Bachelor's project

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

Daniel Krohn Siqveland
Nestor Gerrardo Fortique
Stian Rønningen

Platform for malwareanalysis

Plattform for skadevareanalyse

Bachelor's project in Informasjonssikkerhet
Supervisor: Eigil Obrestad
May 2019

Daniel Krohn Siqveland
Nestor Gerrardo Fortique
Stian Rønningen

Platform for malwareanalysis

Plattform for skadevareanalyse

Bachelor's project in Informasjonssikkerhet
Supervisor: Eigil Obrestad
May 2019

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

Sammendrag av Bacheloroppgaven

Tittel:	**Plattform For Skadevareanalyse**
Dato:	14.01.2019
Deltakere:	Daniel Krohn Siqveland Stian Rønningen Nestor Gerardo Fortique
Veiledere:	Eigil Obrestad
Oppdragsgiver:	NTNUSOC
Kontaktperson:	Christoffer Vargtass Hallstensen, christoffer.hallstensen@ntnu.no
Nøkkelord:	Plattform, Skadevare, Analyse, Cuckoo, Ansible, Sandkasse, TOR, Engelsk
Antall sider:	51
Antall vedlegg:	11
Tilgjengelighet:	Åpen

Sammendrag:

I en moderne verden er det viktig å være oppdatert og kunne lære mer om skadevare. NTNU SOC presenterte et ønske om å lage en plattform som kunne analysere skadevare og som kunne brukes i produksjons-, utviklings-, og testmiljøer. Det var et ønske om at plattformen kunne installeres automatisk i forbindelse med bruk i undervisningsscenarioer, men også fordi den skulle være lett å vedlikeholde og videreutvikle. Det var også ønskelig at plattformen skulle benytte seg av Cuckoo Sandbox, med tett integrering av simulerte nettverkstjenester som The Onion Router(TOR), Virtual Private Network(VPN), og InetSim. Installasjonen av Cuckoo Sandbox ble automatisert ved hjelp av Ansible, et verktøy som brukes av server administratorer verden rundt. Med dette prosjektet har vi levert en løsning som innfyller ønskene til oppdragsgiveren, NTNU SOC, og som etter kort tid vil bli tatt i bruk som en del av NTNU SOC sitt daglige arbeid i å beskytte både studenter og ansatte ved NTNU. Prosjektgruppen har hatt fokus på profesjonell arbeidsmetodikk, ved for eksempel en agil tilnærming av Waterfall-metoden i løpet av utviklings perioden, og har brukt Kanban som hjelp til å skrive denne rapporten.

Summary of Graduate Project

Title:	**Platform for malware analysis**
Date:	14.01.2019
Authors:	Daniel Krohn Siqveland Stian Rønningen Nestor Gerardo Fortique
Supervisor:	Eigil Obrestad
Employer:	NTNUSOC
Contact Person:	Christoffer Vargtass Hallstensen, christoffer.hallstensen@ntnu.no
Keywords:	Platform, Malware, Analysis, Cuckoo, Ansible, Sandbox, TOR, English
Pages:	51
Attachments:	11
Availability:	Open

Abstract:

In the modern world, it is essential to be up-to-date and to learn more about malware. NTNU SOC presented a wish to create a platform that could analyze malware samples and which could be used in production, development, and testing environments. There was a wish that the platform could be installed automatically in combination with use in teaching scenarios, but also because it should be easy to maintain and develop. It was also desirable that the platform was built around Cuckoo Sandbox, with close integration of simulated network services such as The Onion Router (TOR), Virtual Private Network (VPN), and InetSim. The installation of Cuckoo Sandbox was automated using Ansible, a tool used by server administrators around the world. With this project, we have delivered a solution that fulfills the wishes of the employer, NTNU SOC, and which after a short time will be used as part of NTNU SOC's daily work in protecting both students and employees at NTNU. The project team has focused on professional work methodology, for example, by an agile approach to the Waterfall method during the development period, and has used Kanban to help write this report.

Preface

This is a bachelor thesis in IT-Operations and Information Security written and performed by the following students:

Daniel Krohn Siqveland - 473126 - desiqvel@stud.ntnu.no
Stian Rønningen - 140362 - stiaron@stud.ntnu.no
Nestor Gerardo Fortique - 480240 - nestorgf@stud.ntnu.no

First of all the group would like to thank NTNU SOC for giving us such an interesting and challenging project to work on. The group would also like to thank our employer, Christoffer Vargtass Hallstensen, and our supervisor, Eigil Obrestad, for guiding, helping and inspiring us throughout the whole project. Thanks to Frode Haug and Tom Røise for helping the group with great presentations about report writing and project planning. Last but not least the group would like to thank the Digital Security Section and the Institutt for informasjonssikkerhet og kommunikasjonsteknologi (IIK) for lending us the equipment necessary to make this project a reality.

Contents

List of Figures

List of Tables

Listings

Acronyms

API Application Programming Interface. 23

Bash Bourne again shell. xiv

Bionic Beaver Ubuntu 18.04 LTS. 2, 9, 19, 20, 23, 42, 46

DHCP Dynamic Host Configuration Protocol. xv

DNS Domain Name Server. xv, 9, 38

IDS Intrusion Detection System. 2, 9, 57

IIK Institutt for informasjonssikkerhet og kommunikasjonsteknologi. iii

IPS Intrusion Prevention System. 9

LAN Local Area Network. 18

NAT Network Address Translation. xvi, 19, 20

NIC Network Interface Card. x, 25, 26, 34, 35

NSM Network Security Monitoring. 9

NTNU Norges Teknisk-Naturvitenskapelige Universitet. 1–4, 12, 13, 48

OS Operating System. 6, 40, 43, 46, 48

PCAP Packet Capture. 9

PID Process Identifier Number. 21, 36

PPA Personal Package Archives. xvii

SOC Security Operations Center. 1, 2, 4, 43, 48

SSH Secure Shell. 11, 23

SSL Secure Sockets Layer. 30

TLS Transport Layer Security. 30

TOR The Onion Router. 13

UAC User Account Control. 21

URL Uniform Resource Locator. 39

VCSA vCenter Server Appliance. 12, 17, 18, 23, 31, 42, 69

VPN Virtual Private Network. 1, 10, 13, 18, 20

Xenial Xerus Ubuntu 16.04 LTS. 2, 19, 42

Glossary

Android A smartphone Operating System developed by Google and used by many smartphone manufacturers worldwide, one of which is Samsung. xvi, 2, 8, 21, 28, 32, 45, 46, 56, 63

Ansible Ansible is an open-source IT automation engine with which you can provision, configure, manage and deploy machines. 2, 11, 14–16, 22–24, 26, 46, 48, 57, 61, 63

API A software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API. 8

Backlog Product backlog is a list of tasks that is not yet started on. 16

Bash Bourne again shell (Bash) is a command process that typically runs in a text window where the user types in commands that cause actions. 23

Bitbucket Bitbucket is a version management solution designed for professional teams. It is a central place to manage git repositories, collaborate on source code and it helps making the development flow smoother. 20

Botnet A botnet is a group of computers connected in a coordinated fashion for malicious purposes. Each computer in a botnet is called a bot. These bots form a network of compromised computers, which is controlled by a third party and used to transmit malware or spam, or to launch attacks. 1

Configuration Files Configuration files are files used to configure the parameters and initial settings for some computer programs. 8, 19, 21

Container A container is a software package that contains everything the software needs to run. The package includes the executable program as well as system tools, libraries, and settings. Containers are not installed like traditional software programs, which allows them to be isolated from the other software and the operating system itself. 43, 46, 56

Cuckoo Cuckoo is an open source automated malware analysis program that allows you to trow any virus in it and it will give you a detailed report on what the virus is trying to do. viii, xviii, 1–4, 8, 10, 12–16, 18–26, 28–34, 36–43, 45, 46, 48, 63, 64, 69

Cuckoo Rooter Rooter helps Cuckoo out with running network-related commands in order to provide per-analysis routing options. 10, 14, 21, 22, 36, 39

Debugging Debugging is the routine process of locating and removing computer program bugs, errors or abnormalities, which is methodically handled by software programmers via debugging tools. 45

DevOps Devops is a software developing practice made to reduce the time of developement of software while delivering features, fixes, and updates frequently. 57

DHCP Dynamic Host Configuration Protocol (DHCP) is a network management protocol used on UDP/IP networks whereby a DHCP server dynamically assigns an IP address and other network configuration parameters to each device on a network so they can communicate with other IP networks. 13, 21

Digitalocean Digital Ocean is a company where both private and enterprise actors can buy servers online to deploy and deliver their application to the masses. 19, 20, 26

DNS Domain Name Server (DNS) is a protocol that translates domain names to IP addresses so browsers can load Internet resources. 13, 29

Docker Docker is a program designed to facilitate the use of containers, making it easier to create, deploy, and run applications. 43, 46, 56, 61, 63

Elasticsearch Elasticsearch is an open source, broadly distributable, readily scalable search engine. Accessible through an extensive and elaborate API, Elasticsearch can power extremely fast searches that support your data discovery applications. 14, 15, 19, 20, 46

Environment The set of facilities, such as operating system, windows management, database, etc., that is available to a program when it is being executed by a processor, in this project when the group refers to environment. 6–8, 10–12, 17, 18, 46, 60

ESXI VMware ESXi is an enterprise-class, type-1 hypervisor developed by VMware for deploying and serving virtual computers. 6, 12, 17, 18, 20, 23, 28, 31, 42

Firewall A firewall is software used to maintain the security of a private network. Firewalls block unauthorized access to or from private networks and are often employed to prevent unauthorized Web users or illicit software from gaining access to private networks connected to the Internet. 21

Gantt Diagram A Gantt diagram is an illustration of a projects schedule, and it shows all the phases of the project. When those phases are due to start, when the phases end, and which of those phases are running at the same time. 16, 61

Guest The virtual machine running inside the Host. 6

Home Directory A directory where the user stores all personal information and files as well as user information. 19

Host A server which runs virtual machines inside itself. xv, 6, 7, 18

HTTP HTTP or Hypertext Transfer Protocol is a set of rules for transferring files (text, images, sound, video, and other multimedia files) on the World Wide Web. 9

Hypervisor A hypervisor is a process that creates and runs virtual machines (VMs). A hypervisor allows one host computer to support multiple guest VMs by virtually sharing its resources, like memory and processing [1]. 2, 6, 11, 12, 14, 18, 20, 23, 28, 30, 42

INetSim INetSim is a software suite for simulating common internet services in a lab environment, e.g. for analyzing the network behaviour of unknown malware samples. 8, 10, 11, 14, 28, 40

Infected computer A computer that is or has been affected by a virus in any way. 6, 8

interface The layout or design of the interactive elements of a computer program, an online service, or an electronic device. 8, 20

iOS Apples smartphone implementation of Operating System running on phones developed by Apple, for example the iPhone X. 63

IP-address Identifies each computer using the Internet Protocol to communicate over a network. 9–11, 23, 26, 29–36

Kernel The kernel is a program that constitutes the central core of a computer operating system. It has complete control over everything that occurs in the system. 45

Linux Linux is a free open source Operating System based on UNIX that was created in 1991 by Linus Torvalds. Users can modify and create variations of the source code, known as distributions, for computers and other devices. It is also the base Operating System for Android. xviii, 8, 46

MacOS Apples implementation of Operating System running on computers made by Apple. 2, 8, 21, 45, 46, 56, 63

Malware Short for malicious software, is any software intentionally designed to cause damage to a computer, server, client, or computer network. 1–4, 6–8, 10, 11, 13, 14, 16, 32, 39, 43, 45, 48, 56, 57, 61, 63

Moloch Moloch is an open source piece of software that can be used to index very large PCAP files into Elasticsearch. 15, 19, 37, 46

NAT Network Address Translation (NAT) is the process where a network device, usually a firewall, assigns a public address to a computer (or group of computers) inside a private network. 13

OpenVPN OpenVPN is both a VPN protocol and a software that uses VPN techniques to secure point-to-point and site-to-site connections. 12, 18, 20, 22, 26, 27

Operating System An operating system is a huge program that whose purpose is to provide the user with a clean interface towards hardware. xiv, xvi, xix, 8, 63, 64

pcap Pcap consists of an application programming interface for capturing network traffic. 9

pfSense Firewall, VPN, and router functionality for a fraction of the cost of proprietary alternatives. 12–14, 18, 20

Pillow Pillow is a fork of the python image library that will help the group by taking screenshots of the processes happening in the sandboxes. 21

Playbook Playbooks are Ansible's configuration, deployment, and orchestration language. They can describe a policy you want your remote systems to enforce, or a set of steps in a general IT process. 11, 14–16, 19, 20, 22–24, 26, 28, 30–32, 36, 37, 42–46, 48, 63

Repository Often referred to as a Personal Package Archives (PPA). A repository is a private collection of prebuilt packages for Linux systems that users can add to their system to install the latest version of a package. When using a repository the user will not have to compile the program from source code. 19

Risk According to ISO 31000, risk is the "effect of uncertainty on objectives" and an effect is a positive or negative deviation from what is expected. Risk = Likelihood * Impact. 7, 58, 59

Routing Routing refers to establishing the routes that data packets take on their way to a particular destination. 8, 10, 13, 39, 40

Sandbox In computer security, a sandbox is a security mechanism for separating running programs. It is often used to execute untested code, or untrusted programs from unverified third parties, suppliers, untrusted users and untrusted websites. The sandbox typically provides a tightly controlled set of resources for guest programs to run in, such as scratch space on disk and memory. Network access, the ability to inspect the host system or read from input devices are usually disallowed or heavily restricted. xviii, 1, 2, 7, 12–16, 18, 20, 21, 25, 29–31, 38–40, 42, 43, 45, 56

Script A script is a lightweight, quickly constructed and possibly single-use tool programming language. 14, 20, 40, 60

Security Operations Center A unit that deals with security issues on an organizational and technical level. 8

Signature A signature is any detection method that relies on distinctive characteristics being present in an exploit. These signatures are specifically designed to detect known exploits as they contain a specific set of characteristics. 7, 8, 57

Snapshot A snapshot of a virtual machine is a file-based representation of the state of the virtual machine at a given time. It includes disk data and configuration data of the VM. With a snapshot you can restore a machine to a previous state. 14, 23, 30, 37, 43

Spoof Spoofing is the act of disguising a communication from an unknown source as being from a known, trusted source to attempt to gain access to something they should have access to. 1, 13, 14, 56

subnet A subnet is a logical partition of an IP network into multiple, smaller network segments. It is typically used to subdivide large networks into smaller, more efficient subnetworks. 13, 15

Subnet 1 The subnet that makes sure that the platform has Internet connectivity for updates and other services. 18, 23, 34, 35

Subnet 2 The subnet where the sandboxes reside. Used for analyzing and routing between the Sandboxes and Cuckoo. 18, 23, 30, 31, 34, 36

Suricata Suricata is a free and open source, mature, fast and robust network threat detection engine. The Suricata engine is capable of real time intrusion detection (IDS), inline intrusion prevention (IPS), network security monitoring (NSM) and offline pcap processing. 9, 15, 19, 32, 35, 46

systemd systemd is a system and service manager for Linux. It has replaced init for startup- and servicemanagement [2]. 15, 19–21, 36

TOR The Tor browser is a web browser that anonymises your web traffic by bouncing your communications around a distributed network of relays run by volunteers all around the world. 1, 10, 11, 13, 14, 18, 34, 35, 40, 56

Traceroute Traceroute is a computer network diagnostic tool for displaying the route and measuring transit delays of packets across an Internet Protocol network. 40

Trello Trello is a collaboration tool that organizes your projects into boards. Trello tells you what's being worked on, who's working on what, and where something is in a process. 47

Ubuntu Ubuntu is an open-source operating system (OS) based on the Debian GNU / Linux distribution. 14, 19, 25

vCenter Server Appliance The vCenter Server Appliance is a preconfigured Linux virtual machine, which is optimized for running VMware vCenter Server and the associated services on Linux. 18, 63

Version Management In this project, Version Management, means the possibility of installing different versions based on a preset variable in Ansible. 1

Virtual Environment At its core, the main purpose of Python virtual environments is to create an isolated environment for Python projects. The isolated environment means that each project can have its own dependencies(packages) which will not be updated with system updates to avoid breakage. 20, 43

Virtualbox VirtualBox is a powerful x86 and AMD64/Intel64 virtualization product for enterprise as well as home use. 45

VMWare VMware is a virtualization and cloud computing software that allows you to run multiple virtual machines on a single physical machine. 2, 6, 12, 23, 45, 56

Volatility Volatility [3] is an open collection of forensic tools that facilitate the extraction of digital artifacts from volatile memory (RAM) samples. 10, 15, 19, 31, 43

VPN A vpn is a tool used to hide an users traffic by encrypting it. A vpn also hides the users ip-address by redirecting the users traffic through a server. 8, 11, 14, 19, 20, 40, 46, 56

vSphere Connecting ESXi to vCenter Server Appliance turns it into a vSphere Server, expanding its features and allowing full memory dumps of virtual machines. 1, 2, 6, 12, 18, 20, 23, 28, 30, 31

WebGUI webgui, is a Website Graphical User Interface that allows users to interact with electronic devices through graphical icons and visual indicators, instead of text-based user interfaces, typed command labels or text navigation. 20

Windows Microsofts implementation of Operating System. 1, 2, 8, 16, 21, 29, 30, 41–43, 45, 46, 48, 56, 63, 64

Yaml YAML (YAML Ain't Markup Language) is a data-oriented language structure used as the input format for diverse software applications. An application user or administrator specifies data in a YAML file, which the application then can read. 11

Yara Yara is an opensource software that classifies malware by matching it against textual or binary patterns. 8, 10, 15, 19, 43

1 Introduction

Since the dawn of computers and computer networks, Malware has been in existence. Malware is created by hackers or other evil-minded to help break into computers or computer systems, through exploiting vulnerabilities in software. Today there exist hundreds of thousands of different malware. The goals of malware can vary, but the most popular ones usually turn a device into a slave of a Botnet or kidnap its files and encrypt them. Malware expands as fast as or even faster than we can protect from it. It is imperative to understand Malware by how it operates and how the mind behind the Malware think. New sophisticated methods to lure a user into downloading malware and new types of malware are discovered every day. The only way malware researchers can prepare themselves from these threats is to understand their method and structure. With this thesis, we will take a closer look at how an immutable platform is used to analyze any malware and present the results.

1.1 Problem description

This project was created by Christoffer Vargtass Hallstensen, head of Security Operations Center in NTNU Gjøvik. NTNU SOC needed a service platform that could analyze potential malware quick and risk-free. Therefore, SOC made a bachelor submission, which had these factors in mind.

It is time-consuming to do malware analysis manually. There is, therefore, a need to scale up automatic Malware analysis. Scaling issues are addressed with the automation of development, testing, and production. Automatic Malware analysis is versatile but is not always accurate, so one needs to be able to do manual Malware analysis as well. Another challenge is maintaining virtual machines for both manual and automated analyses. The group solved this by using vSphere which makes maintenance of sandboxes easier and the possibility to use the same Sandboxes for both manual and automated analysis when needed, e.g., via VMware Workstation.

1.2 Goals and research question

The goal of the project will be to simplify Malware analysis by being able to use the same platform for both automatic and manual analysis. The group will automate the installation of a malware analysis lab which will be scalable and can be further developed.

1.2.1 Scope

Implementing the Cuckoo analysis platform so it can run Malware on Windows Sandboxes. See appendix A for the full description of the task.

NTNU SOC wanted a platform that had a configuration component that allowed for Version Management and had integration towards systems for threat intelligence. Another requirement from the employer was the implementation of different network services such as TOR and Virtual Private Network (VPN) to Spoof the location of the analyzer and lure the attacker into unleashing the full potential of the Malware for better analysis.

The environment will be built in a VMWare vSphere Hypervisor. Every other machine run in the same environment will be virtual machines. The virtual machines the group will use are Sandboxes for Malware testing, and a Ubuntu 18.04 LTS server for running Cuckoo. Network-based Intrusion Detection System systems will be installed alongside Cuckoo to ensure that traffic generated by the malware is captured and inspected. Cuckoo installation will be automated using Ansible.

1.2.2 Limitation of scope

The original plan was to install the platform on a Ubuntu 16.04 LTS (Xenial Xerus) server (hereby referred to as Xenial Xerus). Since Canonical [1] will no longer be offering updates to Xenial Xerus after April 2021, [4] the students have decided to change the operating system of the Cuckoo host to Ubuntu 18.04 LTS (Bionic Beaver) (hereby referred to as Bionic Beaver), which is supported until April 2023.

The implementation of containers proved to be more challenging than helpful. Therefore the students, together with the employer, excluded containers during the midterm review which can be read in appendix D.

The employer and the students decided to focus on the implementation of a platform for analyzing Windows. The implementation for MacOS and Android would be optional and put at future work.

1.3 Motivation

The group has a common interest in Malware, and they like practical challenges. The group agreed that the bachelor thesis should include practical work, such as networking, virtualization, and advanced server implementations. When "platform for skadevareanalyse" had the presentation about the subject and the task, the group knew right away, this would be a perfect challenge.

In previous subjects, the students learned the basics of automation, infrastructure and Malware so this would be an excellent opportunity to improve the groups' knowledge in these areas. Ansible and Cuckoo were entirely new for the students, and the group knew right away it would be fun in a challenging way to learn new tools and applications.

1.4 Roles

This section will give the reader a better idea of who the authors are, who the employer is, and who the supervisor is.

1.4.1 Employer

Christoffer Vargtass Hallstensen, head of Security Operations Center at Norges Teknisk-Naturvitenskapelige Universitet in Gjøvik, provided the task. The students had progress meetings with the employer every time the group deemed it necessary. In these progress meetings, the group discussed the scope of the project, asked for technical advice regarding the project and showed him some demos of Cuckoo Malware analysis in action.

[1]The Company behind Ubuntu: https://www.canonical.com/

1.4.2 Supervisor

The groups supervisor is Eigil Obrestad, an assistant professor at NTNU. The group had progress meetings with him nearly every Monday. In these meetings the group discussed everything from different infrastructures for Cuckoo and how to write the thesis.

1.4.3 Authors

The authors of this thesis are Daniel Emil Krohn Siqveland, Nestor Gerardo Fortique, and Stian Rønningen. All of whom study IT-Operations and Information Security at NTNU Gjøvik. The group initially received a different project from an external employer, but the group decided that this project was much more interesting and felt they could learn more from this project.

1.5 Students experience

All of the members of the group have a passion for IT-security, and that is why it was an easy decision to take on this project. All of the group members have experience in coding and are familiar with automating infrastructure with the help of Puppet.

Before this thesis, none of the students had any previous experience with Malware analysis, nor had they ever heard of Cuckoo before. The students had to read up on both subjects. Equally none of the group members had written a report of this proportion before, meaning that the group had a lot to learn.

1.5.1 Previous competence

The projects participants have previous experiences from the subjects taken during three years of studying before writing this thesis. Among many subjects, the ones the group felt most relevant for this project can be seen in table 1.

Code	Name	Relevance
IMT2006	Computer Networks	Subnetting
IMT2007	Network Security	VPN, Firewall, Routing
IMT2008	ITSM, Security and Risk Management	Risk Management
IMT2243	Software Engineering	Software development models
IMT2571	Data Modelling and Database Systems	Database administration and Security
IMT2282	Operating Systems	Virtualization, Access Control and Malware, Bash
IMT3003	Service Architecture Operations	Automation, Databases and Web Applications, Architecture
IMT3004	Incident Response, Ethical Hacking and Forensics	Internet and Network Forensics
IMT3005	Infrastructure as Code	Logging, Monitoring and Auditioning/Testing, Configuration Management, Rapid Deployments

Table 1: Previous Competence - Subjects

1.6 About the report

The report is written in Latex, and will have clickable links in PDF-format. The reader will be able to follow links to references, sources, glossaries and acronyms throughout the report. There will also be lists of figures, tables, listings and a table of contents, all clickable. For bibliography, the document is using JabRef to Cite with BibLaTex [5]. A more thorough list of tools used in the making of this thesis can be seen in table 2.

1.6.1 Project period

The project will start on 10.01.2019 and it will end 20.05.2019. A presentation of the thesis will be held 04.06.2019.

1.7 Target audience

The target audience for this project is mainly our employer NTNU SOC, but it can also be an interesting read for other SOC analysts, Malware researchers, academics, digital forensics investigators, incident responders, and students that need a dynamic and agile lab environment for malware analysis and research.

1.8 Thesis structure

This section will describe how the thesis is structured.

Introduction

Chapter 1 gives an overview of the thesis, including the description of the task and the protagonists. It will also tell the reader about what the students have learned from a project of this size, and what experience the group had from previous projects.

Background

Chapter 2 provides the necessary background information for the reader to be able to understand the different types of malware analysis and tools/services that will be used in this thesis. The analysis software uses a variety of services that will be briefly described.

Design

Chapter 3 explains more accurately how the architecture and services are connected, and its use in this environment and platform.

Implementation

Chapter 4 will introduce the methodology, together with the technologies used and how the implementation process of the architecture design was configured and conducted.

Automation

Chapter 5 covers the documentation of the automation installation of Cuckoo.

Testing and Verification

Chapter 6 points out errors that occurred during the implementation with examples and how the troubleshooting was conducted. Additionally, the chapter covers how the different tool's functionality was verified.

Discussion

Chapter 7 will discuss the project from the groups' perspective. The reader will make sense of what obstacles the students faced along the way; the alternative approaches that could have been taken with suggestions for future work, and a critical view on the execution of the task — lastly, evaluation of the groups' effort.

Conclusion

Chapter 8 concludes this thesis. It will discuss the project results, the groups' achievements, and a closing statement from the students.

Bibliography

Displays a list of sources used in this thesis.

Appendix

Short content description of the appendix:

Appendix A: Task description.

Appendix B: Project agreement.

Appendix C: The students plan regarding the embodiment of the thesis.

Appendix D: Midterm review with the employer.

Appendix E: Code for route.py.

Appendix F: OpenVPN Systemd integration.

Appendix G: Cuckoo Systemd integration.

Appendix H: Code for vsphere.py.

Appendix I: Code sample of vsphere.conf.

Appendix J: Cuckoo Web GUI options in an analysis.

Appendix K: Ansible directory structure.

2 Background

To build the malware analysis platform, the group used a lot of different tools and concepts; in this chapter, those tools and concepts will be explained in detail. Giving the readers the necessary information to fully comprehend the rest of the thesis.

The purpose of malware analysis is usually to provide the information one needs to respond to a network intrusion. The goal will be to determine precisely what happened and ensure all of the Infected computer machines and files have been located. When analyzing suspected malware, the key is to find out exactly what a particular suspect binary can do, how to detect it on your network, and how to measure and contain its damage [6]. In these evolving times, detecting and removing malware artifacts is not enough; it is vitally important to understand how they operate to understand the context, the motivations, and the goals of a breach.

2.1 Virtual machine

A virtual machine can be thought of as a computer inside a computer. The machine which runs the virtual machine is often referred to as a Host, while the virtual machine itself is often referred to as a Guest. The host shares its resources with the guest, meaning memory, CPU, Disk, and I/O will be allocated between the two.

To be able to run a virtual machine, the Host needs to have a Hypervisor software installed. There are many variants of Hypervisors, including ones that can be installed as its own Operating System(VMWare ESXI) and others that can be run on top an OS(VMWare Workstation). The Hypervisor the group chose to use in this project is VMWare vSphere. vSphere can be thought of as an extension to VMWare ESXI which allows an administrator to manage multiple ESXI Hosts with one installation of vSphere, in addition to adding more features to the standard ESXI installation.

The Guests running on the Host is considered to be in a safe Environment, where one use case is using this Environment as a platform to test and analyze malware.

2.2 Malware

Malicious software, often referred to as Malware, plays a part in most computer intrusion and security incidents. Malware is often designed to steal sensitive information, or otherwise harm a computer or a network causing downtime. During the last years there has been a significant increase in new malware samples [7], thus increasing the need to analyze and learn how to protect against these threats.

2.2.1 Malware analysis

Malware analysis is the art of dissecting malware to understand how it operates, what it does to the infected system, how to identify it, and how to mitigate or eliminate the damage it may cause. The purpose of malware analysis is to study and determine the functionality, origin, and impact of a given sample.

With millions of malicious software on the Internet and new types of malware being encountered every day, malware analysis is critical for anyone who responds to computer security incidents. There are many ways to gather information from malware; one of these is by performing an analysis on a given malware; this provides a unique Signature for that malware. The Signature can then be added to an anti-virus Signature scan to be able to detect this specific malware. While using signatures to detect malware is a great technique, it is a weakness with anti-virus programs, since most anti-viruses will only look for known signatures provided by its developer [8] meaning the anti-virus can often be classified as outdated. The best way to detect the existence of a new malware signature is to use a Sandbox feature that includes analysis tools.

It is recommended to set up a safe Environment before running an analysis on a Malware sample. Samples of Malware can be full of surprises, and if run in a production Environment, it can quickly spread to other computers on the network and be very difficult to contain and remove. A safe Environment will allow investigation of the malware without exposing other computers on the network to unexpected and unnecessary Risk [9]. Therefore it is preferred to use virtual machines because these machines are isolated from the Host so it does not infect other components than intended. The methods to analyze malware usually fall under two categories, dynamic- and static malware analysis [10].

Static malware analysis

Static or code analysis is usually performed by dissecting the different resources of the binary file without executing it and studying each component. Static analysis can confirm whether a file is malicious, provide information about its functionality, and sometimes provide information which allows the production of simple network Signatures. Basic static analysis is straightforward and can be quick, but it is mostly ineffective against sophisticated Malware and can miss important behaviors [11].

An advanced static analysis consists of reverse-engineering the Malware's internals by loading the executable into a disassembler and looking at the Malwares hardware instructions. The CPU executes the instructions, so advanced static analysis tells you exactly what the Malware does [11].

Static Malware analysis is very similar to manual Malware analysis, because it uses the same techniques to understand how a Malware operates.

Dynamic malware analysis

Dynamic analysis techniques involve running the Malware and observing its behavior on the system in order to remove the infection, produce effective Signatures, or both. Basic dynamic analysis techniques can be used by most people without deep programming knowledge, but they will not be effective with all malware and can miss important functionality [11].

Advanced dynamic analysis uses a debugger to examine the internal state of a running malicious executable. Advanced dynamic analysis techniques provide another way to extract detailed information from an executable. These techniques are most useful when the goal is to obtain information that is difficult to gather with the other techniques[11].

A weakness of dynamic analysis is while testing the malware there is no way of being 100% sure if all of the executable paths of the malware have been successfully tested and mapped [12].

Automation of Analysis

Automation is the use of various control systems for operating and controlling a process or procedure without human assistance. The project's thesis will involve the setup of a Malware analysis platform that is going to be automated. It exists a wide variety of software that will analyze many aspects of what a malware can do - intercept network traffic, changes in file structure and requests to the Operating System. To solve the issue of downloading, installing and linking the software together is where automation becomes handy, with only a few changes in Configuration Files the software can be customized for each specific need.

2.3 Cuckoo Sandbox

This section will give the reader an idea about the main tool and how it is used in this thesis.

2.3.1 What is Cuckoo

Cuckoo Sandbox is the leading open source automated Malware analysis system. In a matter of minutes, Cuckoo will provide a detailed report outlining the behavior of the file when executed inside a realistic but isolated Environment. [13]

Cuckoo Sandbox is free software that automated the task of analyzing any malicious file under Windows, MacOS, Linux, and Android.

2.3.2 What can Cuckoo do

Cuckoo Sandbox is an advanced, extremely modular, and 100% open source automated Malware analysis platform with countless application opportunities.
According to Cuckoo's homepage [1], Cuckoo is able to:

- Analyze many different malicious files (executables, office documents, pdf files, emails, etc) as well as malicious websites under Windows, Linux, MacOS, and Android virtualized Environment.
- Trace API calls and general behavior of the file and distill this into high level information and Signatures comprehensible by anyone working in a Security Operations Center or anyone interested in malware analysis.
- Dump and analyze network traffic, even when encrypted with SSL/TLS. With native network Routing support to drop all traffic or route it through INetSim, a network interface, or a VPN.
- Perform advanced memory analysis of the Infected computer virtualized system through Volatility as well as on a process memory granularity using Yara.

2.3.3 Tools

This section will present some of the important tools that Cuckoo uses when analyzing a file. Each of these tools have an independent purpose, and Cuckoo can take advantage of all of them.

[1]https://cuckoosandbox.org/

Suricata

Suricata is a free and open source, mature, fast and robust network threat detection engine. The Suricata engine is capable of real time Intrusion Detection System (IDS), inline Intrusion Prevention System (IPS), Network Security Monitoring (NSM) and offline PCAP processing. In this project, Suricata is used to intercept traffic in the form of a pcap-file.

Moloch

Moloch is a standalone open source full packet capture system with meta data parsing and searching. Moloch can present intercepted network traffic in a graphical chart in the form of a webpage. Moloch is used as a supplement to Suricata to provide more visibility by indexing the network traffic from the PCAP file into Elasticsearc.

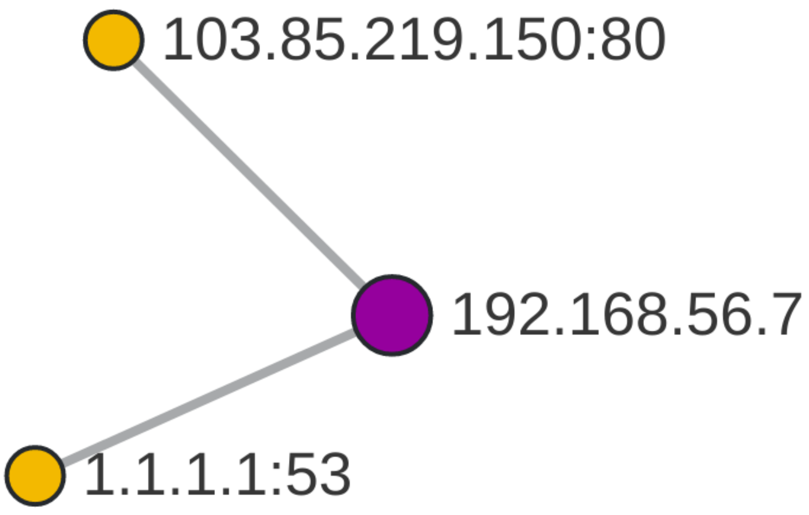


Figure 1: Moloch active connections

Figure 1 shows active connections from IP-address 192.168.56.7. One can see that it contacts **1.1.1.1** with a DNS request(port 53). After resolving the given DNS name, it contacts **103.85.219.150** with a HTTP request(port 80).

Elastichsearch

Elasticsearch is a really fast search and analytics engine and database. Figure 2.1 depicts a working Elasticsearch installation by running **curl localhost:9200** on Bionic Beaver.

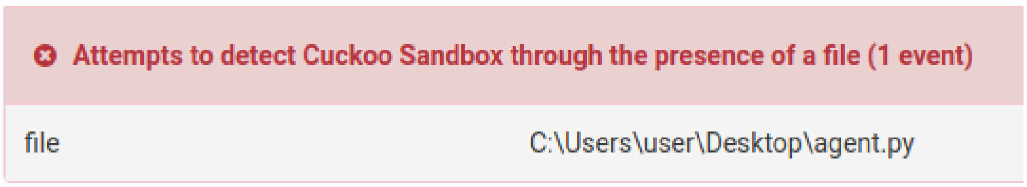
```
{
  "name" : "xvJNENk",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "VNtGOIxlTHCMQb8XR_he6w",
  "version" : {
    "number" : "5.6.16",
    "build_hash" : "3a740d1",
    "build_date" : "2019-03-13T15:33:36.565Z",
    "build_snapshot" : false,
    "lucene_version" : "6.6.1"
  },
  "tagline" : "You␣Know,␣for␣Search"
}
```

Listing 2.1: Elasticsearch

Yara

Yara [14] is a tool that will help an analyst by detecting what the malware is doing and matching it up with a set of rules based on textual or binary patterns so Yara can classify it.

Besides categorizing Malware, Yara [14] can also help the analyst in different ways, one of them is that Yara can detect if the Malware checks if it runs in a virtual Environment.

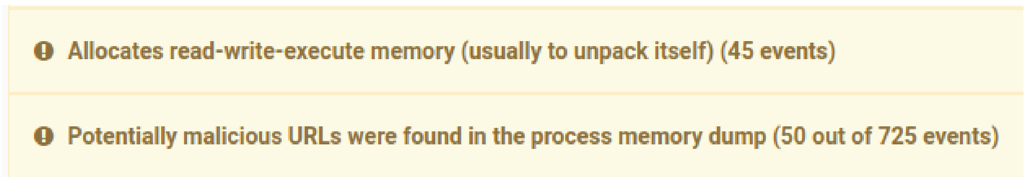

Attempts to detect Cuckoo Sandbox through the presence of a file (1 event)

file C:\Users\user\Desktop\agent.py

Figure 2: Yara rule matching

Volatility

Volatility [3] is an open collection of forensic tools that facilitate the extraction of digital artifacts from volatile memory (RAM) samples. With Volatility integrated into Cuckoo it can analyze memory dumps after running a Malware analysis.


Allocates read-write-execute memory (usually to unpack itself) (45 events)

Potentially malicious URLs were found in the process memory dump (50 out of 725 events)

Figure 3: Volatility warnings

2.3.4 Networking

Cuckoo offers a wide variety of options when it comes to Routing. To do this Cuckoo uses a routing service called Cuckoo Rooter. Cuckoo Rooter supports direct Internet connection, VPN connections, TOR, simulated network services and the default drop all routing.

TOR and VPN are both tools the group will use to hide the location from where the testing is taking place from the Malware. If the malware detects that the traffic is coming from a university network, it might execute differently than if it is coming from an unknown IP-address. INetSim will be used as a service to fake real Internet access from the Malware.

TOR [15]is a routing service that anonymizes its user's traffic by bouncing it around in a distributed network of relays run by volunteers all around the world.

A VPN [16] is a tool that anonymizes the user's traffic by encrypting it, and it also hides its users IP-address by redirecting the traffic from the user through a server.

INetSim [17] is a software suite for simulating common internet services in a lab Environment, e.g. for analyzing the network behavior of unknown Malware samples.

2.4 Ansible

Ansible is an automation tool, developed by Red Hat Inc., capable of administrating multiple hosts at the same time over the Secure Shell (SSH) protocol. Ansible uses human-readable Yaml templates so that users can program repetitive tasks to occur automatically, without learning an advanced language. Ansible can install an entire set of webservers and its backend in a matter of minutes. Ansible is capable of administrating virtual machines and Hypervisors. Ansible Playbook is a collection of different roles put together to allow installation of an entire platform on a single or multiple host simultaneously.

2.4.1 Playbook

The playbook is the core component of any Ansible configuration. An Ansible Playbook contains one or multiple tasks, each of which define the work to be done for a configuration on a managed server.

3 Design

This chapter will describe how the architecture is designed by connecting all the tools in chapter 2 together.

3.1 Architecture

The architecture runs on a VMWare ESXI Hypervisor. The Hypervisor needs to run all of the virtualized Sandboxes for Cuckoo to work. Connecting a vCenter Server Appliance (VCSA) to the ESXI host turns the architecture into a vSphere Environment. Figure 4 describes how the architecture is designed.

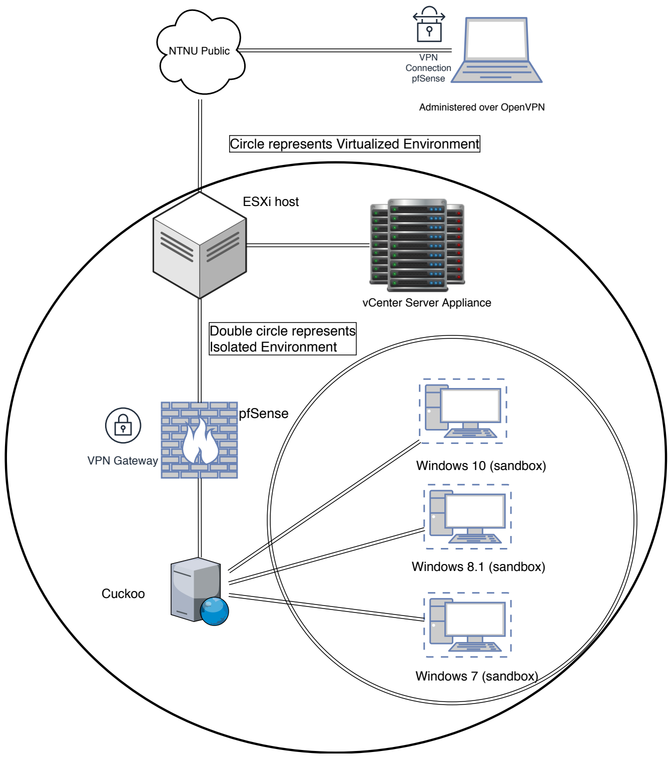


Figure 4: Architecture design

3.2 PfSense

Cuckoo does not have a password protected login and is therefore vulnerable to anyone, if connected NTNU Public directly. pfSense is needed to protect Cuckoo from fraudulent use, and will have an active OpenVPN server which makes it is possible to work on the platform remotely.

3.2.1 Networking

In addition to protecting Cuckoo from NTNU public, pfSense handles Routing, DHCP, DNS and NAT. The architecture needs to have two separate subnets. One subnet in which Cuckoo resides, and one subnet in which the Sandboxes reside. Figure 5 describes the network design in better detail.

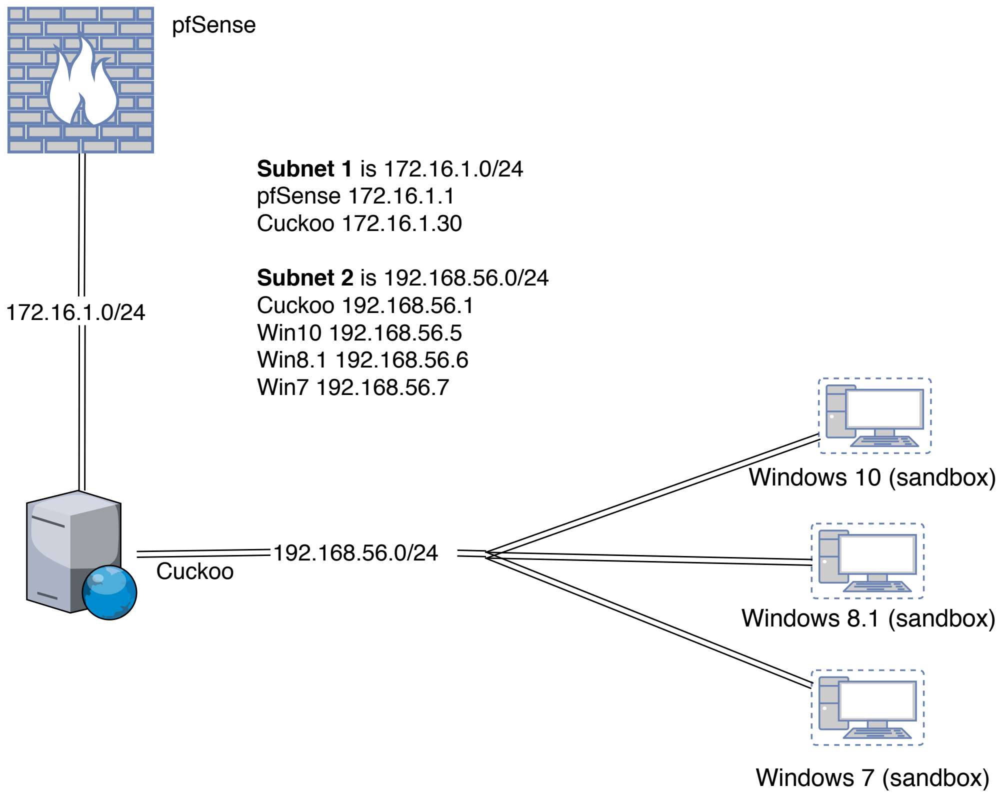


Figure 5: Subnet design

Some Malwares need internet access to reach full potential. Some malware checks the IP-address from where it is executed and may stop when it is from a research center. The next sections describe the routing options for Cuckoo to help an analyzer conceal the location from where the Malware is executed.

The Onion Router - Tor

The Onion Router (TOR) will Spoof the traffic from a Sandbox by distributing the traffic among different TOR nodes. Additionally, Tor encrypts the traffic [15]. Bouncing the encrypted traffic through nodes, ensures that it is nearly impossible for a Malware to know it is being analyzed from NTNU.

Virtual Private Network - VPN

Virtual Private Network (VPN) initiates an encrypted tunneled connection to a remote server [16], which Spoofs the location of the analyzer.

INetSim

INetSim simulates Internet by Spoofing website requests to analyze network behavior of unknown Malware samples [17].

3.3 Cuckoo

Cuckoo needs to run on a Ubuntu server. Everything needed to start and run Cuckoo will be installed by an Ansible Playbook. Cuckoo interacts with the Hypervisor to start, shutdown and restore Snapshots of the Sandboxes. Cuckoo will act as a router between the sandboxes and pfSense to intercept and analyze network traffic generated by the Sandboxes. Using Cuckoo Rooter, Cuckoo is also able to route traffic over internet, TOR, VPN or INetSim. Cuckoo hosts a few databases, including **MongoDB**, **PostgreSQL** and Elasticsearch, all set up and ready to use by the Ansible Playbook.

3.3.1 Sandboxes

This architecture includes three different Sandboxes. The Sandboxes will be used to check if the Malware operates differently on different distributions. The sandboxes have direct contact with Cuckoo by running a python Script, allowing the Sandboxes and Cuckoo to interact to each other.

3.3.2 Tools

Cuckoo uses a variety of tools to perform analysis mentioned in chapter 2, in section 2.3.3. Figure 6 depicts the applications needed.

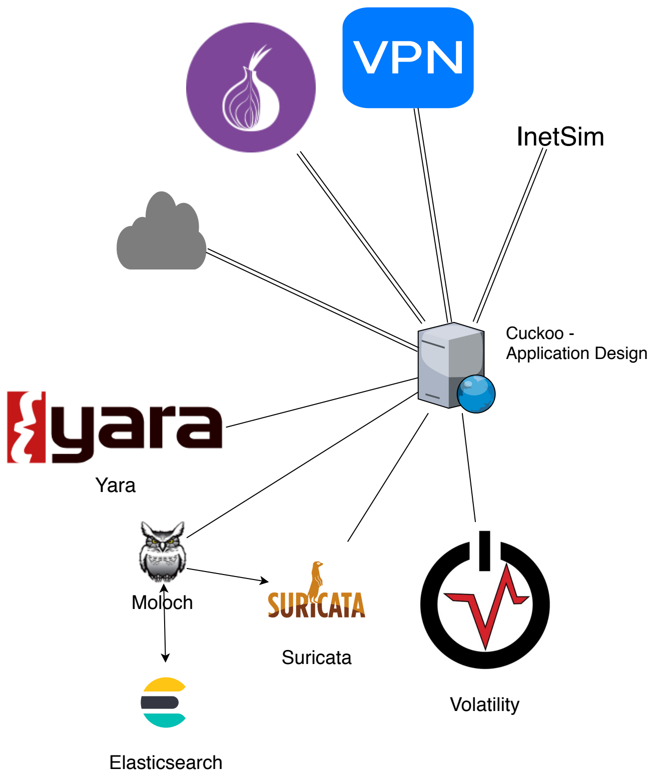


Figure 6: Application design

The tools are dependent on each other to work with the exception of Yara and Volatility which will be detailed better in chapter 4. Suricata must be in place to capture-, intercept- and analyze network traffic generated by the Sandboxes in subnet 2. Elasticsearch and Moloch are both dependent on Suricata to operate.

3.3.3 Systemd

A requirement from the employer was to implement Cuckoo's services to systemd, which is explained in deeper detail in chapter 4.5.6.

3.4 Ansible

The implementation of Ansible, will follow best practice on how to organize the Playbook from the documentation of Ansible [18] and will be further detailed in chapter 5.

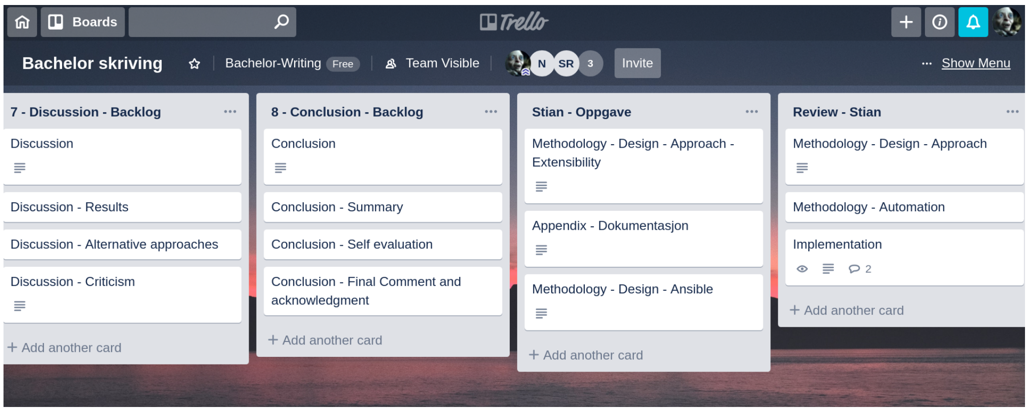
4 Implementation

This chapter will explain how the implementation process of the architecture design was conducted and the methods used in both writing the thesis and workflow. The virtualization platform was the first implementation in this assignment after the foundation was made; the next step was an analyzing server. Cuckoo was firstly implemented manually to understand the concept, then later set up with the help of an Ansible Playbook. The last implementation needed to be hosts for the Malware, which is where Windows was introduced as Sandboxes.

4.1 Methodology

As a development methodology, the students used waterfall, as mentioned in appendix C.5.1. Each sequence or phase in the waterfall methodology must be complete before the next can be started. The waterfall method makes it easy to understand and follow when the project has clear objectives and requirements [19]. Traditionally there is no process for going back, but because of changes in the requirements, some exceptions were made. For the students, the waterfall methodology made it user-friendly to follow the Gantt Diagram flow; when one sequence was done, the students started on a new. As seen in figure C.6.1 implementation has four sub-phases that were worked on in parallel. Waterfall worked great as a development methodology for this project, but for writing the report, the students used a more agile approach.

The students used the Kanban methodology when writing the thesis. The students started by making a Kanban board with a product Backlog in Trello, as seen in figure 7. Trello is a flexible and visual way to manage projects workflow [20]. Using Trello allowed the group to have a better overview of what needed to be done as well as what was left. Trello became the primary tool to arrange the workflow systematically. The Kanban board is to allow team members to track the progress of work through its workflow visually [21].



7 - Discussion - Backlog
Discussion
Discussion - Results
Discussion - Alternative approaches
Discussion - Criticism
+ Add another card

8 - Conclusion - Backlog
Conclusion
Conclusion - Summary
Conclusion - Self evaluation
Conclusion - Final Comment and acknowledgment
+ Add another card

Stian - Oppgave
Methodology - Design - Approach - Extensibility
Appendix - Dokumentasjon
Methodology - Design - Ansible
+ Add another card

Review - Stian
Methodology - Design - Approach
Methodology - Automation
Implementation
2
+ Add another card

Figure 7: Trello

4.1.1 Technologies

Name	Type	Area of application
Trello	Web-based project management application	Project control
Overleaf	Online LaTeX compilator	Thesis document
Bitbucket and Github	Web-based hosting service for version control using Git	Version control
Draw.io	Flowchart Maker and Online Diagram Software	Diagrams and figures
Google Drive	File storage and synchronization service	Document and file storage
Digitalocean	Cloud infrastructure provider	OpenVPN - Cuckoo
pfSense	Firewall/router computer software	OpenVPN - Users
Jabref	Bibliography reference manager	Bibliography
Atom	Text and source code editor	Code editor
Master PDF Editor	Edit PDF files	PDF files
LaTeX Table Generator	Online table generator for LaTeX	Tables

Table 2: Technologies and applications used in the project

Table 2 shows which applications/technologies have been used in the making of this project. These tools helped the group save time by making it easy to review each other's work and also making the report writing process more streamlined.

4.2 Virtualization environment

At the start of the implementation phase as seen in figure C.6.1, the group needed a virtualization Environment, and tried setting up an ESXI server on their personal computers. Installing ESXi on a personal computer did not work, due to the infrastructure requiring more computer resources than what a laptop could handle. The students had to figure out a solution to run ESXI.

After a meeting with the employer, it was concluded that the infrastructure had to be run on a server dedicated to run ESXI. With the assistance of Lars-Erik Pedersen, a senior engineer of NTNU's IT-Department, the group was able to loan a server from the IT-Department. While setting up ESXI on the server, it would not run as intended. After reading the documentation on ESXI, the students learned the server was not compatible with the newest version(6.7) of ESXI. The implementation was put on hold while the server was replaced with a compatible one.

With a new compatible server, the group was able to install ESXI along with vCenter Server Appliance (VCSA) as a virtual machine. With a working server installed running

ESXI and VCSA, it was possible to access more functions like incorporate memory dumping from vCenter Server Appliance and simulating human interaction with the Sandboxes. At this point, the group had a working vSphere Host by allowing VCSA to control and administer the ESXI Hypervisor.

Once the virtualization Environment was working as intended, a server that would monitor and analyze network traffic and perform different analyzes on the sandboxes was next in line.

4.3 Networking

Having set up the Hypervisor, the group needed to implement some subnets in order for the implementation to look like the design in figure 5.

The group set up three virtual network cards on the vSphere host to account for this which can be seen in table 3

Name	**Virtual Machines**	**Subnet**	**vCenter**
Bridge - Internett	2	128.39.142.0/24	128.39.142.137
pfMonitor	2	172.16.1.0/24	128.39.142.137
viruslan	4	192.168.56.0/24	128.39.142.137

Table 3: vSphere Networking

Bridge-Internet makes sure that some of the virtual machines are indirectly connected to the Internet through NTNU Public. Even though these machines are not directly connected to the Internet, it means that other students at NTNU can reach them by using their IP addresses. If Cuckoo used NTNU public, it would mean that users of the network could access Cuckoo and use it by running analysis and filling up the disk space on the vSphere host. pfSense is implemented to protect Cuckoo from direct exposure to NTNU Public. The second virtual machine running with a network card using this virtual switch is vCenter Server Appliance (VCSA). The group decided it would be best to put the VCSA in the same subnet as the ESXI host to avoid having communication problems between the two. The administration of VCSA is also password protected, meaning that the group was willing to take the risk of exposing it in NTNU's public subnet, while also being able to reach it without having to run a VPN. Being able to reach the vCenter Server Appliance server without running a VPN, caused the group to become more flexible in case of pfSense failure which would render the VPN connection useless.

pfMonitor(referred to as Subnet 1 in this thesis) is the Local Area Network (LAN) behind pfSense. The two virtual machines that use this subnet are pfSense and Cuckoo. pfSense routes traffic for Cuckoo. pfSense acts as the gateway for Cuckoo.

viruslan(referred to as Subnet 2 in this thesis) is the Local Area Network (LAN) behind Cuckoo where all of the sandboxes reside. Cuckoo is set up as a gateway for the sandboxes traffic to the Internet, OpenVPN, or TOR.

Name	IP Address	Guest OS
Cuckoo18.04	172.16.1.30 192.168.56.1	Ubuntu 18.04 64-bit
pfSense	128.39.142.158 172.16.1.1	FreeBSD 12 64-bit
vCenter Server Appliance	128.39.142.137	Proton Linux
Win10	192.168.56.5	Windows 10 64-bit
Win8.1	192.168.56.6	Windows 8.1 64-bit
Win7	192.168.56.7	Windows 7 64-bit

Table 4: Network Configuration Virtual Machines

4.4 Software

The first successful Cuckoo installation was done manually following Cuckoo‘s own documentation [22] on an Ubuntu 16.04 LTS (Xenial Xerus). In Cuckoo’s documentation, there are plenty of references to additional software needed to make the most out of Cuckoo’s analysis. Such as Yara, Volatility, Moloch and Elasticsearch [23]. Yara and Volatility had to be downloaded and installed by compiling them. However, after upgrading the server to Bionic Beaver, Volatility ’s latest version was already compiled and ready to install from Ubuntu’s official repositories. The only application that had to be installed by compiling it from source code was Yara. The manual installation went almost seamlessly, and there were some challenges with the version compatibility of some packages; nevertheless, it worked in the end.

Having upgraded to Bionic Beaver, the group was able to install Suricata by adding a Repository which also gave the group the latest version of Suricata [24]. Moloch was installed by heading over to moloch’s download page [25], downloading and installing the latest prebuilt version for Bionic Beaver. The configuration of Cuckoo will be described in the next section.

4.5 Configuration

This section will look into how the platform was configured in this project and will include some of the more important subsections to the configuration of the Playbook, Cuckoo and connecting to the VPN at Digitalocean.

4.5.1 Cuckoo

By default, the Playbook will create a user named cuckoo, and a Home Directory for that user. The Playbook distributes the Configuration Files in this home directory for Cuckoo to use as a working directory. Every time Cuckoo starts up its services, it will read from the configuration files from this home directory. That means, every single time a configuration file has changed, Cuckoo will have to be restarted to reflect these changes. When running the Playbook, Cuckoo will be installed as a systemd service[2]. A systemd service makes it fairly easy to reload new changes to Cuckoo by simply restarting the Cuckoo systemd services, instead of restarting the whole server.

The server is also set up to do basic router functionality, along with Network Address

Translation (NAT). The server is configured with two network cards. One of the network cards sits in the pfSense network, to give the server the Internet connection it needs. The second network card sits in the Sandbox network, where it can reach the sandboxes. The reason the group decided to do this was to make sure all network traffic generated by the sandboxes was intercepted by Cuckoo, and therefore readable.

Due to compatibility issues with Cuckoo and Elasticsearch version 6 [26], the group had to install an older version, namely Elasticsearch version 5. The group downloaded the latest version of Elasticsearch 5, version number 5.6.16, and was able to integrate it with Cuckoo. [27]

4.5.2 Virtual Environment

A Virtual Environment is a directory on the harddrive which installs the packages needed for an application to work. The application in this case, is Cuckoo. In the virtualenv, the administrator can install a specific version of a software to needed to run Cuckoo. In this case, the specific software is python 2.7 and pip 9.0.3.

In addition to installing specific software to the virtualenv, these packets are not updated with the rest of the system, meaning that the application running in this virtualenv is less likely to break because of newer package versions that could stop the app working. The group implemented the virtual environment to the Playbook to future proof the platform, and also to resolve the issues with Cuckoo not being able to run correctly with newer versions of different kinds of software.

4.5.3 OpenVPN

The Playbook will by default include an OpenVPN installation. OpenVPN routes traffic from the sandboxes through a VPN connection to conceal the analyzers location. The OpenVPN server used for this purpose was set up by one of the group members, and it is hosted by Digitalocean[28].

Since this server was only used to test the OpenVPN connection, it is included in a private repo on Bitbucket, because this configuration contains everything needed to connect to the OpenVPN hosted at Digitalocean.

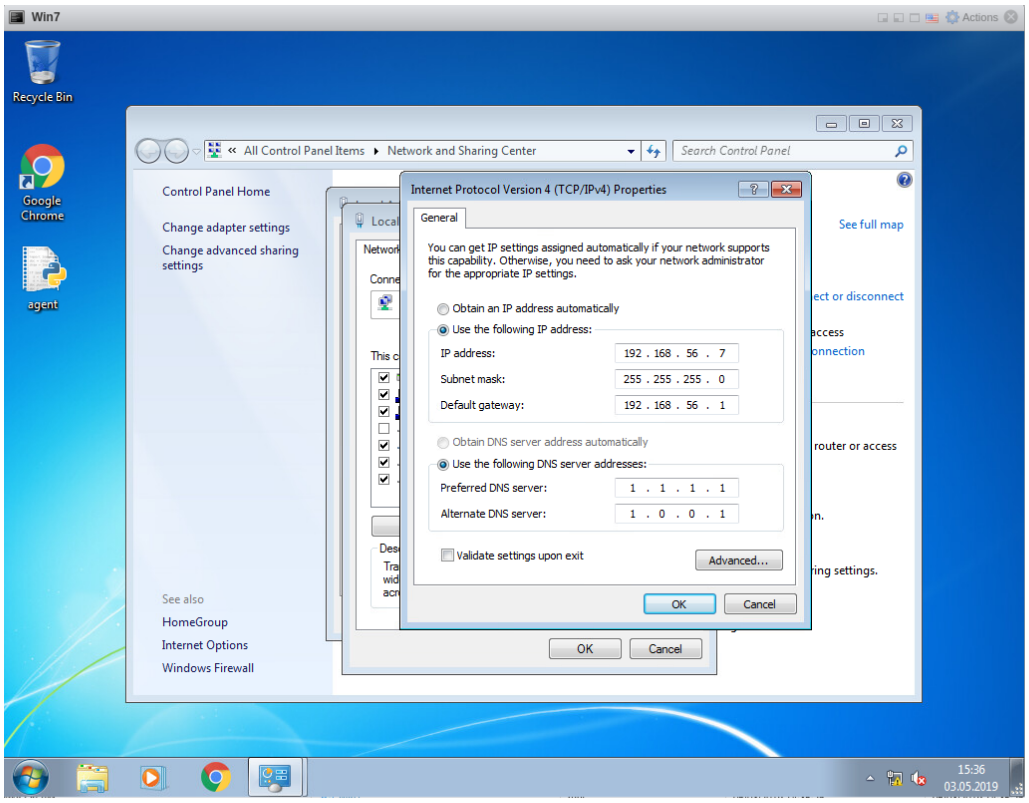
It is set up to create a tun0 interface on the server hosting Cuckoo, so by choosing to route through over VPN in Cuckoo's WebGUI, the Sandboxes will appear to be in France when running analysis. To make this work when connecting to the VPN, the group had to search around for similar issues of how Cuckoo deals with this. A similar issue is described, and the solution was to run a VPN with additional arguments together with the route.py Script (see appendix E and source [29]). The script, along with the complete command, can be found in the source above, by reading doomedraven's 8th comment. Using the same command, the group was able to implement the VPN as a systemd service that starts automatically each time the Bionic Beaver server has been restarted (see appendix F).

4.5.4 Hypervisor

The Hypervisor supported by this Playbook is ESXI and vSphere. Only basic configuration of the ESXI has been done, by creating a datacenter for containtment of the virtual machines running, and three network cards. One for Internet connectivity, one for pfSense network, and one for Sandbox network.

4.5.5 Sandboxes

Three different versions of Windows (7 64-bit, 8.1 64-bit, and 10 64-bit) were installed as virtual machines that would become the Sandboxes. Together with the employer, it was decided to start with only windows hosts to make sure the concept was working. If there were spare time, in the end, the platform would include Android and MacOS. Inside each Sandbox the group installed a couple of programs that were essential for Cuckoo to be able to analyze them. These programs were Python 2.7,Pillow, and an agent for communicating with Cuckoo called agent.py. In order for Cuckoo to be able to communicate with the sandboxes, User Account Control (UAC) and the Windows Firewall was disabled. Additionally, the sandboxes were configured with a static IP configuration, because Cuckoo is known not to be able to communicate with the sandboxes if configured with DHCP. [30]. Figure 6 shows a sample network configuration on one of the sandboxes used in the project.



Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

( ) Obtain an IP address automatically
(•) Use the following IP address:

IP address:	192 . 168 . 56 . 7
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	192 . 168 . 56 . 1

( ) Obtain DNS server address automatically
(•) Use the following DNS server addresses:

Preferred DNS server:	1 . 1 . 1 . 1
Alternate DNS server:	1 . 0 . 0 . 1

☐ Validate settings upon exit

Advanced...

OK Cancel

Figure 8: Network configuration sandboxes

4.5.6 Systemd

The need to implement Cuckoo as a systemd service quickly surfaced as an issue for the group after having installed Cuckoo and needing to restart its services after changing some of the Configuration Files. As such, it was understandable that the employer wanted it implemented as a systemd service, as it became very tedious to restart the server or to kill the PID's every time a change had happened.

Cuckoo Rooter is very special in that it needs every network service to be started beforehand, otherwise it will fail to start. As seen in appendix F and G.1, Cuckoo Rooter needs the OpenVPN service to be started before the Cuckoo Rooter starts, otherwise Cuckoo and all will fail to start. Figure 9 describes the order in which Cuckoo's services and dependencies are started.

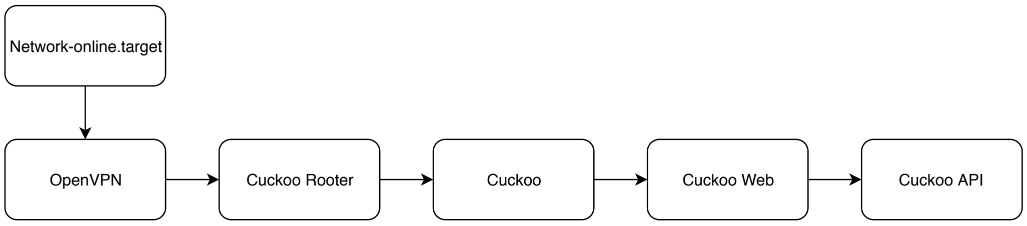


Figure 9: Systemd startup order

4.6 Ansible

When Cuckoo worked as intended after the manual installation, it was time to make the installation process automated with the help of Ansible. The group started reading up on documentation of Ansible and found out that the best way to implement an automatic installation of the platform was to use Playbooks [31]. The reason for using Playbooks is that the designer can include multiple roles to install many applications one by one without requiring any user interaction, thus making the installation process automated.

Note: The Playbook used in creating this platform is based on a similar Playbook that the group found on GitHub. The playbook was over a year old, and very outdated. Having installed Cuckoo manually, the group could implement the steps taken during the manual installation and rewrite the Playbook found on Github. The playbook needed replacement of the old software with newer software versions, also adding new roles to the playbook itself. Because of all the details, and different ways to set up the platform, the group decided to make an entire chapter describing the Playbook and all the different configuration setups in chapter 5.

5 Automation

This chapter covers the documentation of the installation process of Cuckoo. The platform has been successfully installed using the Ansible Playbook developed by the group during this project. The platform runs on an ***Ubuntu 18.04 LTS (Bionic Beaver)*** with the latest updates.

5.1 Prerequisites

Some physical host running ESXI 6.7, managed by vCenter Server Appliance (VCSA) 6.7. Managing the ESXI with VCSA effectively turns the ESXI into vSphere. vSphere is necessary because it allowsCuckoo to retrieve full memory dumps of the virtual machine analyzing a given virus, while also being able to simulate user interactivity. The minimum licence required to run Cuckoo on a VMWare vSphere environment is vSphere Standard. Using the vSphere Essentials licence, which is free of charge, does not include the API support needed by Cuckoo to interact with and control the virtual machines on the vSphere Hypervisor.

During the project, the students found that the server running Cuckoo should have at least 8GB of Ram and 25GB hard drive space. The server also needs to have two Network cards - Subnet 1(Internet connectivity) and one for Subnet 2(Analyzing only).

Do take note that when running Cuckoo with full memory analysis, the disk space needed will increase drastically. The memory dumps are very big in size, and Cuckoo will download these Snapshot memory dumps from the vSphere host itself. The more disk space, the better. An option to delete the memory dumps after processing them can be used to preserve disk space. Using the option to delete the memory dumps will be detailed in section 5.5.6.

5.1.1 Software

Before the installation process begins, make sure that the server is fully updated, and that Python 2.7 is installed. Cuckoo relies heavily on Python 2.7, and will not run with Python 3. Also, make sure that the machine running Ansible has the latest version of Ansible installed or the Playbook might not work.

Before the Playbook can be run, it is vital that some of the files in the Playbook is configured to match the IP-addresses used in the environment. Depending on if its a local installation on a server, or a remote one, two different files can be changed to suit the environment and will be detailed in section 5.2.1 and 5.2.2.

NOTE: All the commands in section 5.2 and on wards are executed in a Bash shell either locally or remotely(through Secure Shell (SSH)) on the Ubuntu 18.04 LTS (Bionic Beaver) server which will be used as a base to install the platform.

5.1.2 Network configuration

The server running Cuckoo needs to have two network cards. The server needs to be configured with a static IP configuration on both network cards in order for Cuckoo to work. In listing 5.1, an example configuration can be seen, and is also the groups network configuration of the server running Cuckoo (etc/netplan/50-cloud-init.yaml):

```
network:
    ethernets:
        ens160:
            addresses:
            - 172.16.1.30/24
            gateway4: 172.16.1.1
            nameservers:
                addresses:
                - 1.1.1.1
                - 1.0.0.1
        ens192:
            addresses:
            - 192.168.56.1/24
            nameservers: {}
    version: 2
```

Listing 5.1: Network Configuration

5.2 Installation of the Platform

To install the platform, make sure to clone the repository from Github [32]. The group has provided a bash script that adds the Ansible repository to the machine used with Ansible. The bash script is only supported on debian-based distributions(e.g. Ubuntu or Linux Mint). Having added the repository and installed Ansible, the Playbook can be configured to the environment. The first file to note is **site.yml** in the directory:

```
cuckoo-playbook/
```

site.yml holds all the roles needed to install Cuckoo.

```
---
- hosts: all
  gather_facts: false
  roles:
   - python
   - cuckoo_user
   - build_directory
   - ubuntu_packages
   - python_packages
   - postgresql
   - elasticsearch
   - cuckoo
   - yara
   - volatility
   - suricata
   - moloch
   - tor
   - inetsim
   - openvpn
   - network
   - start_up_cuckoo
```

Listing 5.2: site.yml

The roles described in **site.yml**(5.2) refer to each folder inside of

```
cuckoo-playbook/roles
```

and the name of each role is case-sensitive.

5.2.1 Local installation

Make sure to have git installed on the server that is going to run Cuckoo, and that the repository [32] is cloned somewhere on that servers' disk. There are a few files that need to be changed in order for the local installation to work.

Step 1

Head into

```
cd cuckoo-playbook/inventories/staging/
```

and open the **hosts** file. Edit the variable **ansible_sudo_pass** to the password of the user, that allows the use **sudo** as depicted in listing 5.3.

```
[all]
localhost ansible_connection=local ansible_sudo_pass=
    password
```

Listing 5.3: Staging hosts

Save and exit the file.

Step 2

Head into

```
cd cuckoo-playbook/inventories/staging/group_vars/
```

and open the **all** file. The **all** file includes variables that are used throughout the entire playbook. The only thing that should need to change here is the variables that hold the names of the Network Interface Cards. Using **ifconfig** or **ip addr** under Ubuntu will tell you the names of each network card. Change them accordingly as depicted in listing 5.4. **nic** is the interface for internet connectivity, and **ananic** is used for the subnet that the Sandboxes reside in.

```
### Network cards ###
nic: ens160
ananic: ens192
```

Listing 5.4: Staging network cards

Save and exit the file.

5.2.2 Remote installation

There are a few files that needs to be changed for the remote installation to work.

Step 1

Head into

```
cd cuckoo-playbook/inventories/production/
```

and open the **hosts** file. The **ansible_host** variable is the IP-address of the server you wish to install the platform to. The **ansible_ssh_user** variable is the user that Ansible will use to log into the target server. The only requirement for this user, is that it has sudo rights. The **ansible_ssh_pass** variable is the password to the **ansible_ssh_user**. Edit them accordingly, and save and exit the file when done. Listing 5.5 depicts an example where the variables have been filled in.

```
[all]
remotehost ansible_host= ansible_ssh_user=user
    ansible_ssh_pass=password ansible_sudo_pass=
    sudoPassword
```

Listing 5.5: Production host

Note: Some Linux distributions allows a user to use a different password for elevating to and running commands as **sudo**. With a standard installation of Ubuntu server, the password used to log in as a given user, is the same as the password used to elevate **sudo** rights.

Step 2

Head into

```
cd cuckoo-playbook/inventories/production/group_vars
```

and open **all** file. As with the local installation, the only variable that needs to be edited is the network card variables. Make sure they reflect the names the of network cards on the server used to install the platform onto. Example Network Interface Cards can be seen in figure 5.6.

```
### Network cards ###
nic: ens160
ananic: ens192
```

Listing 5.6: Production network cards

Save and exit the file.

5.3 OpenVPN

This section will include the steps needed to be taken when either disabling 5.3.1 or enabling 5.3.2 OpenVPN role. Note that it is required to have a working external OpenVPN server for this role to work. To be able to fully test the functionality of the OpenVPN, one of the group members bought a server at Digitalocean [33].

NOTE: To be able to start Cuckoo after the Playbook has installed the platform, some steps need to be considered whether or not the Playbook should be installed. The Playbook is installed in such a way that it requires an OpenVPN to be running before Cuckoo can even start its services. The steps needed to both enable and disable the role completely will be better explained in sections 5.3.1 and 5.3.2.

5.3.1 Disabling the role

A side note when disabling the OpenVPN role, some more work needs to be done to fully disable it from the setup. Commenting it out in **site.yml** is not enough.

Step one

Head into

```
cd cuckoo-playbook/roles/start_up_cuckoo/files/
```

in this folder, open the **cuckoo_rooter.service** file, and remove openvpnclient.service on line 3.

```
1  [Unit]
2  Description=Cuckoo Rooter Service
3  After=network.target openvpnclient.service
```

Listing 5.7: cuckoo_rooter.service

Save the file and exit.

Step two

Head into

```
cd /cuckoo-playbook/roles/cuckoo/files
```

In this folder, open the **routing.conf** file, and change **enabled=yes** to **enabled=no** on line 61 as seen in listing 5.8.

```
59  [VPN]
60  # Are VPNs enabled?
61  enable = yes
```

Listing 5.8: routing.conf

Save the file and exit.

5.3.2 Enabling the role

The role is by default enabled for installation, but it will still require you to change some files around to make it fit your setup. Head into

```
cd cuckoo-playbook/roles/openvpn/files/
```

and open the **openvpn.conf** file. The easiest way to do this, is to replace the contents in this file with the contents of an **ovpn** file that is already in your possession(assuming you have a working openvpn server already). The only vital part about this, is that the interface created by running the configuration file, **must** be named *tun0* as depicted in listing 5.9. More on this in a later section.

```
23  ;dev tap
24  dev tun0
```

Listing 5.9: openvpn.conf

NOTE: *In the configuration file used in this project, the exact line number is 24, but it could be different depending on the OpenVPN server configuration.*

5.4 InetSim

If using an alternative to, and therefore do not want to install INetSim to the platform, simply comment out the inetsim line from site.yml, and no additional configuration will be needed for INetSim to be disabled upon running the Playbook.

5.5 Cuckoo

This section will cover every file used by Cuckoo that will have to be edited before running the Playbook. Do note that these files can be edited after the platform has been installed, but editing them beforehand saves a little bit of hassle.

This section will not go into detail about every single one of these files, but only the ones used in the making of this project. The files not detailed are used for either VirtualBox (virtualbox.conf), KVM (kvm.conf), XenServer (xenserver.conf), and QEMU (qemu.conf) as alternative Hypervisors to ESXI or vSphere. Other files not detailed are avd.conf (Used for configuration of Android emulator) and configuration of analyzes on physical machines(physical.conf).

5.5.1 cuckoo.conf

This section will look at some important variables inside the cuckoo.conf file found in:

```
cd cuckoo-playbook/roles/cuckoo/files/
```

Listing 5.10: cuckoo.conf directory

All of the files that needs to be edited can be found in the directory shown in listing 5.10.

Machinery

In the **cuckoo.conf** file is where one specifies which Hypervisor one would like to use, along with other options like memory dumps. Depending on whether one want to use ESXI or vSphere, go to the line depicted in listing 5.11

```
20 machinery = vsphere
```

Listing 5.11: cuckoo.conf - Machinery

Simply changing **vsphere** to **esx** changes the chosen Hypervisor to ESXI from vSphere. Machinery is by default set to **vsphere**.

Refer to 5.5.2 for more information on **vsphere.conf**. If one want to use ESXI instead, refer to 5.5.3.

Line 25 allows one to choose whether or not one wants to enable memory dumps, by downloading the memory dumps from the Hypervisor after the analysis is done. Refer to listing 5.12.

```
25 memory_dump = yes
```

Listing 5.12: cuckoo.conf - Memory dumps

Simply changing the value to ’no’ turns off memory dumps. Memory dumps can still be turned on in the WebGUI when running an analysis. The options available can be seen in appendix J.

Resultserver

The variable on line 87 describes which IP-address the sandboxes should interact with. In this case, it will be the same IP-address as the platform running Cuckoo as seen in figure 5.

```
87  ip = 192.168.56.1
```

Listing 5.13: cuckoo.conf - Resultserver IP Address

Processing

Line 108 allows the analyst to resolve DNS lookups performed by the Sandboxes during analysis.

```
107  resolve_dns = yes
```

Listing 5.14: cuckoo.conf - Processing

The resolved DNS lookups can be seen in figure 14.

Database

Line 122 through 127 includes the configuration of which databases to use for analysis. Cuckoo will by default use a PostgreSQL database to store analyzed results. However, Cuckoo supports multiple databases. Listing 5.15 shows some examples of which databases that are supported.

```
122  # Examples, see documentation for more:
123  # sqlite:///foo.db
124  # postgresql://foo:bar@localhost:5432/mydatabase
125  # mysql://foo:bar@localhost/mydatabase
126  # If empty, defaults to a SQLite3 database at $CWD/
         cuckoo.db.
127  connection = postgresql://cuckoo:cuckoo@localhost
         :5432/cuckoo
```

Listing 5.15: cuckoo.conf - Databases

Using PostgreSQL, allows Cuckoo to do multiple analysis simultaneously, and it is therefore default here.

Remote control

While the analysis is running, one can choose to remotely control the Sandboxes with a tool called **Guacd**. **Guacd** is installed by the **ubuntu_packages** role, and is enabled by default in **cuckoo.conf**. Lines 145 through 153 include the configuration of this service in listing 5.16. **NOTE**: To be able to remote control Windows Sandboxes, **VNC** will have to be installed on them, otherwise it will not work.

```
145  [Remote Control]
146  # Enable for remote control of analysis machines
         inside the web interface.
147  enabled = yes
148
149  # Set host of the running guacd service.
150  guacd_host = 192.168.56.1
```

```
151
152 # Set port of the running guacd service.
153 guacd_port = 4822
```

Listing 5.16: cuckoo.conf - Remote control

The **guacd_host** variable(line 150) must be set to the same IP-address as Cuckoo has in Subnet 2 as seen in figure 5.

5.5.2 vsphere.conf

The first lines one need to look at here are the lines 14 through 17. These lines hold the configuration parameters to vSphere through the use of an IP-address, a username and a password.

```
14 host = ip.address.to.vcsa
15 port = 443
16 user = username@some.domain
17 pwd = Passw0rd
```

Listing 5.17: vsphere.conf - Connection

Unfortunately, these parameters are not enough to make sure Cuckoo is able to interact with the vSphere server. The parameters are not enough because in later versions of python(developed after the last version of Cuckoo), it has enabled SSL/TLS connections(PEP-0476) by default because of a security issue described here [34]. Appendix H describes how the group managed to work around this issue by using SSL instead of TLS. The patched version of **vsphere.py** is installed by the Playbook, so that the connection between vSphere and Cuckoo works after the installation has finished. Do note that this should only be necessary when using a self-signed certificate specified at line 32 in listing 5.18:

```
32 unverified_ssl = yes
```

Listing 5.18: vsphere.conf - Certificate

Moving on with the configuration file, the Sandboxes that are hosted by the vSphere can be defined. Line 22 describes this in further detail, but names in this file must match the names of the virtual machines specified on the vSphere host. Listing 5.19 shows the groups configuration.

```
22 machines = Win10, Win81,Win7
```

Listing 5.19: vsphere.conf - Sandboxes

The configuration of the Windows 7 Sandbox which the group used for most of its testing can be seen in appendix I. The configuration of Windows 8.1 and Windows 10 is identical to the sample, except for **ip =** and **osprofile =**. Both of these variables are found at lines 107 and 117, respectively, in the sample configuration. The **snapshot =** variable can be somewhat confusing. What it means is that it wants the name of the Snapshot(on the Hypervisor) of the virtual machine when the agent is running. The group used ***clean_agent*** as a reference to the rightful state of the virtual machine, but you can use any name that makes sense to you.

Lastly, line 27 specifies the name of the network card on the Cuckoo server that is connected to Subnet 2 and is used for intercepting network traffic generated by the Sandboxes. The name of this network card will be the same as the one described in section 5.2.2 (**ananic**).

```
27 interface = ens192
```

Listing 5.20: vsphere.conf - Network card

Save the file and exit.

5.5.3 esx.conf

Very similar to **vsphere.conf**, but the IP-address, username and password will have to be different, because vSphere/VCSA is a different server than the ESXI server. Defines Sandboxes the same way as done in **vsphere.conf**.

5.5.4 limits.conf and sysctl.conf

Running multiple analysis at once, might cause Cuckoo to stop and crash because it has reached the limit of too many open files at once. To avoid this, two files need to be edited in order to increase the open file limit. The group included a way to increase the open file limit to the Playbook. **NOTE:** The group never actually hit the open file limit once during the testing of the platform, but was done to future proof the platform for the employer. The issue is described here [35].

The fix was implemented by appending some lines of configuration to both **/etc/security/limits.conf** (5.21) and **/etc/sysctl.conf** (5.22) [36]. Both of these files have been included to the Playbook and will be installed by default by the cuckoo role.

limits.conf

Looking at the **limits.conf** file, line 56 through 59 was implemented to overcome this issue.

```
56 *           hard        nofile        500000
57 *           soft        nofile        500000
58 root        hard        nofile        500000
59 root        soft        nofile        500000
```

Listing 5.21: limits.conf - Open file limit

sysctl.conf

Line 63 of **sysctl.conf** is the line that was implemented to fix the issue.

```
63 fs.file-max = 2097152
```

Listing 5.22: sysctl.conf - Open file limit

5.5.5 memory.conf

memory.conf handles all the tuning options for Volatility. **memory.conf** is the kind of file where you would want to enable some parameters by changing **enabled = no** to **enabled = yes**.

One very useful option for environments that does not have much harddrive space is to enable the option to delete memory dumps after processing.

```
# Basic settings
[basic]
# Profile to avoid wasting time identifying it
guest_profile = WinXPSP2x86
# Delete memory dump after volatility processing.
# Change this to yes to avoid running out of space
delete_memdump = no
```

Listing 5.23: memory.conf - File processing

Other than that, the options that are enabled in **memory.conf** already, are what provided the group with the best results during testing and is therefore enabled by default when installing the Playbook.

5.5.6 processing.conf

Processing.conf handles everything from behavior analysis to hits of Malware signature on VirusTotal.

Most of the sections here are self-explanatory, while some will have to be explained in deeper detail. Among the few that do, is **[procmemory]**. **[procmemory]**'s section creates process memory dumps for each analyzed process, right before they terminate themselves or right before the analysis finishes. The group enabled it for better analysis, and an extra option here is to delete the process memory dump after analysis to save disk space. On low hard drive systems, this is recommended, because this quickly eats up disk space. Memory dump can be enabled on line 87 of **processing.conf**:

```
86 # Delete process memory dumps after analysis to save
       disk space.
87 dump_delete = no
```

Listing 5.24: processing.conf - Process memory dumps

Other options include testing **apk**'s. **APK**'s is the file-extension of every Android app that exists out there. Since Android will not be tested here, it is therefore disabled by default. The option to use Suricata is also enabled here. Lines 122 through 142 includes options for Suricata. Since Suricata is only setup to listen in on traffic, and later used to analyze the traffic, none of these options need to change from this project to deployment on yours. It is possible to setup Cuckoo to increase performance here by letting Cuckoo talk directly to the Suricata socket, which can be found at line 142.

```
142 socket =
```

Listing 5.25: processing.conf - Suricata

5.5.7 reporting.conf

This configuration file holds all the necessary IP-addresses for each component that runs alongside Cuckoo. These components include things like MongoDB, Moloch and Elastic-search.

Lines 34 through 43 hold the configuration for MongoDB:

```
34 [mongodb]
35 enabled = yes
36 host = 127.0.0.1
37 port = 27017
38 db = cuckoo
39 store_memdump = yes
40 paginate = 100
41 # MongoDB authentication (optional).
42 username =
43 password =
```

Listing 5.26: reporting.conf - MongoDB

Elasticsearch can be configured on lines 45 through 60. The free version of Elasticsearch is wide open, so there is no option here to configure a username or a password:

```
45 [elasticsearch]
46 enabled = yes
47 # Comma-separated list of ElasticSearch hosts. The
       format is IP:PORT, if the port is
48 # missing the default port is used.
49 # Example: hosts = 127.0.0.1:9200, 192.168.1.1:80
50 hosts = localhost:9200
51 # Increase default timeout from 10 seconds, required
       when indexing larger
52 # analysis documents.
53 timeout = 300
54 # Set to yes if we want to be able to search every API
        call instead of just
55 # through the behavioral summary.
56 calls = no
57 # Index of this Cuckoo instance. If multiple Cuckoo
       instances connect to the
58 # same ElasticSearch host, then this index (in Moloch
       called "instance") should
59 # be unique for each Cuckoo instance.
60 index = cuckoo
```

Listing 5.27: reporting.conf - Elasticsearch

Running Elasticsearch on the same server as Cuckoo will require no changing of parameters. Elasticsearch will work as-is, if one wants to run Elasticsearch on a different server, it is required to change line 50 of **reporting.conf**. Change it from **localhost:9200** to whatever IP-address it runs on, but do not forget the *:9200* at the end.

Lines 71 through 78 cover Moloch's installation:

```
71 [moloch]
72 enabled = yes
73 # If the Moloch web interface is hosted on a different
        IP address than the
74 # Cuckoo Web Interface then you'll want to override
       the IP address here.
75 host = 172.16.1.30
```

```
# If you wish to run Moloch in http (insecure) versus https (secure) mode,
# set insecure to yes.
insecure = yes
```

Listing 5.28: reporting.conf - Moloch

The host in this section is the same host as Cuckoo and will therefore have its IP-address set to the same as the one mentioned in Subnet 1 and in figure 5.

5.5.8 routing.conf

This file is a big one, and you are able to configure all the routing by Cuckoo here. There are a few small variables that need to change in this file. The first one is the configuration of **InetSim**, depending on your network setup:

```
[inetsim]
enabled = yes
server = 192.168.56.1
```

Listing 5.29: routing.conf - Inetsim

Change the variable **server** to the IP-address you need it to be. Make sure that it matches the configuration of your network setup, use figure 5 as a reference point, but remember that this IP-address needs to be in Subnet 2.

The second variable is the name of the network card handling the **Internet** routing. The variable is located at line 19:

```
19 internet = ens160
```

Listing 5.30: routing.conf - Network Interface Card (NIC)

This should be changed to be the same network card that as used in Subnet 1.

TOR can also be somewhat configured in this file, but TOR is also configured in **torrc**, which we will return to in section 5.8. A few lines still needs to be mentioned here, because it is imperative that it matches the configuration in **torrc**:

```
52 [tor]
53 # Route a VM through Tor, requires a local setup of Tor (please refer to our
54 # documentation).
55 enabled = yes
56 dnsport = 5353
57 proxyport = 9040
```

Listing 5.31: routing.conf - TOR routing

The ports configured in line 56 and 57 must match the ports in **torrc**.

5.6 Elasticsearch

The Elasticsearch role is a rather simple one and only requires you to change an IP-address in its configuration file in order to work properly. The configuration file in question is **elasticsearch.yml**, and is found under.

```
cuckoo-playbook/roles/elasticsearch/files/
```

The line we need to edit is found in the yml file at line 55:

```
55 network.host: 172.16.1.30
```

Listing 5.32: elasticsearch.yml

Make sure that this IP-address matches the one in Subnet 1. Use listing 5.1.2 as a reference.

5.7 Suricata

This section covers all of Suricata's files that need to be edited depending on your network configuration. All files are located under

```
cuckoo-playbook/roles/suricata/files/
```

5.7.1 suricata

This file has only one line that needs to be changed, and is once again dependant on Network Interface Card (NIC) naming policy of the linux kernel. The line we need to change is 19 as listing 5.33 describes further:

```
18 IFACE=ens192
```

Listing 5.33: suricata

5.7.2 suricata.yaml

The last file to edit for Suricata to be fully functional is **suricata.yaml**. On line 15, IP-addresss for address groups is required for better performance and accuracy:

```
13 # more specifc is better for alert accuracy and performance
14 address-groups:
15   #HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"
16   HOME_NET: "[192.168.56.0/32]"
```

Listing 5.34: suricata.yaml

Make sure the address groups fits subnet's correctly.

5.8 Tor

torrc is a file that is used by TOR for its configuration and is located in:

```
cuckoo-playbook/roles/tor/files/
```

For this project and its features, it is enough for the group to look at lines 15 and 16 of the file:

```
15 TransPort 192.168.56.1:9040
16 DNSPort 192.168.56.1:5353
```

Listing 5.35: torrc

Make sure that the IP-address here matches the one configured at ens192 in listing 5.1. Also do make sure that the port number is the same as mentioned in listing 5.31. Otherwise will not work.

5.9 Cuckoo Startup

This role is a rather simple one. All it does is copy four preconfigured systemd files to **/etc/systemd/system**. The idea with this is that you can restart Cuckoo's services(rooter, cuckoo itself, web and api) without having to restart the entire server or kill the services with PID's every time you change a configuration file. Each one of the systemd services can be more thoroughly viewed through appendixes G.1, G.2, G.3 and G.4. These also be started in the correct order for Cuckoo to work. For example, if running a VPN, the Cuckoo Rooter must be the first service to start. But the Cuckoo Rooter also handles Internet routing, Tor and InetSim as well. Which is why the **openvpnclient.service** (appendix F) MUST be disabled in **cuckoo_rooter.service** before running the Playbook to avoid any start up issues.
Note: Appendixes G.3 and G.4 also have IP-addresses in them. Change these addresses to reflect your configuration.

5.10 Running the playbook

After having completed the previous steps, the playbook may be run with the following command from the terminal inside the **cuckoo-playbook** directory:

```
ansible-playbook -i inventories/production site.yml
```

Listing 5.36: Running the Playbook

Depending on whether the Playbook is run on a remote or a local server, replace ***production*** with ***staging***.

5.11 Final thoughts

Most of the steps described in the previous sections can be omitted if Cuckoo's Subnet 2 is defined with the same IP-addresss the group used. The IP-addresss of a production environment might not be the same as the ones the group used, and will therefore have to be changed.

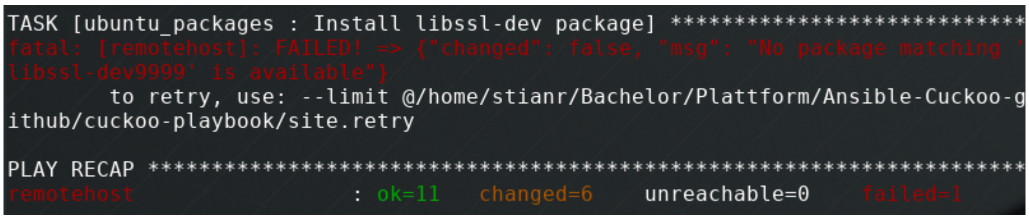
To avoid manually changing all of these IP-addresss, the group has included a section in chapter 7 under alternative approaches and future work (7.2.4) which would make installation more automated.

6 Testing and verification

Testing is an essential step to point out errors made during the implementation phase and to deliver a quality project. That is why the group focused on making tests that made sure everything worked as intended before it was delivered to the employer.

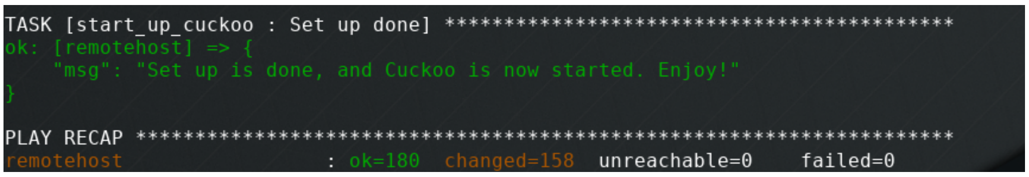
6.1 Automation

To ensure that the Playbook worked without any input from the user, the group ran the playbook on a fresh server after each test by reverting the server to an earlier Snapshot. When an error occurred, the students would analyze the error message and correct the issue in the Playbook as seen in figure 10. The group repeated this test until there were no errors in the Playbook's installation process, as shown in figure 11.



```
TASK [ubuntu_packages : Install libssl-dev package] ****************************
fatal: [remotehost]: FAILED! => {"changed": false, "msg": "No package matching '
libssl-dev9999' is available"}
        to retry, use: --limit @/home/stianr/Bachelor/Plattform/Ansible-Cuckoo-g
ithub/cuckoo-playbook/site.retry

PLAY RECAP *********************************************************************
remotehost                 : ok=11   changed=6    unreachable=0    failed=1
```

Figure 10: Ansible test with errors



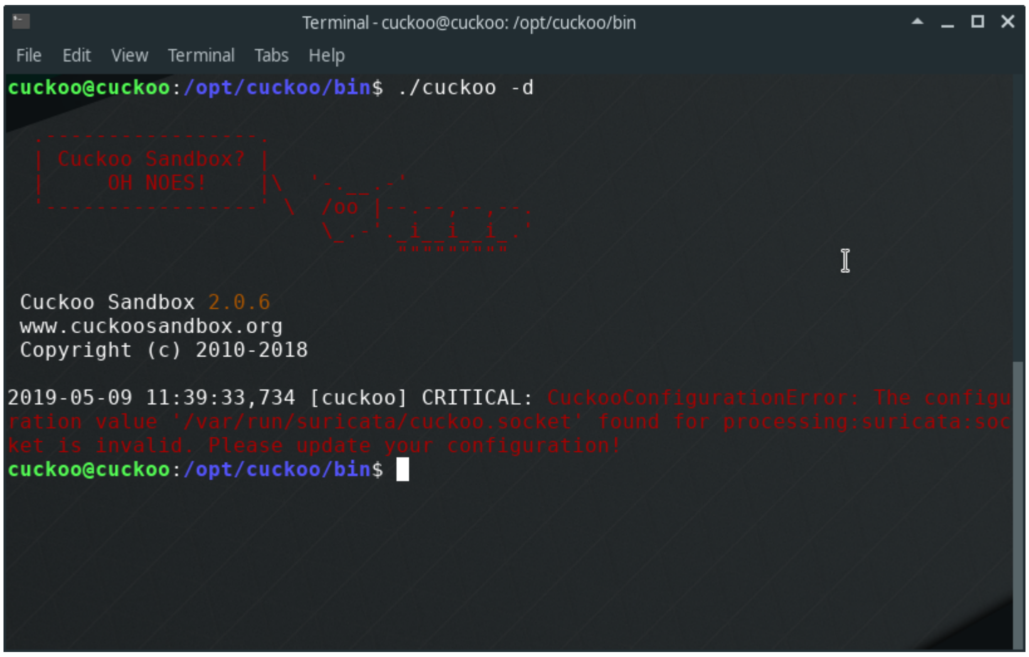
```
TASK [start_up_cuckoo : Set up done] *******************************************
ok: [remotehost] => {
    "msg": "Set up is done, and Cuckoo is now started. Enjoy!"
}

PLAY RECAP *********************************************************************
remotehost                 : ok=180  changed=158  unreachable=0    failed=0
```

Figure 11: Ansible test without errors

While the group was testing the playbook, it came to the group's attention that Moloch stopped the installation of the playbook, requiring input from the user for the Playbook to continue installing the platform. After discussing this issue with the group's employer, it was agreed that this needed to be changed. To remedy this issue, the group added a configuration file that adds a few parameters which result in Moloch not requiring user input.

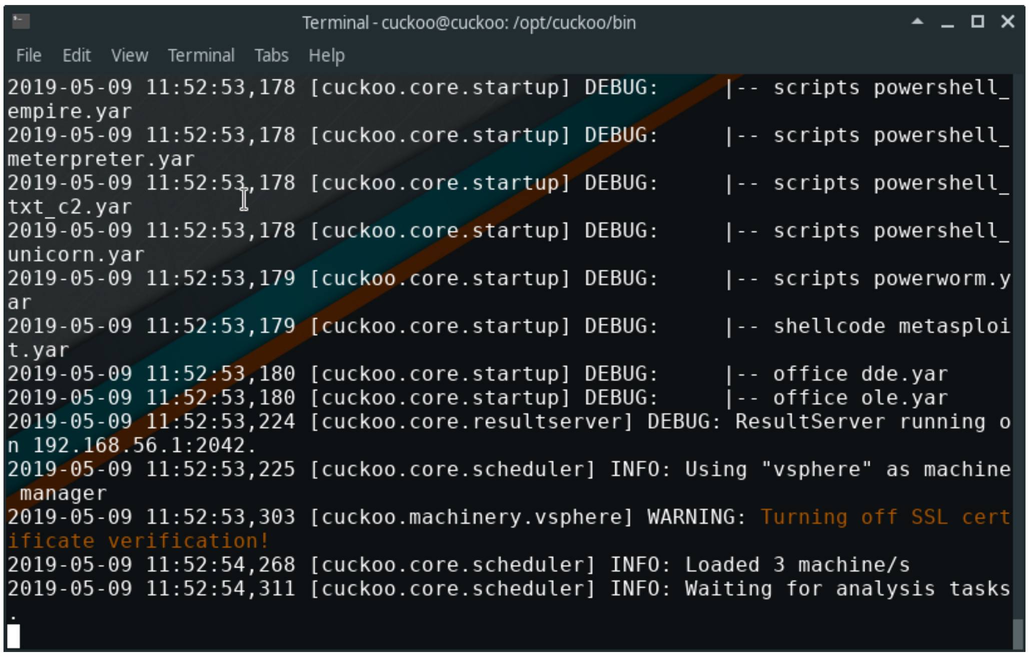
6.2 Cuckoo

To test if Cuckoo was configured correctly the group ran **./cuckoo -d** as shown in figure 12. Running Cuckoo with its debug parameter, the group was able to find any configuration problems by using the error messages. Moreover, fixing all the errors from Cuckoo's output, allowed Cuckoo to start and wait for an analysis to be submitted, a demonstration can be seen in figure 13.



```
cuckoo@cuckoo:/opt/cuckoo/bin$ ./cuckoo -d

  .-----------------.
  | Cuckoo Sandbox? |
  |     OH NOES!    |\ '-.__.-'
  '-----------------' \ /oo |--.--,--,--.
                        \_.-'._i__i__i_.'
                              """""""""

 Cuckoo Sandbox 2.0.6
 www.cuckoosandbox.org
 Copyright (c) 2010-2018

2019-05-09 11:39:33,734 [cuckoo] CRITICAL: CuckooConfigurationError: The configu
ration value '/var/run/suricata/cuckoo.socket' found for processing:suricata:soc
ket is invalid. Please update your configuration!
cuckoo@cuckoo:/opt/cuckoo/bin$ 
```

Figure 12: Cuckoo test with errors

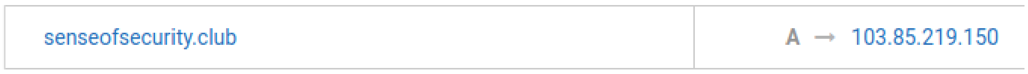


```
2019-05-09 11:52:53,178 [cuckoo.core.startup] DEBUG:     |-- scripts powershell_
empire.yar
2019-05-09 11:52:53,178 [cuckoo.core.startup] DEBUG:     |-- scripts powershell_
meterpreter.yar
2019-05-09 11:52:53,178 [cuckoo.core.startup] DEBUG:     |-- scripts powershell_
txt_c2.yar
2019-05-09 11:52:53,178 [cuckoo.core.startup] DEBUG:     |-- scripts powershell_
unicorn.yar
2019-05-09 11:52:53,179 [cuckoo.core.startup] DEBUG:     |-- scripts powerworm.y
ar
2019-05-09 11:52:53,179 [cuckoo.core.startup] DEBUG:     |-- shellcode metasploi
t.yar
2019-05-09 11:52:53,180 [cuckoo.core.startup] DEBUG:     |-- office dde.yar
2019-05-09 11:52:53,180 [cuckoo.core.startup] DEBUG:     |-- office ole.yar
2019-05-09 11:52:53,224 [cuckoo.core.resultserver] DEBUG: ResultServer running o
n 192.168.56.1:2042.
2019-05-09 11:52:53,225 [cuckoo.core.scheduler] INFO: Using "vsphere" as machine
 manager
2019-05-09 11:52:53,303 [cuckoo.machinery.vsphere] WARNING: Turning off SSL cert
ificate verification!
2019-05-09 11:52:54,268 [cuckoo.core.scheduler] INFO: Loaded 3 machine/s
2019-05-09 11:52:54,311 [cuckoo.core.scheduler] INFO: Waiting for analysis tasks
.
```

Figure 13: Cuckoo test without errors

The message ***INFO: waiting for analysis tasks*** in figure 13 means that all the modules were loaded successfully and Cuckoo is just waiting for the user to submit malware for analysis.

When an analysis has completed, Cuckoo is set up to resolve all DNS requests made by the Sandboxes. Figure 14 shows how this looks like in Cuckoos report in the WebGUI:

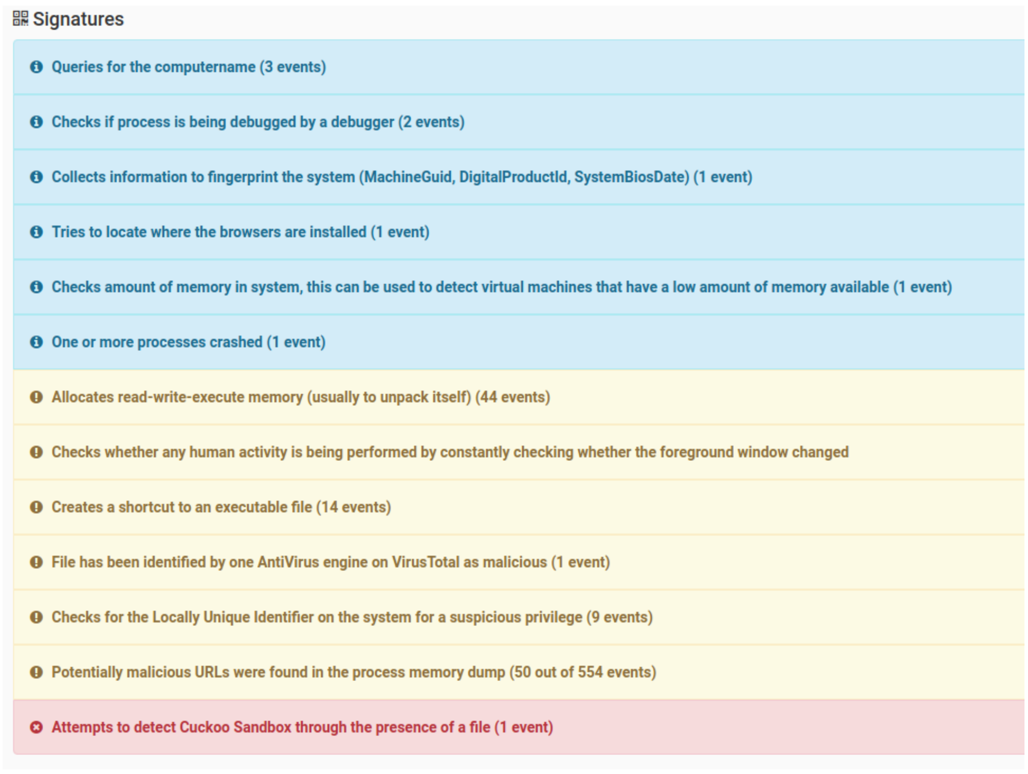


senseofsecurity.club | A → 103.85.219.150

Figure 14: Resolved DNS requests done by Cuckoo

6.3 Malware

To test Cuckoo's functionality, the group ran a variety of different Malware. The different malware was tested so Cuckoo could analyze them, and then the group could see if the results from the Cuckoo analysis matched the expected outcome.



Signatures

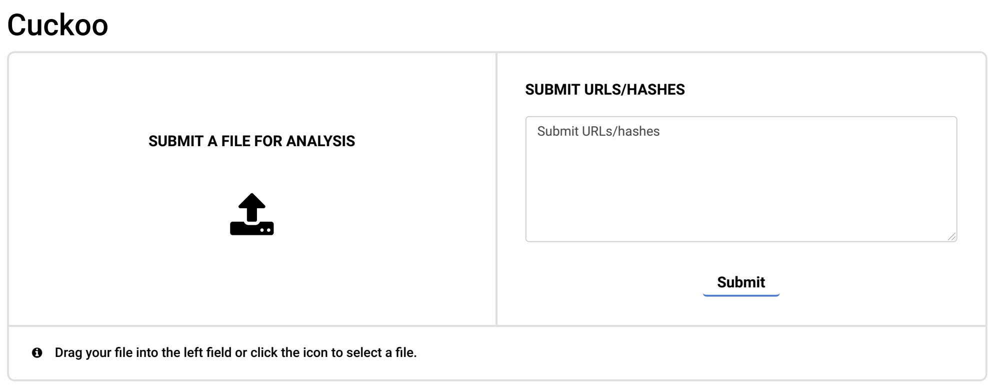
- Queries for the computername (3 events)
- Checks if process is being debugged by a debugger (2 events)
- Collects information to fingerprint the system (MachineGuid, DigitalProductId, SystemBiosDate) (1 event)
- Tries to locate where the browsers are installed (1 event)
- Checks amount of memory in system, this can be used to detect virtual machines that have a low amount of memory available (1 event)
- One or more processes crashed (1 event)
- Allocates read-write-execute memory (usually to unpack itself) (44 events)
- Checks whether any human activity is being performed by constantly checking whether the foreground window changed
- Creates a shortcut to an executable file (14 events)
- File has been identified by one AntiVirus engine on VirusTotal as malicious (1 event)
- Checks for the Locally Unique Identifier on the system for a suspicious privilege (9 events)
- Potentially malicious URLs were found in the process memory dump (50 out of 554 events)
- Attempts to detect Cuckoo Sandbox through the presence of a file (1 event)

Figure 15: Signatures of Trojan.Heur.FU.dmW@a8VPjLb tested in Cuckoo

After having tested a Malware and Cuckoo presented the results, it did not seem like a test had happened; therefore, the group suspected something was wrong at some point in the analyzing process. With Cuckoo reporting no suspicious activity, it could be an indicator that one or more of the applications did not work as intended. Cuckoo displays screenshot from the events on the Sandboxes with the help of Pillow, and the issue was often found when the group into them. If the screenshots did not provide any useful information, then the log files would.

6.4 Network

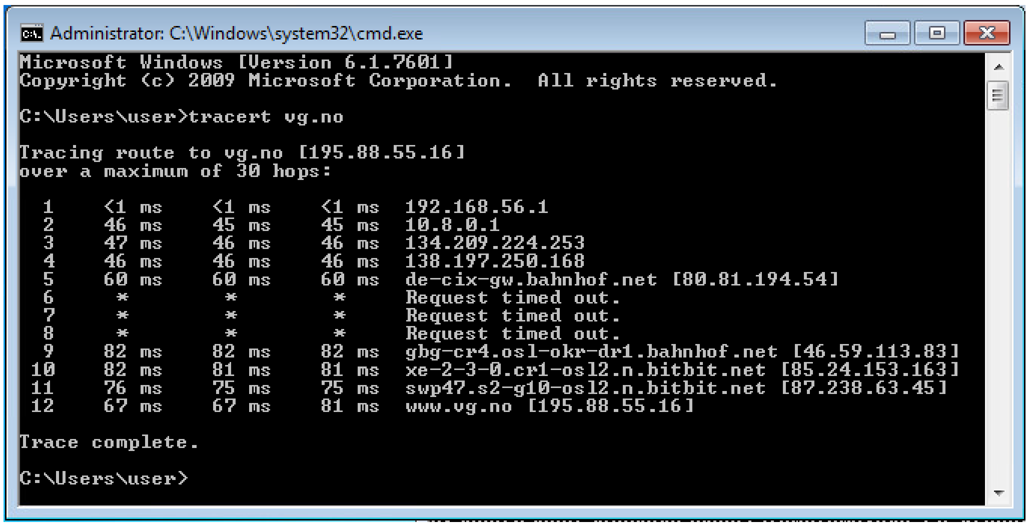
To test if the Sandboxes had internet access without having to log into each one and trying to reach a website with every routing options, the group used Cuckoo's Routing tool Cuckoo Rooter. When writing a website URL in the "SUBMIT URLS/HASHES" field (see figure 16) cuckoo tries to connect to the website from each sandbox. Cuckoo will not only start the browser but will also attempt to actively instrument it in order to extract interesting results such as executed Javascript, iframe URLs, etc[22]. The test was executed with each routing option VPN, TOR, INetSim and Internet. These options can be seen in appendix J.



Drag your file into the left field or click the icon to select a file.

Figure 16: Cuckoo Web GUI

In order to test if Cuckoo is Routing the data packets correctly the group made a Traceroute Script. Submitting this file for analysis and choosing a network service to route through(Internet, VPN). The Traceroute script traces what path the data packets took towards the Internet and helped the group fix routing issues too.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\user>tracert vg.no

Tracing route to vg.no [195.88.55.16]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  192.168.56.1
  2    46 ms    45 ms    45 ms  10.8.0.1
  3    47 ms    46 ms    46 ms  134.209.224.253
  4    46 ms    46 ms    46 ms  138.197.250.168
  5    60 ms    60 ms    60 ms  de-cix-gw.bahnhof.net [80.81.194.54]
  6     *        *        *     Request timed out.
  7     *        *        *     Request timed out.
  8     *        *        *     Request timed out.
  9    82 ms    82 ms    82 ms  gbg-cr4.osl-okr-dr1.bahnhof.net [46.59.113.83]
 10    82 ms    81 ms    81 ms  xe-2-3-0.cr1-osl2.n.bitbit.net [85.24.153.163]
 11    76 ms    75 ms    75 ms  swp47.s2-g10-osl2.n.bitbit.net [87.238.63.45]
 12    67 ms    67 ms    81 ms  www.vg.no [195.88.55.16]

Trace complete.

C:\Users\user>
```

Figure 17: Traceroute script over VPN

6.5 Sandbox

To test the compatibility between Cuckoo and the Operating System (OS)'s running in the different sandboxes. The group tested the same malware on the three different sandboxes(win7,win8.1, and win 10). The results of the analysis were different in all three instances; this could be because as cuckoos own documentation states[37], Windows 7 is the recommended OS for the Sandbox. Since Windows 7 is older than both Windows

8.1 and Windows 10, it is believed that it does not possess all of the security mechanisms newer versions of windows have [38]. Another reason for the difference in results in these analyses could be that Cuckoo is more compatible with Windows 7.

7 Discussion

In this chapter, the reader will make sense of the difficulties for the students along the way. The group will discuss alternative approaches that also could be used for future work. In the end, the group will present self-criticism then evaluate the groups' effort.

7.1 Obstacles

This section will describe the issues encountered and how the group resolved these issues.

7.1.1 Playbook

The Playbook the group used to successfully install Cuckoo the first time, was based on a Playbook that the group found on GitHub. [39]. The Playbook was used as a backbone, because at the point of its creation it installed Cuckoo successfully on an Ubuntu 16.04 LTS (Xenial Xerus) server. Due to the Playbook being over a year old, the group updated it to work with packets that were newer and compatible.

During a meeting with the employer, the group discussed the matter of running the platform on Ubuntu 16.04 LTS. The employer suggested and wanted that the platform should be updated to run on Ubuntu 18.04 LTS. Updating the platform to Ubuntu 18.04 LTS meant that the group had to rework the Playbook. After doing some research on updating the Playbook, the group quickly realized that most of the packages that the Playbook previously had downloaded and compiled from scratch were no longer needed, because the official Bionic Beaver repositories already had most of the prior compiled packages. Some of the roles had to be rewritten to reflect that some of the software did not have to be downloaded, compiled, which made it easier to install the given packages. The group used an evening to rewrite the Playbook to make it work on Bionic Beaver.

7.1.2 vCenter Server Appliance (VCSA)

During one of the meetings where the group had prepared a demo for the employer, it became clear that the virtual machine running vCenter Server Appliance (VCSA) had crashed. The crash caused the group not to be able to show a demo to the employer.

After some research, the group found out that the version of vCenter Server Appliance (VCSA) had the wrong compatibility with ESXI, which caused VCSA to crash. The version of VCSA had been running version 6.5, while the ESXI was running version 6.7. The group quickly decided to download and install a compatible version of VCSA to match the ESXI Hypervisor to solve this issue permanently. Having reinstalled VCSA to run version 6.7 it has been stable, and not a single crash has occurred ever since.

7.1.3 Sandboxes

In addition to VCSA, the group had additional difficulties regarding the Sandboxes. During the execution of an analysis on either Windows 8.1 or Windows 10, Cuckoo's agent stopped communicating with Cuckoo. The group never really found the root issue of

Cuckoo's agent stopping to communicate with Windows 8.1 and 10. Restarting the agent, creating a new Snapshot and running a further analysis only provides a temporary solution, because the issue reappears after running one analysis on those two Sandboxes.

7.1.4 Cuckoo

During the early testing of the platform, the group quickly encountered an error with python's package manager called **pip**. After upgrading **pip** to its latest version at the time of this writing (19.1.1), Cuckoo would stop working. Cuckoo is reliant on pip version 9.0.3 to run and work properly. In addition to pip, Cuckoo does not support python version higher than 2.7.

All of these version checks have been implemented to the Playbook, and in addition to using version checks, the Playbook installs Cuckoo into python's Virtual Environment.

7.2 Alternative approaches and future work

The group will discuss alternative approaches to the project and the platform, which could be implemented in future work.

7.2.1 Docker

A big decision was to exclude Docker Containers, which is mentioned in appendix D. The decision to exclude Docker was to be certain that the group would deliver a working product. The employer was like-minded on the issue and wanted the group to focus on deploying a functional product. Additionally, the platform would not have much performance difference compared to using containers, and may even be easier to troubleshoot with its current implementation.

7.2.2 Development methodology

When it comes to the development methodology, there were plenty of possibilities. Scrum could have worked better when considering the sprints and have a more detailed visual view of what needs to get done during a sprint. The choice of development method does not mean the students choose the wrong development methodology, because the waterfall methodology gave the group a good overview, they knew what needed to be done during a phase and executed that well.

7.2.3 Cuckoo Sandbox

Like mentioned earlier in the thesis, Cuckoo is extremely modular. Additionally, the tuning of Cuckoo is something to take into consideration. By tuning Cuckoo, the group means that the results of the analysis can vary, based on the options enabled/disabled by changing the configuration files, changing memory profiles used by Volatility or using custom made Yara rules. Tuning Cuckoo was mentioned in chapter 5.5, but the tuning of Cuckoo could potentially be different based on which sandboxes are run in the environment. Even though the options the group used provided the best test results for Windows, might not be the case in other environments running other OS's than Windows.

It might prove beneficial to create different Cuckoo servers to test different OS's, this way the Malware researcher/SOC analyst can customize the Cuckoo servers to better analyze the type of Malware made for the given OS.

7.2.4 Automation

Even though the installation of the platform is very automated as is, there are a few more changes that could have been implemented to automate the installation process fully. Making the platform even more automated involves adding a new role which makes use of regular expressions [40] to edit a line directly in a file that has already been copied over to the server by the Playbook. By using regular expressions in the Playbook installation, one could avoid having to follow most of the steps in chapter 5.

```
- network
- ip_addresses # <-- New Role
- start_up_cuckoo
```

Listing 7.1: Role added to site.yml

Additionally, this would allow the new role to use variables in

```
cuckoo-playbook/inventories/(production/staging)/all/
   group_vars
```

```
### Network cards ###
nic: ens160
ananic: ens192

### IP Addresses ###
ipnic: 172.16.1.30      # <-- new variable
ipananic: 192.168.56.1 # <-- new variable
```

Listing 7.2: group_vars file

Listing 7.3 shows an example of how the file could look like:

```
---
- name: Setting correct IP Address to cuckoo.conf [ 1
   / 2 ]
  become: true
  become_method: "{{␣cuckoo_user␣}}"
  lineinfile:
    dest: /home/cuckoo/.cuckoo.conf/cuckoo.conf
    state: present
    regexp: '^ip ='
    line: 'ip  = "{{␣ipananic␣}}"'

- name: Setting correct IP Address to cuckoo.conf [ 2
   / 2 ]
  become: true
  become_method: "{{␣cuckoo_user␣}}"
  lineinfile:
    dest: /home/cuckoo/.cuckoo.conf/cuckoo.conf
    state: present
    regexp: '^guacd_host'
    line: 'guacd_host = "{{␣ipananic␣}}"'
```

Listing 7.3: main.yml for ip_address role

It would be necessary to add more sections to main.yml in the ip_adress role by using regular expressions to edit the IP addresses of the files that need to change. Adding regular expressions would only require some initial editing of files and the Playbook would install perfectly. The group did not implement regular expressions, due to time limitations. Using regular expressions became clear for the students as a solution to automate the environment even further, but was realized too late into the project period.

Note: The listings above are only an example of how the group would implement this additional role, and is only meant reference of how the Playbook could have been written differently.

7.2.5 Sandboxes

Unfortunately, due to time constraints the group was only able to implement Windows based Sandboxes. In future work, the project should be updated with support for sandboxes with MacOS and Android. In Cuckoos documentation, it was mentioned Android analysis may not work as expected due to becoming a Cuckoo Package [41]. Cuckoo states it is possible to use MacOS, but have no documentation on how it is done. A variety of Sandboxes were implemented to run analysis to see how Malware reacts to different operating systems.

Another adjustment that hopefully could be implemented in the future is the installation of more software in the Sandboxes, for example, different versions of Microsoft Office, this is so the platform can provide better test results for a given Malware. In addition to installing more software to the Sandboxes, kernel debugging is an option that has to be done manually. Still, it is reasonably easy to implement, by adding a COM-port to the virtual machine, and then connecting to this COM-port with VMWare Workstation.

Kernel debugging

Something the group would like to see be implemented in the future is Kernel Debugging for the Sandboxes. Kernel debugging can be done to further analyze a malware, by directly connecting to the Windows Kernel, and by attaching the debugger to a given process, an analyst can see what code the malware executes relative to the Kernel [42].

7.2.6 Software

In the future, the group would also like for VMCloak to be implemented into the platform [43]. VMCloak is a program that hides/cloaks virtual machines by making it harder Malware to detect it is being run in a virtual environment. The group did not implement VMCloak due to the lack of documentation when it comes to setting it up with VMWare infrastructure. Before successfully implementing VMCloak, it would mean the group would have had to go through a lot more trial and error to get an effective integration.

VMCloak's documentation is based on Virtualbox. A possible solution to installing VMCloak on VMWare, would be to create the virtual machine in Virtualbox first, and export/import it to VMWare. The group did not have the time to test the implementation of VMCloak but should work in theory.

7.2.7 Environment

There are also a few infrastructure changes that could be made to the platform in the future. One of them could be to adjust the projects Playbook by moving different parts of the platform into different dedicated servers, thereby allowing the infrastructure of the platform to not depend on one server, and be more redundant to failure. Distributing the infrastructure should be done to avoid losing data when upgrading the server, potentially installing a package that would otherwise harm or break the platform.

In the group's experience, Cuckoo can sometimes stop routing correctly, which causes Internet and VPN routing to stop working. Without Internet routing, the malware analysis results may be affected negatively. Due to time limitations, the group found that rolling the Bionic Beaver server back to its fresh state, install updates and then run the Playbook again would solve these issues. To avoid spending too much time, in the end, troubleshooting the routing issues. Having dedicated servers for PostgreSQL, Elasticsearch and Moloch, along with Suricata, the installation of the platform could be reduced to less than three minutes making it a better choice rather than having to install it all again.

As stated previously in section 7.2.1 the group chose to abstain from implementing Docker Containers in the projects infrastructure, nevertheless it would be beneficial to implement it in the future because it allows the infrastructure to be used in any Environment and it also helps with redundancy by separating the platform's tasks. Implementing containers will not entirely remove the need for virtual machines, because a windows container can not be run under Linux and vice versa. Since this platform is almost entirely built on linux, the need for windows virtual machines would still be needed even if implementing docker Containers.

7.3 Criticism

One of the main lessons the group learned during the assignment is that writing from the beginning is very important. Since the group did not document too much in the beginning, the report writing took longer than anticipated.

The group could have documented more in the early stages. Due to the lack of documentation, the group had to go back to reproduce some of the errors and setups during the implementation and install Cuckoo through Ansible. Due to lack of documentation, this was done to get the necessary and relevant screenshots of the installation process.

The group also discovered that meeting and working together at the university proved to be more productive than working alone from home. As a result, the group held most meetings at the university.

Having read Cuckoo's documentation, the group should have dropped implementing Windows 8.1 and 10, because these OS's are poorly documented and badly supported by Cuckoo. The group should have used their time implementing Android and MacOS instead, which is something the group regrets not having implemented since Android and MacOS was a wanted functionality by the employer.

7.4 Evaluation of the groups work

The group worked together very well, the group held meetings constantly and worked hard in each of them. The students helped each other whenever they were stuck with a given task, which helped keep the standards for the project high by reviewing each others work and motivated each other throughout the project.

By using Trello, the group could see what the other members of the group were working on, and also assist them if they were stuck with the task they were given. Trello helped keep the standards for the project high, by reviewing each other's work and motivating each other throughout the project. The group has worked a lot on the report, and it has been re-manufactured many times to find the correct layout, and so the chapters and its contents made sense to one another. The group feels like they have delivered a good report that reflects the subject and the task they have been given very well.

8 Conclusion

This chapter concludes this thesis; it presents the results achieved in this project, followed by the students' opinions. Lastly, the students give their closing statement.

8.1 Results

NTNU SOC wanted a platform for malware analysis, which was configurable and installed through the use of Ansible. Implementation of the platform was accomplished, although not every functionality described in the task description was fulfilled. The students and the employer had regular contact throughout the thesis. Regular communication with the employer was massively appreciated and used when issues or uncertainties occurred.

The employer was satisfied with the finished product. Additionally, the students felt their job was well executed; this is why the students think the task was accomplished.

An Ansible Playbook was created to implement a fully working Cuckoo analysis platform, integrated with additional tools for thorough malware analysis. Cuckoo is optimized to work together with Windows OS, but works best on Windows 7 as described in chapter testing and verification under the sandbox section 6.5.

By following chapter 5, it is possible to get a complete walk-through of the setup used in this thesis.

8.1.1 Group achievements

This project was a tremendous learning experience; the group learned a lot about Malware analysis, the different tools, and technologies used in this project, how those technologies interact with each other and most importantly how to work as a team.

At the end of the project, the group delivered a functional platform for Malware analysis and managed to implement most of what the employer required in the task description. The group is pleased with everything achieved and learned during this project. That being said if the group had more time, the group would have loved to implement the technologies named in future work.

8.2 Closing statement

The students learned a lot of valuable lessons throughout this project that we will carry with us for the rest of our lives. We hope NTNU SOC will find our platform useful and we would like to thank Christoffer Vargtass and NTNU SOC for trusting us with such an interesting and relevant project to work with and learn from. Furthermore, NTNU SOC has many possibilities for how they want to use this project for both production and development environments. The code used in the making of the automation of Cuckoo is on an open repository on GitHub and is therefore free for anyone to use. Finally, we would like to thank our supervisor, Eigil Obrestad, for giving us critical feedback throughout the project, pushing us to write a better report and developing a better project.

Bibliography

[1] VMWare. 2019. What is a hypervisor? URL: https://www.vmware.com/topics/glossary/content/hypervisor.

[2] Debain. March 2019. systemd - system and service manager. URL: https://wiki.debian.org/systemd.

[3] Oktavianto, D. & Muhardianto, I. *Cuckoo Malware Analysis*, chapter 02. Packt Publishing, 2013.

[4] Ubuntu. April 2019. Ubuntu lifecycle (eol). URL: https://endoflife.software/operating-systems/linux/ubuntu.

[5] JabRef. Bibliography reference manager. URL: https://www.jabref.org/#jabref.

[6] Sikorski, M. & Honig, A. February 2012. Practical malware analysis - the goals of malware analysis. *O'REILLY*. URL: https://learning.oreilly.com/library/view/practical-malware-analysis/9781593272906/ch01s01.html.

[7] Av-test. April 2019. Malware. URL: https://www.av-test.org/en/statistics/malware/.

[8] McDowell, R. What are the disadvantages of antiviruses? URL: https://www.techwalla.com/articles/what-are-the-disadvantages-of-antiviruses.

[9] Sikorski, M. & Honig, A. February 2012. Practical malware analysis - malware analysis in virtual machines. *O'REILLY*. URL: https://learning.oreilly.com/library/view/practical-malware-analysis/9781593272906/ch03.html.

[10] Sikorski, M. & Honig, A. February 2012. Practical malware analysis - what is malware analysis? *O'REILLY*. URL: https://learning.oreilly.com/library/view/practical-malware-analysis/9781593272906/pr06s01.html.

[11] Sikorski, M. & Honig, A. February 2012. Practical malware analysis - malware analysis techniques. *O'REILLY*. URL: https://learning.oreilly.com/library/view/practical-malware-analysis/9781593272906/ch01s02.html.

[12] Krister, K. M. Automated analyses of malicious code. Master's thesis, Norwegian University of Science and Technology, 2009. URL: http://hdl.handle.net/11250/251364.

[13] Oktavianto, D. & Muhardianto, I. *Cuckoo Malware Analysis*, chapter 01. Packt Publishing, 2013. URL: https://learning.oreilly.com/library/view/cuckoo-malware-analysis/9781782169239/ch01s04.html.

[14] Impe, K. V. June 2015. Signature-based detection with yara. URL: https://securityintelligence.com/signature-based-detection-with-yara/.

[15] Porup, J. July 2018. What is the tor browser? how it works and how it can help you protect your identity online. URL: https://www.csoonline.com/article/3287653/what-is-the-tor-browser-how-it-works-and-how-it-can-help-you-protect-your-identity-online.html.

[16] O'DRISCOLL, A. April 2019. What is a vpn connection, what does it do, and how do you set one up? URL: https://www.comparitech.com/blog/vpn-privacy/what-is-a-vpn-connection/.

[17] Honig, A. & Sikorski, M. *Practical Malware Analysis*, chapter 03. No Starch Press, 2012. URL: https://learning.oreilly.com/library/view/practical-malware-analysis/9781593272906/ch04s08.html.

[18] Foundation, C. April 2019. Best practices. URL: https://docs.ansible.com/ansible/latest/user_guide/playbooks_best_practices.html.

[19] Synopsys-Editorial-Team. March 2017. Top 4 software development methodologies. URL: https://www.synopsys.com/blogs/software-security/top-4-software-development-methodologies/.

[20] Atlassian. URL: https://trello.com/about.

[21] Atlassian, D. R. What is kanban? URL: https://www.atlassian.com/agile/kanban.

[22] Foundation, C. October 2018. Cuckoo sandbox documentation. URL: https://buildmedia.readthedocs.org/media/pdf/cuckoo/latest/cuckoo.pdf.

[23] Vanderzyden, J. September 2015. What is elasticsearch, and how can i use it? URL: https://qbox.io/blog/what-is-elasticsearch.

[24] Suricata. Ubuntu installation - personal package archives (ppa). URL: https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Ubuntu_Installation_-_Personal_Package_Archives_(PPA).

[25] Moloch. 2019. Download moloch. URL: https://molo.ch/downloads.

[26] January 2018. Elasticsearch throws an unhandled exception! Github Issues. URL: https://github.com/cuckoosandbox/cuckoo/issues/2085.

[27] Elasticsearch 5.6.16. URL: https://www.elastic.co/downloads/past-releases/elasticsearch-5-6-16.

[28] Digitalocean. URL: https://www.digitalocean.com/.

[29] Dhatheway. February 2018. Rooter not starting tun interface or able to reach the internet 2118. URL: https://github.com/cuckoosandbox/cuckoo/issues/2118.

[30] Foundation, C. 2018. Network configuration. URL: https://cuckoo.readthedocs.io/en/latest/installation/guest/network/.

[31] Foundation, C. April 2019. Ansible documentation. URL: https://docs.ansible.com/ansible/latest/index.html.

[32] Forked from julianoborbara, edited by Stian, N. & Daniel. May 2019. Ansible-cuckoo. URL: https://github.com/Desiqvel/Ansible-Cuckoo.

[33] Drake, M. May 2018. How to set up an openvpn server on ubuntu 18.04. URL: https://www.digitalocean.com/community/tutorials/how-to-set-up-an-openvpn-server-on-ubuntu-18-04.

[34] Gaynor, A. August 2014. Enabling certificate verification by default for stdlib http clients. URL: https://www.python.org/dev/peps/pep-0476/.

[35] Foundation, C. 2018. Ioerror:[errno 24] too many open files. URL: https://cuckoo.sh/docs/faq/#ioerror-errno-24-too-many-open-files.

[36] EasyEngine. 2019. Increase "open files limit". URL: https://easyengine.io/tutorials/linux/increase-open-files-limit.

[37] Foundation, C. 2018. Creation of the virtual machine. URL: https://cuckoo.readthedocs.io/en/latest/installation/guest/creation/.

[38] Ganacharya, T. January 2018. A worthy upgrade: Next-gen security on windows 10 proves resilient against ransomware outbreaks in 2017. URL: https://www.microsoft.com/security/blog/2018/01/10/a-worthy-upgrade-next-gen-security-on-windows-10-proves-resilient-against-ransomware-outbreaks-in-2017/.

[39] julianoborba. Mars 2018. Ansible-cuckoo. URL: https://github.com/julianoborba/Ansible-Cuckoo.

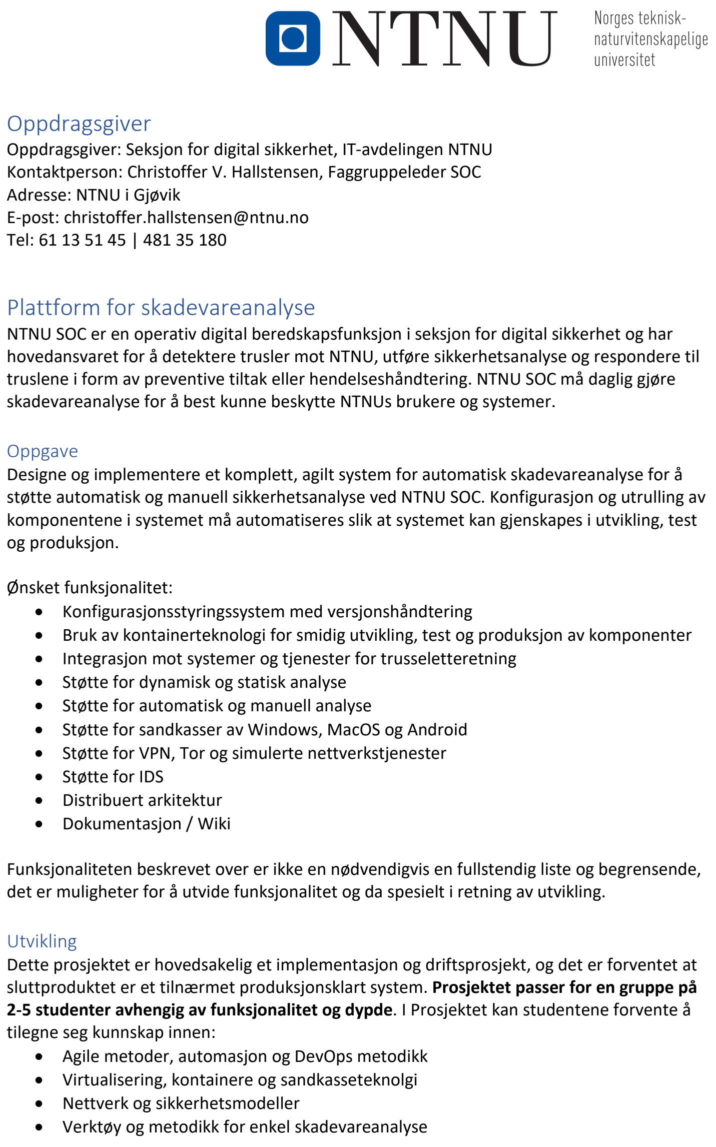
[40] Inc., R. H. March 2019. lineinfile - ensure a particular line is in a file, or replace an existing line using a back-referenced regular expression. URL: https://docs.ansible.com/ansible/2.4/lineinfile_module.html.

[41] Foundation, C. Configuration (android analysis). URL: http://docs.cuckoosandbox.org/en/latest/installation/host/configuration_android/.

[42] everdox. July 2017. Setting up kernel debugging using windbg and vmware. URL: https://www.triplefault.io/2017/07/setting-up-kernel-debugging-using.html.

[43] Bremer, J. 2015. Vmcloak documentation. URL: https://vmcloak.readthedocs.io/en/latest/.

A Task Description



NTNU Norges teknisk-naturvitenskapelige universitet

Oppdragsgiver

Oppdragsgiver: Seksjon for digital sikkerhet, IT-avdelingen NTNU
Kontaktperson: Christoffer V. Hallstensen, Faggruppeleder SOC
Adresse: NTNU i Gjøvik
E-post: christoffer.hallstensen@ntnu.no
Tel: 61 13 51 45 | 481 35 180

Plattform for skadevareanalyse

NTNU SOC er en operativ digital beredskapsfunksjon i seksjon for digital sikkerhet og har hovedansvaret for å detektere trusler mot NTNU, utføre sikkerhetsanalyse og respondere til truslene i form av preventive tiltak eller hendelseshåndtering. NTNU SOC må daglig gjøre skadevareanalyse for å best kunne beskytte NTNUs brukere og systemer.

Oppgave

Designe og implementere et komplett, agilt system for automatisk skadevareanalyse for å støtte automatisk og manuell sikkerhetsanalyse ved NTNU SOC. Konfigurasjon og utrulling av komponentene i systemet må automatiseres slik at systemet kan gjenskapes i utvikling, test og produksjon.

Ønsket funksjonalitet:

- Konfigurasjonsstyringssystem med versjonshåndtering
- Bruk av kontainerteknologi for smidig utvikling, test og produksjon av komponenter
- Integrasjon mot systemer og tjenester for trusseletteretning
- Støtte for dynamisk og statisk analyse
- Støtte for automatisk og manuell analyse
- Støtte for sandkasser av Windows, MacOS og Android
- Støtte for VPN, Tor og simulerte nettverkstjenester
- Støtte for IDS
- Distribuert arkitektur
- Dokumentasjon / Wiki

Funksjonaliteten beskrevet over er ikke en nødvendigvis en fullstendig liste og begrensende, det er muligheter for å utvide funksjonalitet og da spesielt i retning av utvikling.

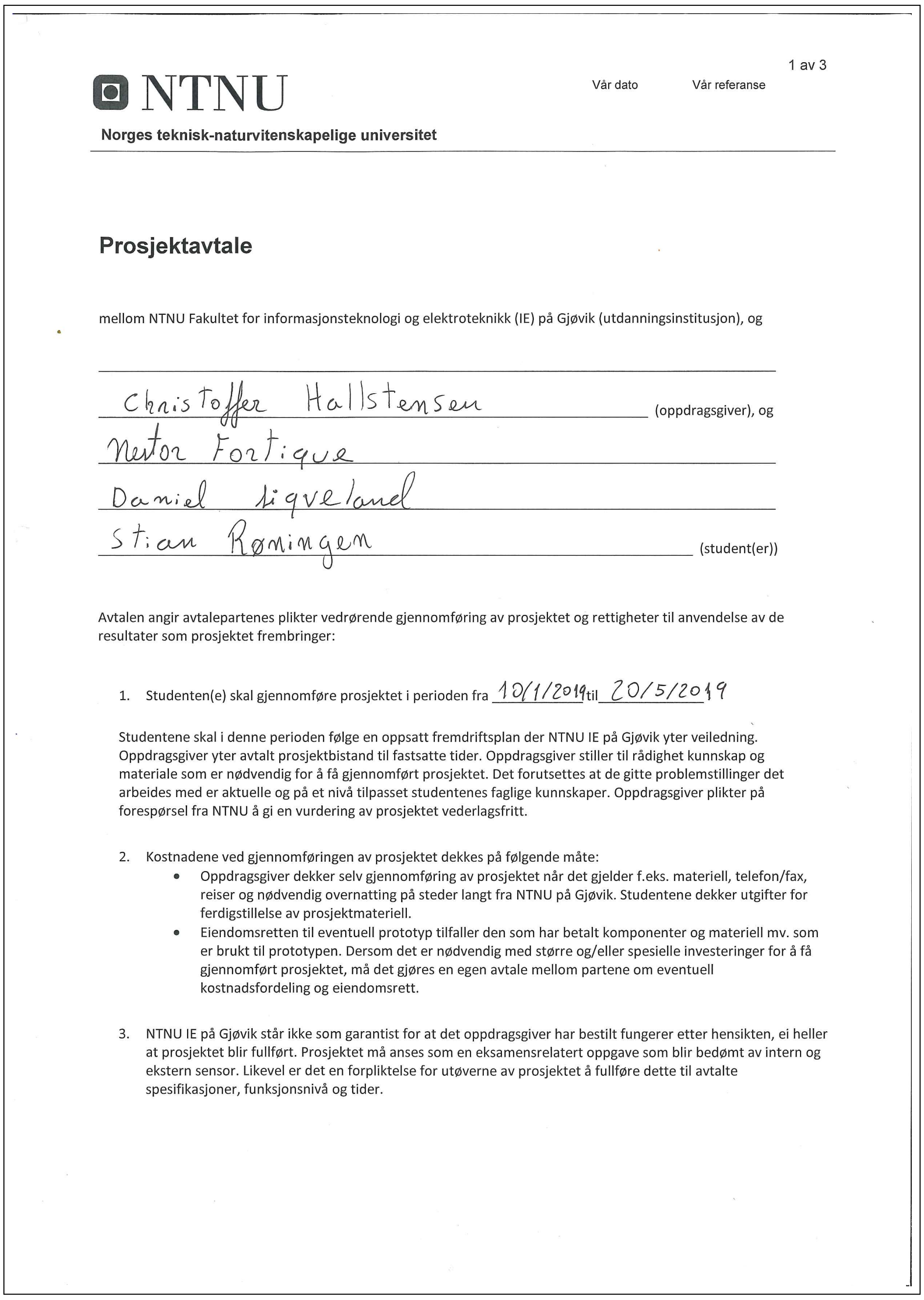
Utvikling

Dette prosjektet er hovedsakelig et implementasjon og driftsprosjekt, og det er forventet at sluttproduktet er et tilnærmet produksjonsklart system. **Prosjektet passer for en gruppe på 2-5 studenter avhengig av funksjonalitet og dypde**. I Prosjektet kan studentene forvente å tilegne seg kunnskap innen:

- Agile metoder, automasjon og DevOps metodikk
- Virtualisering, kontainere og sandkasseteknolgi
- Nettverk og sikkerhetsmodeller
- Verktøy og metodikk for enkel skadevareanalyse

Figure 18: Task Description

B Project Agreement



1 av 3

NTNU

Vår dato Vår referanse

Norges teknisk-naturvitenskapelige universitet

Prosjektavtale

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

Christoffer Hallstensen (oppdragsgiver), og

Nestor Fortique

Daniel Tiqveland

Stian Røningen (student(er))

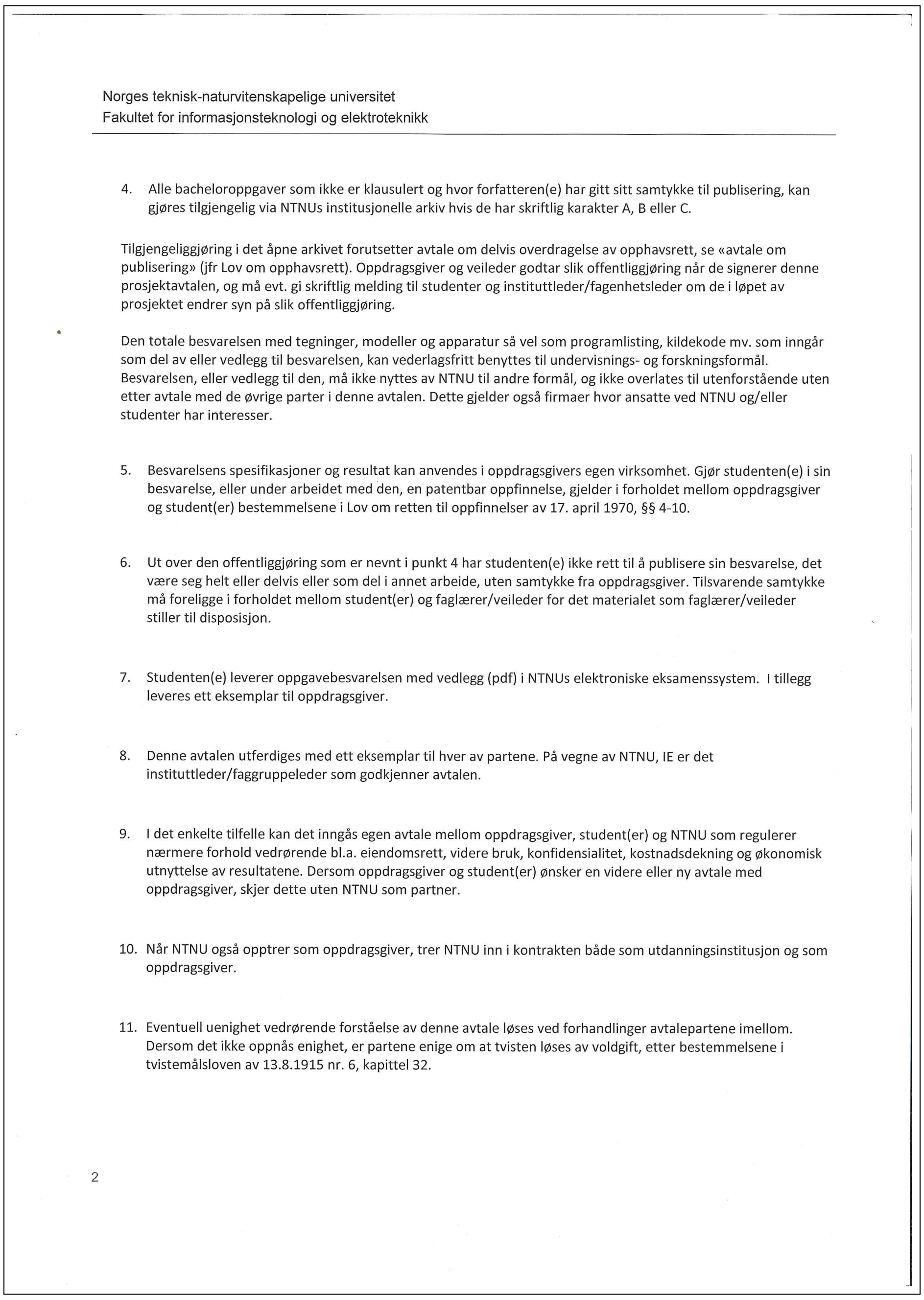
Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra 10/1/2019 til 20/5/2019

   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.



Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk

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.

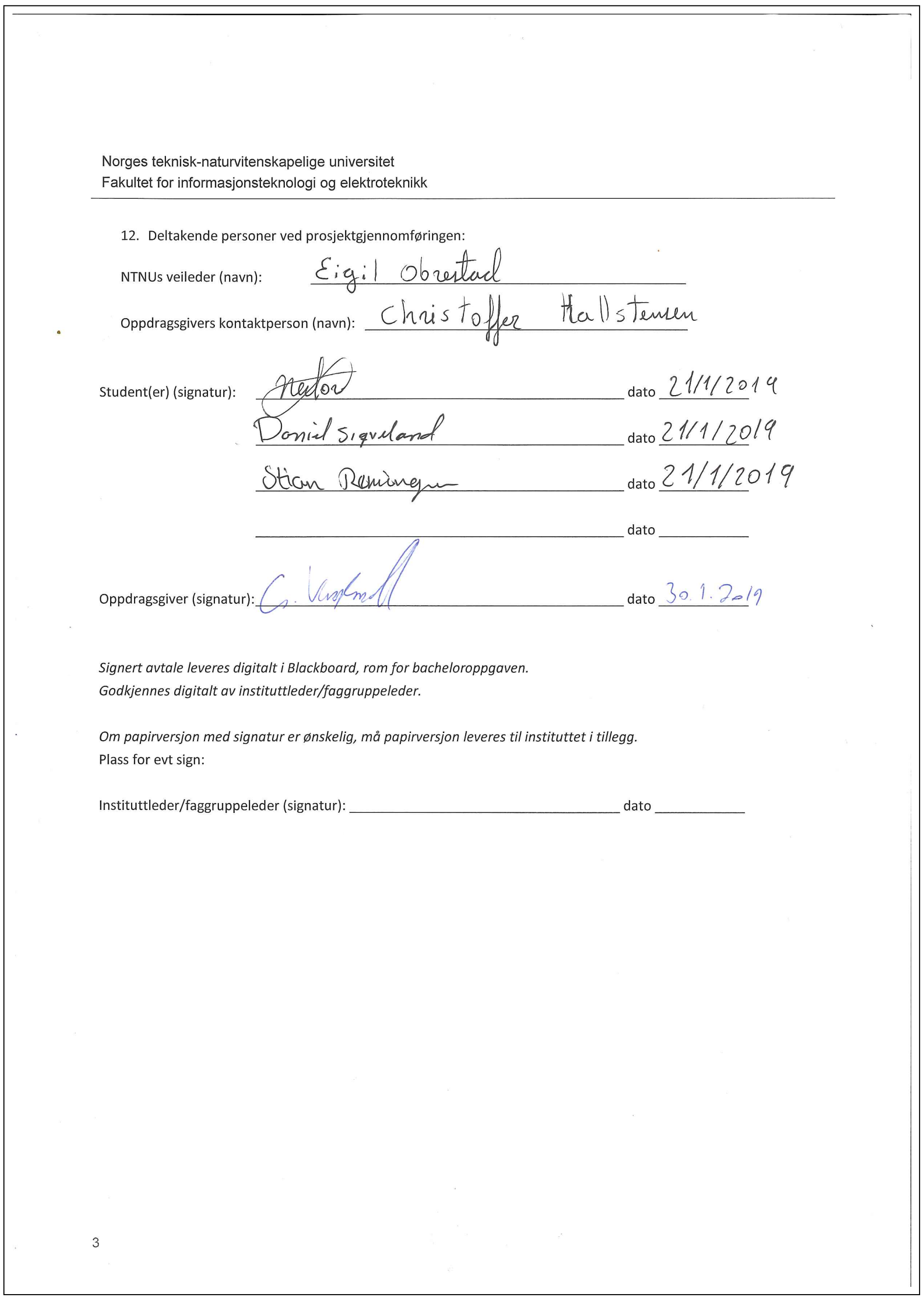
8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, IE er det instituttleder/faggruppeleder som godkjenner avtalen.

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.

2



Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk

12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): Eigil Obrestad

Oppdragsgivers kontaktperson (navn): Christoffer Hallstensen

Student(er) (signatur): [illegible] dato 21/1/2019

Daniel Sigveland dato 21/1/2019

[illegible] dato 21/1/2019

dato

Oppdragsgiver (signatur): [illegible] dato 30.1.2019

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 ________

3

Figure 19: Task Description

C Project Plan

C.1 Introduction

In these evolving times, detecting and removing Malware artifacts is not enough: it is vitally important to understand how they operate in order to understand the context, the motivations, and the goals of a breach.

C.2 Goals and Scope

C.2.1 Project's goals

Design and implement a complete and agile platform automatic Malware analysis. The project aims to assist NTNU SOC in its daily work by detecting threats against NTNU and analyzing these threats.

Result goals

- Automated configuration and deployment of system components
- Immutable platform for use in development, test and production
- Platform for simple malware analysis

Effect goals

- Automatic and manual security analysis of malware
- Analyze new types of malware, as well as existing ones
- Used in context of lectures in the future

C.2.2 Project's scope

Brief description of the project

This project aims to create a platform for Malware analysis by using a handful of tools to automate and administrate the installation process along with operating the system when it is finished. The main focus will be Windows(100% implementation and malware analysis) and if time allows it, the project will include Android and MacOS.

Timeframe

The project period will be from 10.01.2019-20.05.2019. After the final date, the work on the project will cease regardless of the project's state unless arrangements are made.

Employer's scope

Requirements / limitations for project and final report:

The employer wants a platform that is mostly built on Containers. The platform must have a configuration component that allows for version management and has integration towards systems for threat intelligence. It is required that there is an implementation of different network services such as TOR and/or VPN to Spoof the analyzers location and lure the attacker into unleashing the full potential of the Malware for better analysis. The platform will be built upon VMWare and containers will be built using Docker. Sandboxes will be run in VMWare, while Cuckoo will be used for malware analysis. The entire platform will be administrated through the use of Ansible. Network Intrusion Detection System (IDS) systems will be used to ensure that malware is detected by the use of Signatures and to help analysts identify the signature set a specific malware sample triggers.

Student's scope

The following requirements/limitations the students has to deliver/do for this project:

Write a project plan containing scheduling of the different phases during the project (Delivered to the supervisor). Design and implement a complete agile system for automatic and manual analysis of Malware. The configuration and deployment of components in the system with DevOps in mind. (A final report to be graduated in Bachelor in IT-Operations and Information Security at NTNU Gjøvik (Delivered to NTNU Gjøvik). Finally have a presentation of the thesis.

C.3 Organization

C.3.1 Organizational structure

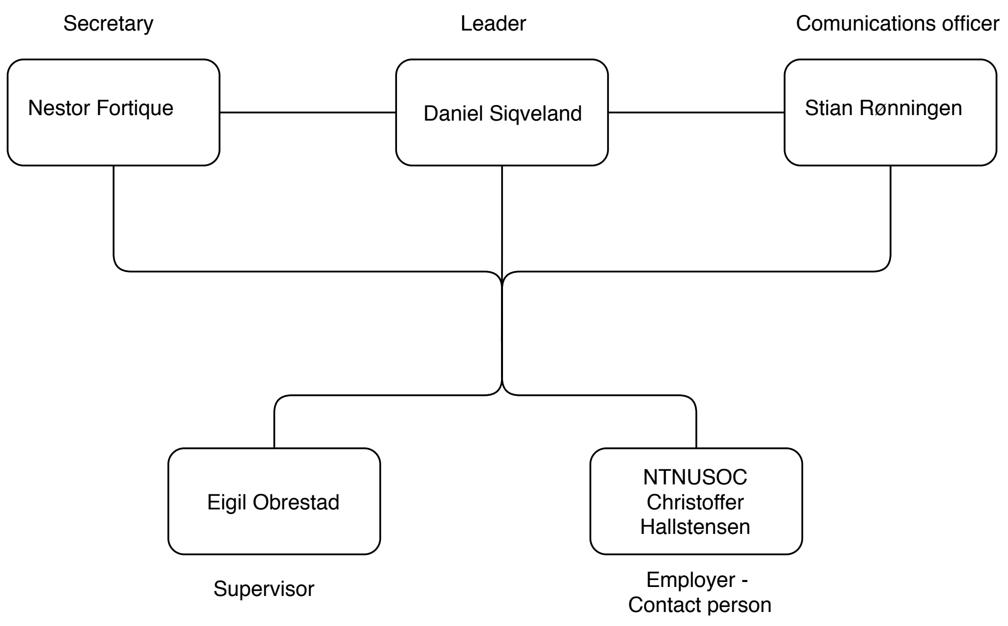


Figure 20: The organizational structure of this project

C.3.2 Roles

The following roles have been specified for this project:

- **Employer & Product Owner** - Christoffer Vargtass Hallstensen representing NTNU SOC
  - Ensure the quality of the student's works and proof that it matches the wished requirements.
  - Provide the needed information about systems for the students during the defined period.
- **Supervisor** - Eigil Obrestad
  - Follow the student's working process.
  - Provide feedback, and ensure that the students is *on track*.
- **Students & task performers** - Daniel E. Krohn Siqveland, Nestor Gerrardo Fortique & Stian Rønningen
  - Daniel, as the chosen leader of the group, will set time and dates for when the group meets.
  - Nestor will be the secretary, responsible of writing summaries of meetings.
  - Stian will be the one in charge for sending mails and arranging meetings with the supervisor and the employer.
  - Responsible for project's time management and delivering the requirements of the project within the deadlines.

C.3.3 Time management and meetings

During this project, the students work Monday-Friday and sometimes during weekends. The students are also responsible of maintaining the communication with their employer. A meeting between the employer and the students is defined by dropping by the office or arranging meetings by email if needed, and could be changed later according to the process and time management. Additionally, the students are responsible for the meetings between the supervisor and the students. It is already defined to be once each week on Mondays.

C.4 Risk Management

This section will contain the Risks that could appear during this project. Each risk will be evaluated relative to it's likelihood to happen and consequence if a risk occurs. In the risk analysis subsection under, it's defined the likelihood and consequence on a scale from one (1) to five (5). Where one is defined as *None Critical* and five is considered *Critical*.

C.4.1 Risk analysis

Nr#	Risk	Likelihood	Consequence	Countermeasure
1	The project is not completed before the deadline or delivery delay. Reasons could be Backup lost, poor estimate of points in activities.	2	5	Yes
2	Major changes in requirements specifications from employer.	1	4	No
3	Loss of resources (data or report).	2	5	Yes
4	Project shutdown	1	5	No
5	The final product does not match the requirements of the employer.	1	4	Yes

Table 5: Project's risks

C.4.2 Countermeasures

To ensure the quality of the student's work in the project, a list of countermeasures for the mentioned Risks is listed in the table below.

Nr#	Risk Countermeasures
1	The students will be using a more agile version of the waterfall development model. So, both the quality of the work and the delivery should be matching the requirements.
3	The data and the report of the project is vulnerable to be lost. Therefore, it should be stored and protected. Also, a back-up of data will be often during the project period. The report is hosted in Overleaf cloud storage and a back-up will happen periodically.
5	The students should organize the work and share the process with the supervisor and employer to ensure that they are following the right track.

Table 6: Project's risk countermeasures

C.5 Planning & Report

C.5.1 Working process

Due to the task requirements from the employer, the project is divided into four phases. The project will go through comprehensive testing, setup of tools and Scripts to make the end product as simple as possible along with having included the requirements from the employer.

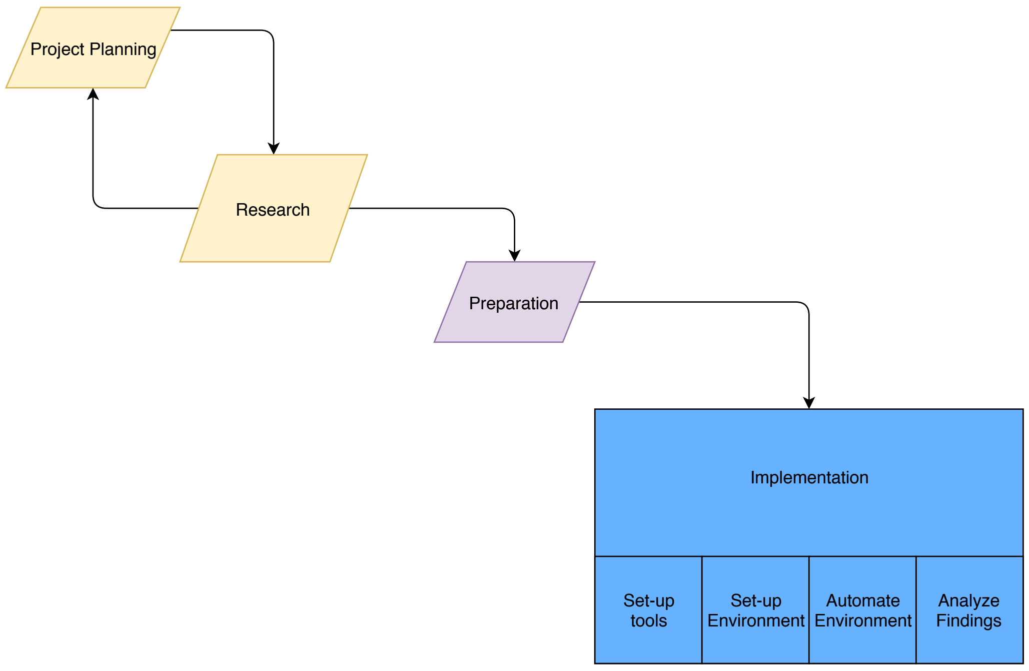


Figure 21: Working process during the project

- **Phase 1 - Project planning**
  - Define the goals of the project.
  - Set up a plan of the working process.
  - Establish communications with the supervisor and the employer.
- **Phase 2 - Research**
  - The students use the period to obtain the relevant information for the project, by learning and reading about the different tools based on documentations.
  - It is possible that during this phase the predefined project plan will change if needed.
- **Phase 3 - Preparation**
  - The students use this time to prepare for the experiment by preparing and learning the tools for the project. This includes preparing to deploy the Environment based on *best practices*.
- **Phase 4 - Implementation**
  - The implementation phase contains 4 sub-phases that should run in parallel:

- **Phase 4.1 - Set up the required tools**
- **Phase 4.2 - Set up environment**
- **Phase 4.3 - Automate environment**
- **Phase 4.4 - Analyze findings**

The report (thesis) will be written in parallel with all phases as the students are starting to see a working product.

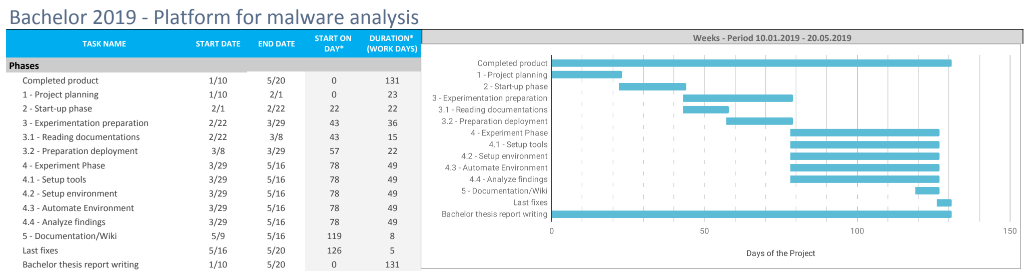
C.5.2 Experiment's Notes

The reason for running all the sub-phases at the same time under the experiment is to ensure the quality of the environment. What is likely to happen the first time is that the students set up the environment manually and then automate with Ansible and Docker from there to create an agile environment that can be used in later situations. NTNU's IT-department has been very kind to lend the students a few servers to run the environment on, because the students laptops were not up to the task of running the environment.

C.6 Schedule

C.6.1 Gantt diagram

The figure below is a Gantt Diagram that shows how the mentioned phases in C.5.1 are divided over 18 weeks and four days as mentioned in C.2.2.


Bachelor 2019 - Platform for malware analysis

TASK NAME	START DATE	END DATE	START ON DAY*	DURATION* (WORK DAYS)
Phases				
Completed product	1/10	5/20	0	131
1 - Project planning	1/10	2/1	0	23
2 - Start-up phase	2/1	2/22	22	22
3 - Experimentation preparation	2/22	3/29	43	36
3.1 - Reading documentations	2/22	3/8	43	15
3.2 - Preparation deployment	3/8	3/29	57	22
4 - Experiment Phase	3/29	5/16	78	49
4.1 - Setup tools	3/29	5/16	78	49
4.2 - Setup environment	3/29	5/16	78	49
4.3 - Automate Environment	3/29	5/16	78	49
4.4 - Analyze findings	3/29	5/16	78	49
5 - Documentation/Wiki	5/9	5/16	119	8
Last fixes	5/16	5/20	126	5
Bachelor thesis report writing	1/10	5/20	0	131

Figure 22: Gantt chart of the project flow and phases timing

Planning is taking a part of this thesis from the start, and it is estimated to take three weeks of work. During the planning, the students will use the time to obtain knowledge about the assignment.

The next phase in line is the start-up phase. The students will start learning and reading about the tools - what the best practices are, how to implement them and why they should be implemented. Estimated time for this phase will be roughly three weeks.

After getting the basics, the project will enter the preparation-for-experimentation phase. This is the phase where tools are tested for the first time, getting the hands-on experience. Since some of the tools can be very overwhelming, the time frame for this phase is set to three weeks.

The students will use five weeks of the project to implement the environment by automating installation of components. This is also the phase where everything will be tested, including testing Malware in the environment and analyzing the system thoroughly.

Even though the thesis will be written in parallel with all the phases, the project will have four weeks at the end where all of the time will be dedicated to writing and polishing the thesis.

D Midterm review

This section presents the the midterm review, where the students met with their employer. The meeting took place 8th of October and lasted one hour.

D.1 Status of the project before the meeting

The students had a working step by step guide to install, configure and run Cuckoo, but not a working Ansible Playbook that downloads and installs everything necessary to run Cuckoo. The Gantt-diagram in C.6.1 was followed and the students where on schedule, with focus on the technical solution. This resulted in less writing the thesis than originally planned. In preparation to the meeting the students had prepared a demo run with an Malware that would show how Cuckoo worked. Also prepared a few questions that would help with clarifying the process ahead.

D.2 Summary of the meeting

The employer answered questions regarding how much of the setup was to be automated with Ansible. The students got an impression that the employer was more than happy with the environment so Cuckoo could run smoothly. When it was time to present the demo to the employer there was an issue with vCenter Server Appliance resulting in the demo not working, but luckily the students where able to present the results of an earlier analysis performed prior to the meeting.

The following two main factors were discussed during the meeting:

1. **Clarification**
   During the meeting the students got clarification on what to automate. The wanted functionality from the employer was that the installation process of Cuckoo would be automated. Support for more Operating System than Windows was also discussed.
2. **Docker**
   During the Planning Phase (see Appendix C) the students mentioned Docker to be used as a way to run Cuckoo. This proved to be more challenging than helpful. The students then chose to abandon the docker implementation of Cuckoo and rather focus on getting a manual installation of it working. This resulted in a much cleaner approach to the installation process itself by allowing easier automation with Ansible. Following this approach will also leave the installation with a larger backlog of earlier analyzes, restricted only by harddrive space.

D.3 The student's decision

Due to time constraints, the students chose to only implement and test the platform on a Windows environment. If time allows it, implementation of MacOS, iOS and Android will be included in the platform. All of the Operating System's mentioned in this section are supported by Cuckoo, but not yet tested in this project.

D.4 Modifying the topic

In order to modify the original topic to a new one, several elements has been taken into consideration, such as:

- **The predefined project's motivation, goals and research question.**
  The employer wish was to have as many Operating System as possible, but due to time constraints, the students and the employer agreed on fully implementing Windows as the primary goal.
- **Time constraints**
  The manual installation and troubleshooting took way longer than the students anticipated. Additionally Cuckoo's configuration opportunities and documentation were hard to familiarize and learn. As a result there wasn't much time to write on the thesis, the students focused more on providing a working prototype to show the employer, also gathering enough knowledge about the environment to get more specific questions.

E Route.py

```python
#!/usr/bin/python
import os
import subprocess

IP_ROUTE_TABLES='/etc/iproute2/rt_tables'

if __name__ == '__main__':
    config_name = os.environ.get('config')
    local_ip = os.environ.get('ifconfig_local')
    vpn_gateway = os.environ.get('route_vpn_gateway')
    common_name = os.environ.get('common_name')
    dev = os.environ.get('dev')
    ip_table = common_name
    print 'ip rule add from {} table {}'.format(local_ip, ip_table)
    print 'ip route add default via {} dev {} table {}'.format(vpn_gateway, dev, ip_table)
    subprocess.call(['ip', 'rule', 'add', 'from', local_ip, 'table', ip_table])
    subprocess.call(['ip', 'route', 'add', 'default','via', vpn_gateway, 'dev', dev, 'table', ip_table])
```

F OpenVPN Systemd

```
[Unit]
Description=Openvpn Client Service
After=network-online.target
Wants=network-online.target

[Service]
ExecStart=/usr/sbin/openvpn --config openvpn.conf --
   script-security 2 --route-noexec --route-up route.
   py
Restart=on-failure
User=root
Group=root
WorkingDirectory=/etc/openvpn/client

[Install]
WantedBy=multi-user.target
```

G Cuckoo Systemd

G.1 Cuckoo Rooter

```
[Unit]
Description=Cuckoo Rooter Service
After=network.target openvpnclient.service

[Service]
ExecStart=/opt/cuckoo/bin/cuckoo rooter
Restart=on-failure
User=root
Group=root
WorkingDirectory=/home/cuckoo/.cuckoo

[Install]
WantedBy=multi-user.target
```

G.2 Cuckoo

```
[Unit]
Description=Cuckoo Sandbox Service
After=network.target cuckoo_rooter.service

[Service]
ExecStart=/opt/cuckoo/bin/cuckoo
Restart=on-failure
User=cuckoo
Group=cuckoo
WorkingDirectory=/home/cuckoo/.cuckoo

[Install]
WantedBy=multi-user.target
```

G.3 Cuckoo Web

```
[Unit]
Description=Cuckoo Web Service
After=network.target cuckoo.service

[Service]
ExecStart=/opt/cuckoo/bin/cuckoo web runserver
    172.16.1.30:8880
Restart=on-failure
User=cuckoo
Group=cuckoo
WorkingDirectory=/home/cuckoo/.cuckoo

[Install]
WantedBy=multi-user.target
```

G.4 Cuckoo API

```
[Unit]
Description=Cuckoo API Service
After=network.target cuckoo_web.service

[Service]
ExecStart=/opt/cuckoo/bin/cuckoo api --host
    172.16.1.30 --port 8881
Restart=on-failure
User=cuckoo
Group=cuckoo
WorkingDirectory=/home/cuckoo/.cuckoo

[Install]
WantedBy=multi-user.target
```

H vsphere.py

This is the version that Cuckoo installs by default

```
97          # Workaround for PEP-0476 issues in recent Python versions
98          if self.options.vsphere.unverified_ssl:
99              sslContext = ssl.SSLContext(ssl.PROTOCOL_TLSv1)
100             sslContext.verify_mode = ssl.CERT_NONE
101             self.connect_opts["sslContext"] = sslContext
102             log.warn("Turning␣off␣SSL␣certificate␣verification!")
```

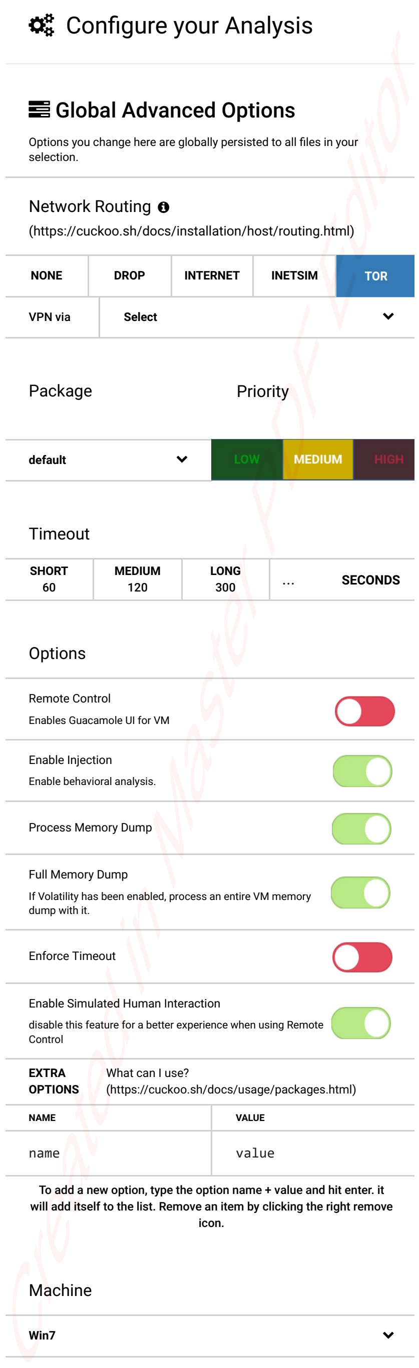
This is the patched version that works when Cuckoo interacts with VCSA. Notice that line 99 has changed from **ssl.PROTOCOL_TLSv1** to **ssl.PROTOCOL_v23**.

```
97          # Workaround for PEP-0476 issues in recent Python versions
98          if self.options.vsphere.unverified_ssl:
99              sslContext = ssl.SSLContext(ssl.PROTOCOL_SSLv23)
100             sslContext.verify_mode=ssl.CERT_NONE
101             self.connect_opts["sslContext"] = sslContext
102             log.warn("Turning␣off␣SSL␣certificate␣verification!")
```

I Sample Virtual Machine config - vsphere.conf

```
[Win7]
# Specify the label name of the current machine as specified on your
# vSphere host.
label = Win7

# Specify the operating system platform used by current machine
# [windows/darwin/linux].
platform = windows

# Please specify the name of the snapshot. This snapshot should be taken
# while the machine is running and the agent started.
snapshot = clean_agent

# Specify the IP address of the current virtual machine. Make sure that the
# IP address is valid and that the host machine is able to reach it. If not,
# the analysis will fail.
ip = 192.168.56.7

# (Optional) Specify the OS profile to be used by volatility for this
# virtual machine. This will override the guest_profile variable in
# memory.conf which solves the problem of having multiple types of VMs
# and properly determining which profile to use.
# Profiles defined by Volatility, see link below for more information
# https://github.com/volatilityfoundation/volatility/wiki/2.6-Win-Profiles
# Windows profiles only. Linux and Mac OS profiles can be found in the link below
# https://github.com/volatilityfoundation/profiles
osprofile = Win7SP1x64_23418
```

J Cuckoo Web GUI options



Configure your Analysis

Global Advanced Options

Options you change here are globally persisted to all files in your selection.

Network Routing (https://cuckoo.sh/docs/installation/host/routing.html)

NONE | DROP | INTERNET | INETSIM | [✓ TOR]

VPN via: Select

Package: default

Priority: LOW | [✓ MEDIUM] | HIGH

Timeout: SHORT 60 | MEDIUM 120 | LONG 300 | … SECONDS

Options

Remote Control Enables Guacamole UI for VM	Off
Enable Injection Enable behavioral analysis.	On
Process Memory Dump	On
Full Memory Dump If Volatility has been enabled, process an entire VM memory dump with it.	On
Enforce Timeout	Off
Enable Simulated Human Interaction disable this feature for a better experience when using Remote Control	On

EXTRA OPTIONS What can I use? (https://cuckoo.sh/docs/usage/packages.html)

NAME	VALUE
name	value

To add a new option, type the option name + value and hit enter. it will add itself to the list. Remove an item by clicking the right remove icon.

Machine: Win7

Figure 23: Cuckoo Web GUI options

K Ansible directory structure

```
Ansible-Cuckoo/
|-- cuckoo-playbook
|   |-- inventories
|   |   |-- production
|   |   |   |-- group_vars
|   |   |   |   --- all
|   |   |   --- hosts
|   |   --- staging
|   |       |-- group_vars
|   |       |   --- all
|   |       --- hosts
|   |-- roles
|   |   |-- build_directory
|   |   |   --- tasks
|   |   |        main.yml
|   |   |-- cuckoo
|   |   |   |-- files
|   |   |   |   |-- auxiliary.conf
|   |   |   |   |-- avd.conf
|   |   |   |   |-- cuckoo.conf
|   |   |   |   |-- esx.conf
|   |   |   |   |-- kvm.conf
|   |   |   |   |-- limits.conf
|   |   |   |   |-- memory.conf
|   |   |   |   |-- physical.conf
|   |   |   |   |-- processing.conf
|   |   |   |   |-- qemu.conf
|   |   |   |   |-- reporting.conf
|   |   |   |   |-- routing.conf
|   |   |   |   |-- sysctl.conf
|   |   |   |   |-- virtualbox.conf
|   |   |   |   |-- vmware.conf
|   |   |   |   |-- vsphere.conf
|   |   |   |   |-- vsphere.py
|   |   |   |   --- xenserver.conf
|   |   |   --- tasks
|   |   |       --- main.yml
|   |   |-- cuckoo_user
|   |   |   --- tasks
|   |   |       --- main.yml
|   |   |-- elasticsearch
|   |   |   |-- files
|   |   |   |   --- elasticsearch.yml
|   |   |   --- tasks
|   |   |       --- main.yml
|   |   |-- inetsim
|   |   |   |-- files
|   |   |   |   |-- inetsim
|   |   |   |   --- inetsim.conf
|   |   |   --- tasks
|   |   |       --- main.yml
|   |   |-- moloch
|   |   |   |-- files
|   |   |   |   |-- Configure
|   |   |   |   |-- molochcapture.service
|   |   |   |   --- molochviewer.service
|   |   |   --- tasks
|   |   |       --- main.yml
|   |   |-- network
|   |   |   --- tasks
|   |   |       --- main.yml
|   |   |-- openvpn
|   |   |   |-- files
|   |   |   |   |-- openvpnclient.service
|   |   |   |   |-- openvpn.conf
|   |   |   |   |-- route.py
|   |   |   |   --- rt_tables
|   |   |   --- tasks
|   |   |       --- main.yml
|   |   |-- postgresql
|   |   |   --- tasks
|   |   |       --- main.yml
|   |   |-- python
|   |   |   --- tasks
|   |   |       --- main.yml
|   |   |-- python_packages
|   |   |   --- tasks
|   |   |       --- main.yml
|   |   |-- start_up_cuckoo
|   |   |   |-- files
|   |   |   |   |-- cuckoo_api.service
|   |   |   |   |-- cuckoo_rooter.service
|   |   |   |   |-- cuckoo.service
|   |   |   |   --- cuckoo_web.service
|   |   |   --- tasks
|   |   |       --- main.yml
|   |   |-- suricata
|   |   |   |-- files
|   |   |   |   |-- oinkmaster.conf
|   |   |   |   |-- suricata
|   |   |   |   |-- suricata.log
|   |   |   |   --- suricata.yaml
|   |   |   --- tasks
|   |   |       --- main.yml
|   |   |-- tor
|   |   |   |-- files
|   |   |   |   --- torrc
|   |   |   --- tasks
|   |   |       --- main.yml
|   |   |-- ubuntu_packages
|   |   |   --- tasks
|   |   |       --- main.yml
|   |   |-- volatility
|   |   |   --- tasks
|   |   |       --- main.yml
|   |   --- yara
|   |       --- tasks
|   |           --- main.yml
|   --- site.yml
|-- install_ansible.sh
|-- LICENSE.md
--- README.md

49 directories, 62 files
```

Listing K.1: Ansible directory structure