMTT84F9",
        "ts": "1556144247.022400"
    }
][
    {
        "client_msg_id": "520a6475-e6a7-4da0-86f0-13a841bebeb2",
        "type": "message",
        "text": "Hva jeg har gjort:\nKodet delen for å plukke opp uevents
         ↪  for block-devices og hente FS_TYPE og DEVNAME til mounting
         ↪  senere.\nFunnet en (forhåpentligvis) dynamisk kodesnutt som
         ↪  kan brukes til å mounte nye devices for sending til
         ↪  IRMA.\n99% bestemt meg for å bruke rsync for å overføre
         ↪  filene.\n\nHva jeg planlegger:\nTeste at de nåværende
         ↪  funskjonene nå fungerer i det hele tatt.\nTilpasse
         ↪  IRMA-frontend scan wrapperen fra tidligere.\nReise til gjøvik
         ↪  for å finne design skissen min for å se hva jeg har
         ↪  glemt.\n\"Long term\": Implementere enkel try-catch error
         ↪  handling.\n\nPotensielle problemer:\nAner ikke hvordan jeg
         ↪  skal kunne skrive noe automatiserte tester for casene her, så
         ↪  må gjøres manuelt.\nActive Directory authorisation uheldigvis
         ↪  utenfor midlertidig scope.\nUsikker på hvor filoverføringen
         ↪  bør ende og håndteres;\n- TCP port på kgb, sende filene til
         ↪  \/tmp\/&lt;dir&gt; der, så videre til IRMA?\n- Rett til IRMA
         ↪  frontend og håpe alt går slik det skal der?\n- Annet?",
        "user": "UGMTT84F9",
        "ts": "1556316672.012900"
    }
][
    {
        "client_msg_id": "f4406b59-3e37-4a57-8c6e-36b5d7e0946e",
        "type": "message",
        "text": "Hva jeg har gjort:\nTestet kode, refaktorert kode, addet
         ↪  ny kode, totalt endret mount funksjonen. Nå funker den, men
         ↪  error handling er fraværende i forhold til clib
         ↪  approachen.\n\nHva jeg planlegger:\nMøte i morgen og
         ↪  diskutere vegen videre.\n\"Autorisering\"\nImplementere rsync
         ↪  filoverføring+vise resultat når ferdig\n\nPotensielle
         ↪  problemer:\nLite tid igjen, mangler vi fortsatt logging og kø
         ↪  som var i den originale kravspeken?",
        "user": "UGMTT84F9",
        "ts": "1556665668.017600"
    }
][
    {
        "client_msg_id": "5d7434dd-a26b-4a5c-aff2-d1f616d3cc3d",
        "type": "message",
        "text": "Hva jeg har gjort:\nFant python LDAP library som
         ↪  håndterer AD communication enkelt.\nFikk oppgaver
         ↪  udtelt.\n\nHva jeg planlegger:\nLitt koding, mye
         ↪  skriving\/lesing\/retting.\n\nPotensielle problemer:\nIngen
         ↪  ordentlig mulighet til å teste LDAP biblioteket, så blir bare
         ↪  litt framework som _burde_ funke, men er mulig det ikke
         ↪  går.",
        "user": "UGMTT84F9",
        "ts": "1556760171.020400"
    }
][
    {
        "client_msg_id": "f8a08eea-fc96-426b-a46a-b6eff90860f4",
        "type": "message",
    "text": "Hva jeg har gjort:\nStudert eksisterende windows av
    ↪ roller før jeg begynte å implementere min egen. Har ikke
    ↪ begynt å teste men er 99% sikker på at det ikke virker
    ↪ akkurat nå;)\nBegynt å lage use case og misuse case sammen
    ↪ med beskrivende tabeller. Skrevet litt om funksjonelle,
    ↪ operasjonelle og eksterne krav.\n\nHva jeg planlegger:\nTeste
    ↪ ut windows av rolle og prøve å få det til å virke men blir
    ↪ ikke førsteprioritet. Gå over hva jeg har skrevet i kravspek
    ↪ og føre inn i Overleaf. Lage sekvensdiagram.\n\nPotensielle
    ↪ hindringer:\nVar en spesifikk ting i win av rollen som jeg
    ↪ ikke forsto. Med emsisoft sin rolle som eksempel så brukes
    ↪ det en install_path variabel. Jeg skjønner funksjonen den har
    ↪ i rollen men ikke hvordan man vet hvilken path den skal ha,
    ↪ og vankselig å lage en fungerende rolle uten det. Fint om
    ↪ noen kunne sett på det :slightly_smiling_face:",
    "user": "UGMNQ83QS",
    "ts": "1556802578.025800",
    "edited": {
      "user": "UGMNQ83QS",
      "ts": "1556802625.000000"
    }
  },
  {
    "client_msg_id": "7db4e07e-8783-4a00-b3d2-c75465d3d8bc",
    "type": "message",
    "text": "Hva jeg har gjort:\nBegynt å skrive om
    ↪ introduksjonen.\n\nHva jeg planlegger:\nSkrive om
    ↪ introduksjonen.\n\nPotensielle hindringer:\nIntroduksjonen må
    ↪ praktisk talt totalendres. Vil ta tid.",
    "user": "UGMTT84F9",
    "ts": "1556850717.028100"
  }
][
  {
    "client_msg_id": "34ee7762-7d82-40d9-9dd2-3ed4e0ef75cf",
    "type": "message",
    "text": "Hva jeg har gjort fredag mai 3:\nSkrevet om use case og
    ↪ high-level use case. Laget sekvensdiagram men ikke lagt til i
    ↪ rapporten enda.",
    "user": "UGMNQ83QS",
    "ts": "1557060793.029700"
  },
  {
    "client_msg_id": "e66e192b-b1fe-4db2-a4e1-0fe8f394fbcf",
    "type": "message",
    "text": "Hva jeg hart gjort:\nSkrevet om malware og malware
    ↪ detections innen static og automatic.\n\nHva jeg
    ↪ planlegger:\nSkrive om automatisk provisjonering generelt +
    ↪ info om ansible, puppet, salt &amp; chef. Skulle egentlig
    ↪ være gjort til i dag men må se hva jeg rekker. Drar fra
    ↪ skolen nå for å trene og lage middag, må se om jeg får gjort
    ↪ noe etter det.\n\nPotensielle hindringer:\nFinne gode
    ↪ kilder.",
        "user": "UGMNQ83QS",
        "ts": "1557060945.032000",
        "edited": {
            "user": "UGMNQ83QS",
            "ts": "1557060958.000000"
        }
    },
    {
        "client_msg_id": "2e04667f-4b7f-44ff-b650-09e0656f9afd",
        "type": "message",
        "text": "Hva jeg har gjort:\nIntroduksjon nå 90% ferdig og 99%
         ↪  presentabel, mangler kun få småting.\n\nHva jeg
         ↪  planlegger:\nReview checke det jeg hadde fått beskjed
         ↪  om.\n\nPotensielle hindringer:\nEvt. manglende kilder, tror
         ↪  derimot det ikke trengs noe særlig av i introduksjon.",
        "user": "UGMTT84F9",
        "ts": "1557108112.034400",
        "edited": {
            "user": "UGMTT84F9",
            "ts": "1557108197.000000"
        }
    }
][
    {
        "client_msg_id": "8ddb07a2-84cc-48e9-8561-7743b7c50eeb",
        "type": "message",
        "text": "Hva jeg har gjort:\nLest gjennom og utbedret på
         ↪  teori.\n\nHva jeg planlegger:\nFortsette å lese gjennom og
         ↪  utbedre section by section.\n\nPotensielle hindringer:\nTar
         ↪  tid, men går.",
        "user": "UGMTT84F9",
        "ts": "1557455360.001600"
    }
][
    {
        "client_msg_id": "771c7543-0971-43ee-a615-8dcae72cf1f7",
        "type": "message",
        "text": "Hva jeg har gjort:\nUtarbeidet og utvidet generelle
         ↪  delen under Analytical Infrastructure med. Muligens en del
         ↪  over på implementasjon, men ellers introdusert og oppsummert
         ↪  alle grunnprinsipper som IRMA og andre slike infrastrukturer
         ↪  logisk sett vil bygge på.\n\nHva jeg planlegger:\nFortsette
         ↪  med readthrough, skrive om, skrive på, kommentere, notere,
         ↪  merke, etc etc\n\nPotensielle hindringer:",
        "user": "UGMTT84F9",
        "ts": "1557497679.004400"
    }
][
    {
        "client_msg_id": "b6ae94e9-8903-470f-889f-74aec09f119d",
        "type": "message",
        "text": "Hva jeg har gjort:\nSkrevet generell IRMA teori,
         ↪  begrunnelse for valget, og om arkitekturen (front
         ↪  end+brain).\n\nHva jeg planlegger:\nLese over og fikse
         ↪  største feilene.\n\nPotensielle hindringer:\nUsikker på hva
         ↪  som menes med noen av kommentarene over ønsket innhold. Spør
         ↪  gruppa i morgen.",
        "user": "UGMTT84F9",
        "ts": "1557712630.002600"
    }
][
    {
        "client_msg_id": "56318d4a-23c2-4e62-81a2-85119ed7bbe3",
        "type": "message",
        "text": "Hva jeg har gjort:\nSkrevet om infrastruktur teori til å
         ↪  være litt mer riktig og kortet ned, skrevet litt mer på
         ↪  IRMA.\n\nHva jeg planlegger:\nStarte på kiosk skriving,
         ↪  videre.\n\nPotensielle problemer:\nForferdelig treg dag
         ↪  mentalt.",
        "user": "UGMTT84F9",
        "ts": "1557790030.004400"
    }
]
```

M Efficiency result raw-logs

M.1 Vagrant setup times

```
./Vagrant-04:08:14
    real 4m27,489s
    user 0m14,757s
    sys 0m7,276s
    ./Vagrant-05:08:15
    real 4m32,530s
    user 0m14,912s
    sys 0m7,188s
    ./Vagrant-06:08:15
    real 4m33,801s
    user 0m14,888s
    sys 0m7,094s
    ./Vagrant-07:08:14
    real 4m25,498s
    user 0m14,717s
    sys 0m7,090s
    ./Vagrant-08:08:14
    real 4m28,227s
    user 0m14,823s
    sys 0m7,270s
    ./Vagrant-09:08:14
    real 4m21,528s
    user 0m14,811s
    sys 0m7,049s
    ./Vagrant-11:19:15
    real 4m25,855s
    user 0m14,970s
    sys 0m7,159s
    ./Vagrant-12:19:14
    real 4m27,575s
    user 0m14,668s
    sys 0m7,463s
    ./Vagrant-13:19:14
    real 4m33,370s
    user 0m15,060s
    sys 0m7,055s
    ./Vagrant-14:19:14
    real 4m30,567s
    user 0m14,831s
    sys 0m7,350s
    ./Vagrant-15:19:15
    real 4m35,168s
    user 0m15,122s
    sys 0m7,168s
    ./Vagrant-16:19:14
    real 4m26,931s
    user 0m14,721s
    sys 0m7,186s
    ./Vagrant-17:19:14
    real 4m49,968s
    user 0m14,728s
    sys 0m7,296s
    ./Vagrant-18:19:14
    real 4m42,696s
    user 0m14,884s
    sys 0m7,244s
    ./Vagrant-19:19:14
    real 4m25,392s
    user 0m15,075s
    sys 0m6,930s
    ./Vagrant-20:19:14
    real 4m39,175s
    user 0m14,720s
    sys 0m7,302s
    ./Vagrant-21:19:15
    real 4m26,098s
    user 0m14,788s
    sys 0m7,231s
```

M.2 IRMA-Ansible setup times

```
./Ansible-04:12:42
    real 23m40,000s
    user 1m37,788s
    sys 0m34,459s
```

```
./Ansible-05:12:47
real 26m0,053s
user 1m54,873s
sys 0m44,152s
./Ansible-06:12:48
real 21m11,711s
user 1m30,022s
sys 0m33,830s
./Ansible-07:12:40
real 29m43,627s
user 1m59,311s
sys 0m44,771s
./Ansible-08:12:42
real 20m22,771s
user 1m27,372s
sys 0m32,084s
./Ansible-09:12:36
real 28m32,984s
user 1m55,682s
sys 0m44,371s
./Ansible-11:23:40
real 27m48,141s
user 1m56,124s
sys 0m42,558s
./Ansible-12:23:42
real 25m26,800s
user 1m52,860s
sys 0m43,683s
./Ansible-13:23:48
real 27m9,405s
user 1m53,340s
sys 0m42,754s
./Ansible-14:23:45
real 29m0,267s
user 2m1,164s
sys 0m45,308s
./Ansible-15:23:50
real 24m44,946s
user 1m49,631s
sys 0m43,475s
```

```
./Ansible-16:23:41
real 28m12,310s
user 1m57,511s
sys 0m45,084s
./Ansible-17:24:04
real 27m13,812s
user 1m50,723s
sys 0m43,485s
./Ansible-18:23:57
real 27m25,137s
user 1m59,829s
sys 0m44,488s
./Ansible-19:23:40
real 20m25,919s
user 1m30,191s
sys 0m32,806s
./Ansible-20:23:53
real 28m32,542s
user 1m56,747s
sys 0m44,159s
./Ansible-21:23:41
real 24m51,022s
user 1m51,585s
sys 0m42,443s
```

NTNU
Norwegian University of
Science and Technology