jamk.fi

Performing remote live response on organizational environment

Joni Ahonen

Bachelor’s thesis
February 2020
Technology
Degree Programme in Information Technology
Cyber Security

jamk.fi

Description

Author(s) Ahonen, Joni	Type of publication Bachelor's thesis	Date February 2020
		Language of publication: English
	Number of pages 75	Permission for web publication: Yes
Title of publication **Performing remote live response on organizational environment**		
Degree programme Information Technology		
Supervisor(s) Saarisilta, Juha Kotikoski, Sampo		
Assigned by JAMK University of Applied Sciences; JYVSECTEC; CYBERDI project; Vatanen, Marko		

Abstract

With the digitalization of society, the traditional crimes have manifested in digital operational environment in cybercrimes, which have created a need for organizations to build a lasting ability to detect and respond to the occurring security incidents. JAMK University of Applied Sciences in association with the Police University College have striven to find means to improve the organizations' capabilities in a cooperation project named CYBERDI. One result of the project has already been published, namely the conceptual cybercrime prevention model, Prepare – Hunt – Response. The model created by JYVSECTEC's specialists helps organizations to improve their capability to prepare, detect, and respond to incidents occurring in organizations' information systems and networks. During the project the model will be supplemented to include a technical environment for cybercrime prevention.

Before the technical environment execution, JYVSECTEC being the assignor of the thesis, had the desire to examine the capabilities of two of existing open source remote live response tools, and performance restrictions when utilized for investigation in an organizational environment consisting of hundreds of endpoints. The assignment was set to be the objective of the research. The research was constructed using two different test cases which produced the results of how the tools' performance. The results verified that the researched tools are capable of gathering data from occurred incident on organizational environment, and usage of the tools in the assignor's technical environment execution is justifiable. However, the tools require additional expertise when performing intensive investigation as they have similar features. All things considered, the researched tools should still be considered as a combination of two tools and when selected to investigate an organization environment, they should be extending each other, not excluding the other.

Keywords/tags (subjects)
Cyber security, cybercrime, live response, incident response, digital forensics

Miscellaneous (Confidential information)

jamk.fi

Kuvailulehti

Tekijä(t) Ahonen, Joni	Julkaisun laji Opinnäytetyö, AMK	Päivämäärä Helmikuu 2020
		Julkaisun kieli Englanti
	Sivumäärä 75	Verkkojulkaisulupa myönnetty: Kyllä

Työn nimi
Performing remote live response on organizational environment

Tutkinto-ohjelma
Tieto- ja viestintätekniikka

Työn ohjaaja(t)
Juha Saarisilta
Sampo Kotikoski

Toimeksiantaja(t)
JAMK Jyväskylän ammattikorkeakoulu; JYVSECTEC; CYBERDI-projekti; Marko Vatanen

Tiivistelmä

Yhteiskunnan digitalisoituminen on aiheuttanut rikollisuuden siirtymistä digitaaliseen toimintaympäristöön. Tietoverkoissa tapahtuvat rikokset ovat luoneet organisaatioille tarpeen kehittää organisaatiokohtaista kyvykkyyttä kyberhyökkäyksien tunnistamiseen, havaitsemiseen sekä niiltä suojautumiseen.

Jyväskylän ammattikorkeakoulu on yhdessä poliisiammattikorkeakoulun kanssa pyrkinyt löytämään keinoja organisaatioiden kyvykkyyksien kehittämiseen CYBERDI-yhteishankkeen avulla. Osana hankkeen tavoitteita on luoda organisaatioille edellytyksiä parantaa sekä tietoisuutta että teknistä kyvykkyyttä kyberrikollisuudelta puolustautumiseen, johon on pyritty jo hankkeen aikana vastaamaan JYVSECTECin, työn tilaajan, julkaisemalla konseptitasoisella mallilla. Mallia täydennetään hankkeen aikana teknisellä ympäristöllä, joka sisältää avoimen lähdekoodin työkaluja, joista osaa voidaan käyttää keräämään tietoa organisaatioympäristössä tapahtuvista poikkeamista reaaliaikaisesti verkon yli.

Työn tilaajalla oli tarve selvittää ennen kahden reaaliaikaiseen tutkintaan tarkoitetun työkalun käyttöönottoa, mitkä ovat työkalujen kyvykkyydet, kun työkaluja käytetään edistyneiden ja kohdistettujen kyberhyökkäyksien havaitsemiseen kohdejärjestelmistä, sekä työkalujen resurssivaatimukset, kun tutkimusta suoritetaan organisaatioympäristössä, joka koostuu sadoista tutkittavista päätelaitteista. Tilaajan tarve asetettiin vastaamaan tutkimuksen tavoitetta, joka saavutettiin käyttämällä kahta testitapausta, joiden avulla voitiin muodostaa tutkimustulokset.

Tulokset osoittavat, että työkalujen käyttöönotolle on riittävät perusteet ja että tulokset ovat vertailukelpoisia, vaikka työkalut pitävätkin sisällään päällekkäisiä ominaisuuksia ja vaativat asiantuntijuutta, kun työkalujen avulla tutkitaan organisaatioympäristöjä.

Avainsanat (asiasanat)
Kyberturvallisuus, kyberrikollisuus, tietoturvapoikkeaman hallinta, digitaalinen forensiikka

Muut tiedot (Salassa pidettävät liitteet)

Contents

Figures

Tables

Acronyms

AES	Advanced Encryption Standard
AFF4	Advanced Forensic Format 4
AI	Artificial Intelligence
API	Application Programming Interface
APT	Active Persistent Threat
ARP	Address Resolution Protocol
BASH	Bourne Again Shell
BGP	Border Gateway Protocol
C2	Command and Control
CBC	Cipher Block Chaining
CLI	Command Line Interface
CPU	Central Processing Unit
CSV	Comma Separated Values
DC	Domain Controller
DDoS	Distributed Denial of Service
DFIR	Digital Forensics and Incident Response
DNS	Domain Name System
DoS	Denial of Service
EU	European Union
FOSS	Free and Open Source Software
GCE	Google Compute Engine
GRR	GRR Rapid Response
GTS	Global Time System
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IoC	Indicator of Compromise
IoT	Internet of Things

IP	Internet Protocol
IR	Incident Response
ISP	Internet Service Provider
IT	Information Technology
IV	Initial Vector
JYVSECTEC	Jyväskylä Security Technology
LR	Live Response
MITM	Man in the Middle
NIST	National Institute of Standards and Technology
ODT	OpenDocument Text
OS	Operating System
PKI	Public Key Infrastructure
R&D	Research and Development
RAM	Random Access Memory
RAT	Remote Access Trojan
RFC	Request for Comments
RGCE	Realistic Global Cyber Environment
RSA	Rivest-Shamir-Adleman
SNMP	Simple Network Management Protocol
SQL	Structured Query Language
SSH	Secure Shell
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TTP	Tactics, Techniques, and Procedures
VBA	Visual Basic for Applications
VCS	Version Controlling System
WMI	Windows Management Instrumentation
YAML	YAML Ain’t Markup Language

Glossary

.dockerignore	Special file of Docker which Docker Engine utilizes to exclude sensitive and unnecessary files and folders from build context of image.
Apache License 2.0	Open source license by Apache, version 2.0.
Cobalt Strike Beacon	Feature of Cobalt Strike penetration testing tool developed by Raphael Mudge which allows modelling threat actors actions using common protocols such HTTP, HTTPS or DNS via payload.
Docker image	Production of Dockerfile which is used as a base when a running instance of image, Docker container, is created.
Docker Swarm	Clustering tool of Docker used for deploying and scaling services on multiple Docker hosts.
Docker	Refers to Docker Engine. Enables the creation, managing and running of containerized applications on any major operating system without dependency issues.
Dockerfile	Special text file, a receipt, which the Docker utilizes when the image is built.
Git	Version Controlling System (VCS) enabling paraller changes to the project to be made and versioning and storing the data as a stream of snapshots.
Hybrid cryptosystem	Involves use of more than one cryptographical method for data encryption, where the public-key cryptosystem is used for key encapsulation and symmetric-key cryptosystem for data encapsulation.
Kubernetes	Orchestration tool for containerized applications.
KYHA19tv	Cyber security exercise planned, organized, and executed by JYVSECTEC for Finland’s national defensive agencies held in fall 2019.
MySQL	Oracle Corporation’s developt database management system which utilizes relational database tables.
Nginx	Server software designed to versatile use such as HTTP web server serving with reverse proxying and load balancing, initially written by Igor Sysoev.
PHR model	Conceptual Prepare – Hunt – Respond model designed by JYVSECTEC’s specialists to help organizations improve their ability to prepare, detect, and response to the incidents.

PowerShell	Powerful command-line shell designed for automated use of systems and advanced system administration tasks invented by Jeffrey Snover, initially targeted for Windows operating systems but latter open sourced and cross-platformed.
REL_DB	Database format which replaced the use of older AFF4 database format in GRR from version 3.3.0.0 onwards.
SQLite	Widely used and efficient SQL database engine written in C programming language with long term support of developers.
Terraform	Tool for descriping infrastructure as a code. Enables the manageable way to build, change, and version infrastructure.

1 Introduction

1.1 Motivation

The digitalisation of society has placed citizens and organizations to face a new manifestation of crime, called cybercrime, in their everyday life. In today's world, a single individual can confront harassment, fraud, abuse, or even get killed by cybercrime committed utilizing digital operational environment. When a cyber-attack is targeted at organization's information systems and networks, it can be vital for continuance of the operation if the critical assets are accessed and revealed by the attacker. (Sihto 2019; Tietoverkkorikollisuuden torjuntaa koskeva selvitys 2017.)

At the same time when the amount of criminal activity has increased in information networks, the amount of reports of an offence made to the Police of Finland has been trailed. The reasons for this can be seen to vary. Single individuals can feel shame about what have happened, and an organization can face image-related problems. On the other hand, a single individual does not necessarily know how to act in this kind of situation, organizations may not have corresponsive processes, or lack knowledge, insufficient observation ability and resource targeting during and after when the security incident occur. (Anttila 2018; Valkama 2019.)

The objective of this thesis, assigned by JYVSECTEC (Jyväskylä Security Technology) is to examine and test the capability and performance requirements of two preselected open source tools, GRR Rapid Response and osquery, designed for performing remote live response. The objective is reached using two different test cases that measure both the tool's ability to gather data from endpoints to identify and contain assumed information security incident from the organizational Information Technology (IT) environment simulating a real-world organizational environment, and performance requirements that the tool sets for the underlying hardware when used for active investigation on organizational environment that includes hundreds of endpoints.

The cyber-attacks used in test cases are based on Techniques, Tactics, and Procedures (TTPs) that Active Persistent Threat (APT) actors have executed on their campaigns in the past years. Promising results lead the implementation of the tools as a part of a larger cybercrime prevention environment that will be designed and built during the CYBERDI project’s cybercrime prevention working package in association with Police University College, the Finnish police education facility.

1.2 JYVSECTEC – Jyväskylä Security Technology

According to the JYVSECTEC’s official website (JYVSECTEC overview n.d) JYVSECTEC is an independent cyber security research, development, and training center located in Jyväskylä, Central Finland. JYVSECTEC is a part of the Institute of Information Technology at JAMK University of Applied Sciences, and nationally it is one of the leading operators in the domain of cyber security, as well as a globally noticed partner. JYVSECTEC operates Finland’s national Cyber Range and is the assignor of this thesis. (Finnish cyber security expertise in Singapore – JAMK University of Applied Sciences and Singapore Polytechnic collaborating in Centre of Excellence in Applied Cyber Security 2019; JYVSECTEC organization 2019; JYVSECTEC overview n.d.)

One of JYVSECTEC’s objectives is to offer high-quality cyber security exercises for Finnish national government’s defensive agencies and organizations operating under public and private sector. Along with planning, providing, and executing the exercises, JYVSECTEC participates in numerous research and development (R&D) projects as a part of the Institute of Information Technology at JAMK University of Applied Sciences, executes system and software testing, consulting, offering also lightweight certification programs. (JYVSECTEC services n.d.)

The cyber security exercises organized by JYVSECTEC utilize the unique Realistic Global Cyber Environment (RGCE). RGCE simulates the real-world Internet, its core functionalities, and services as accurately as possible. For instance, RGCE consists of implementation of following protocols and systems:

- Border Gateway Protocol (BGP) routed network
- a real-world public Internet Protocol (IP) addressing
- hierarchically implemented Domain Name System (DNS)
- Global Time System (GTS)
- Public Key Infrastructure (PKI). (JYVSECTEC Cyber Range – RGCE and solutions n.d, 2–3.)

To create a more realistic environment, use of different public services such as social media platforms Twitter and Facebook, application store, news media and instant messengers are resolved in RGCE. In addition, RGCE includes numerous organizational environments from the field of business and industry as well as Internet Service Provider (ISP) and cloud providers, each of them with unique sectoral and corporative features. In this research, two of the RGCE's organizational environments were used to simulate a real-world organizational environment in test cases. (JYVSECTEC Cyber Range – RGCE and solutions n.d, 3–11.)

One of the biggest advantages of RGCE is that it is fully isolated from the production network, i.e. RGCE's Cyber Range can be thought of as a digital shooting range, where the organizations participating in the exercise have the possibility to practice offensive as well as defensive actions in real-world circumstances without insulting the existing laws or regulations. In Cyber Range, a large collection of different real-world cyber threat attack vectors typical to a specific threat actor have been implemented. There are also modelled single person actors such as script kiddies, small hacktivists and activists' groupings with political or other agenda as well as several national APTs and organized criminal adversaries. Depending on the threat actors' capability, it is possible to execute threat vectors such as variant Distributed Denial of Service (DDoS) attacks, ransomware and phishing campaigns, watering holes, malware variants originating from different malware families, as well as viruses, worms and Remote Access Trojans (RATs). (ibid., 2–4.)

1.3 Project CYBERDI

The name of CYBERDI project stands for the sentence "Cybercrime prevention, awareness raising and capacity building by RDI on modern cyber-attacks". CYBERDI is an R&D project funded by the Ministry of Education and Culture of Finland and administered by JAMK University of Applied Sciences with the association of Police University College, which is the Finnish police education facility. The total duration of the project is three years and it is executed in years 2018–2021. The project has been divided into three different working packages: cybercrime prevention, awareness raising, and capacity building. (5 miljoonaa euroa ammattikorkeakoulujen soveltavaan tutkimukseen ja innovaatioihin 2019; CYBERDI – Kansallista & kansainvälistä kyber-osaamista kasvattamassa n.d; CYBERDI n.d.)

In cybercrime prevention, the project executors' intend to enhance their capability to prevent cybercrimes in modern digital environments. Prevention is built using the latest technologies such as Artificial Intelligence (AI), machine learning, and data analytics. Furthermore, new effective cooperation models are developed for cybercrime prevention and investigation to ease collaboration between different authorities. The research performed in this thesis is published as a part of the cybercrime prevention working package. (CYBERDI – Kansallista & kansainvälistä kyberosaamista kasvattamassa n.d.)

In awareness raising the focus is to increase the target audience's knowledge about the digital world around them and threat landscape that it brings. The target audience consist of organizations that operates in Finland's public and private sector and healthcare organizations as well as students in upper secondary education and comprehensive school. (CYBERDI – Kansallista & kansainvälistä kyberosaamista kasvattamassa n.d.)

In capacity building the cooperation with the partners from national and global networks is expanded to strengthen already existing partnerships and create new viable productive stakeholders across the globe, especially in the region of European Union (EU). The objective is to reinforce the already existing profiles of JAMK University of Applied Sciences and JYVSECTEC, when speaking of actors in the cyber security domain. (CYBERDI – Kansallista & kansainvälistä kyberosaamista kasvattamassa n.d.)

2 Research design

2.1 Research problem

One of the CYBERDI project's results has already been published by JYVSECTEC, namely the Prepare-Hunt-Respond conceptual model (PHR model). PHR model has been designed to ease the ability of organizations to absorb and understand how to successfully detect, defend, and build efficient countermeasures against modern cyber-attacks. During the project, the model will be also developed to include a technical environment for the use of organizations consisting of a collection of tools, the source code of which has been published under open source license. (PHR model 2019.)

JYVSECTEC in the role of the assignor of the thesis has the problem that the tools need to be examined and tested extensively before they can be accepted and applied as a part of the PHR model's technical execution. The scope for the tool survey is set to include investigation and documentation of the technical capabilities and limitations as well as performance requirements of the tools so that it is possible to accomplish the applied execution. The absolute goal of the assignor is to find the best possible tools to tackle a specific field of the PHR model. The tools should also have integration possibilities and least possible overlap with other selected tools.

One of the fields of the PHR model is Triage and Respond which involves an organization's use of digital forensic methods and incident response processes in a detailed investigation during the incidents (PHR model 2019). For the technical environment execution, the assignor had preselected two different existing tools, GRR Rapid Response and osquery, which should cover both digital forensics and incident response actions, when the remote live response is performed. However, before the tools can be approved and applied as a part of the PHR model's technical environment execution, the research problem needs to be set and answered first. Hence, the research problem for this thesis is defined using a set of research questions:

- How well and reliably can the selected tool be used to identify and contain incidents from the selected organizational environment?
- What are the performance requirements when the selected tool is used for investigation performed in organizational environment that consists of hundreds of endpoints?

2.2 Research method

The research made in this thesis should be considered as an applied research, since it solves problems and find solutions in a practical manner. Furthermore, in research the specified tools are tested, and research is targeted for the assignor and its needs. In addition, the generated results should offer added value when practical execution based on the research is made or further research executed. On the other hand, the research relies to the researcher’s empirical concrete observation and for this reason the research involves use of empirical research methods. (Baimyrazaeva 2018, 6; Hirsjärvi, Remes, & Sajavaara 2008, 128–129.)

According to Baimyrzaeva (2018, 17–38) applied research should be considered as a process involving the research to be executed in five steps:

1. Clarify your research focus
2. Scan existing information
3. Plan your research tasks and methods
4. Collect, analyse, and interpret data
5. Share your work.

The base for the research work is constructed according to what is the objective of the research to be made (Baimyrzaeva 2018, 18). In this research, the objective is to find answers to the found research questions in a way that the produced results are reliable and can be used when applied execution or further research is considered. Therefore, the research questions dictate the base and direction for the research, and the construction of research methodology. The direction of the research is modelled and focused during the research to provide more accurate results. However, the main objective of the research, answering the predefined research questions, remains as the same.

To get better perception of the researched field, the research is started by examination of the collection of scientific researches, papers, and resources as well as tools technical documentations published by the developers of the tools. The source criticism is used when the reference material is gathered for the research. Based on the gathered material, a literature review is included to the research. The literature review is the theoretical base of the research and includes a presentation of the existing theory in the scope of the research and provides overview to the technical operation of the researched tools.

Research is continued by executing the initial tool deployment in local test environment to get early perception of the tools. The tools’ caveats and possible development targets are discussed with the assignor. If any tool development targets are recognized they are to be solved before the research is continued.

Research objective is achieved by executing two different test cases that are constructed with an aim to produce accurate results for the research questions. In test case construction, the used testing environments in RGCE are taken into account and are resolved before any further construction work for the test cases is executed. There is discussion with JYVSECTEC's specialists who are also consulted when the testing environment selections are made to improve the quality of test cases.

After the testing environment selections, the tools are deployed in the environments on RGCE where the real-world organizational environment circumstances exist. A suitable construction for the tools in the testing environments is resolved in cooperation with the assignor, since the used environments in RGCE are controlled by the assignor and involve the assignor's administration, expertise, and consultation. This also applies to the tool deployment. Before and during the deployment process of the tools, the requirements and other expectations set by the assignor of the thesis are resolved and respected. This provides transparency between the parties during the tool deployment process.

The first test case, capability testing, is constructed to answer the capability related research problem. To produce reliable and accurate results, the real-world APT level cyber-attack is generated on testing environment, a real-world organizational environment. Use of realistic adversary techniques enables the investigation made using the tools to be executed in amidst of real-world security incident.

In the second test case, performance requirement testing, the components of the GRR Rapid Response server are monitored using pre-existing advanced network monitoring solution. In this research the monitoring focuses on the server-side of the tool and the monitoring of GRR and osquery agent is excluded. The reason for this is twofold. First, when research is made only to the GRR's server-side, it enables the research to be more targeted. Targeting behalf provides more accurate results for the research question related to performance requirements. Secondly, the parties of the research are aware that there is already research made about the performance constraints and footprint of GRR's agent, and the research is available for the use of the assignor (Moser & Cohen 2013). Furthermore, GRR agent resource usage can be controlled on server-side in considerable manner. This also applies to the osquery agent, since in this research the osquery agent is used and controlled by GRR Rapid Response server and should remain within the same restrictions that are set for the GRR agent.

As mentioned, the execution of test cases enables the objective of the research to be reached and provides results for the research questions. The criticisms and objective standpoint are used during the test cases and in presentation of the results to avoid any human errors. In other words, the presented results answer only the research questions and any ambiguousness is omitted. Based on results, it is possible to determine how well the research questions can be answered and deduce the success and reliability of the research and concern possibilities for further research.

3 Digital forensics and incident response

3.1 Incident response process

As the name of the term suggests, incident response or in shorter IR is an organized and premeditated way to respond to security incidents occurring in organizational environments. The incident itself can be classified as any malicious activity that exceeds an organization's security policy. The objective of the incident response is to decrease the incident's impact so that its severity can be reduced and the recovery process accelerated. (Luttgens, Pepe, & Mandia 2014.)

Incident response is executed as a process called incident response process, which is carried out by an organization's team, often designated for only this purpose. Incident response is a continuing variegated process and in every organization where the incident response process is implemented, the process takes into account the organization's needs, resources, and procedures, meaning that the incident response process is always organization-specific way to respond occurring security incidents. The incident response process is self-developing and every incident occurring in the organization's environment should develop and improve it. (Luttgens et al. 2014.)

Regardless of how the incident response process methodology is applied to an organization, it is normally seen as a cycle of procedures involving actions from an incident response team assigned to the process as well as organization-wide actions. National Institute of Standards and Technology (NIST) (2012) suggests that the incident response process should be executed as a four-step cycle process:

- Preparation
- Detection and analysis
- Containment, eradication, and recovery
- Post-incident activity.

The detection and analysis phase involves that the organization has a general procedure to detect and analyze the most common incidents occurring in organization's environment, since the preparation for every possible incident bonds unnecessary resources and is not a lasting solution. When the organization has a described and organized procedure to recognize the typical attack vectors used by adversaries and detect the common IoCs (Indicator of Compromise) from the organizational environment, the possible occurring or an already occurred incident can be detected earlier, and the investigation started and recovery from incident is accelerated. (NIST – National Institute of Standards and Technology 2012.)

3.2 Digital and computer forensics

The term forensics originates from Latin and according to Merriam-Webster dictionary (n.d), the definition for the word is "application of scientific knowledge to legal problems". Forensics, in a term as it stands, does not take into account what the scientific knowledge brought to courts of judicature is or what is the field of forensics or the methods that has been used. When forensic science involves evidence collection from digital devices or more specifically from computers, the terms digital forensics and computer forensics are used. (Graves 2013; Reith, Carr, & Gunsch 2002.)

Digital forensics intend to find traces about criminal activity in a digital operational environment such as digital devices and networks so that the emphasized knowledge of what was happened can be proved and used as evidence in court. Digital devices can be classified to include all those physical computing devices that use electronic signals. Beyond these devices, everyday devices such as laundry dryers, which are traditionally not seen as digital devices, are now becoming digitalized by the Internet of Things (IoT). Appending IoT devices to the list of digital devices expands the list of potential devices for digital forensics drastically. (Bourgeois 2019; Marcella & Menendez 2008; Reith et al. 2002.)

Speaking of computer forensics, the investigation is targeted at computers; however, the means are the same as in digital forensics: to find evidence from malicious actions that are accepted in court (Reith et al. 2002). When crime scene investigation is started by first responders the National Institute of Justice (2008) suggests that the following digital devices should be considered as potential evidence:

- Computer systems
- Storage devices
- Handheld devices
- Peripheral devices
- Computer networks
- Other potential sources of digital evidence.

As incident response, digital forensics is executed as a predefined process. According to Johansen (2017) digital forensics process involves investigator to execute investigation in six steps: identification, preservation, collection, examination, analysis and presentation. Johansen also informs that the digital forensics process is an important part of incident response process; when applied, digital forensics process is a component of investigation workflow which should deliver knowledge about what has happened and bring evidence to the investigation that can be used connecting the party behind the incident to the incident. (Johansen 2017.)

3.3 Data acquisition

As mentioned, during computer forensics process the investigator intend to find evidence from computer devices. The process driven by an investigator usually involves taking a duplication from the target device such as hard drive or memory. Imaging allows the investigator to accomplish forensic investigation to the duplicated target without affecting the device’s original state. However, improvements that the digital devices have faced have brought new methods of how the investigation is executed, since imaging is a time consuming process and when the number of investigated systems grows, the forensic process involves more and more time. (Luttgens et al. 2014.)

Live response, or in shorter LR, refer to the real-time data collection for the use of investigation process. The gathered data vary; however, the means is the same; with live response an investigator intend to gather volatile data from a target system that can otherwise be lost. Live response also answers to the challenges that the traditional forensics produces to the investigation by forensic duplication, since live response intend to gather data in real-time, answering the investigation related questions more rapidly and without delaying the continuance of investigation process. (Luttgens et al. 2014.)

According to Luttgens, Pepe, and Mandia (2014), performing live response can be considerable when following observations in investigation are recognized:

- Volatile data consists of data that can not be investigated using other methods
- The change to the target system is controllable and as minimal as possible
- The number of contaminated systems is large
- The performed imaging preserve time and possibility for failing exists
- Legal or other considerations require as much data as possible.

However, the live response should be considered harmful if the process of acquisition of volatility data is not well automated and described, and the impact of the tools is unknown. Live response tools used for data gathering inevitably change the target system's state and without precise knowledge about the used tool's impact to the target system, live response can paralyse, mislead or damage the investigation process. For instance, Walters and Petroni inform (2007) that specific tools used in live response research have changed the system state significantly: how drastic change to the system was, Walters and Petroni notices that the impact is bigger than letting the system run 15 hours incessantly. In some cases, live response might also be the ringing bell for the attacker in system to go undercover. (Luttgens et al. 2014; Walters & Petroni 2007.)

Brezinski and Killalea (2002) inform in Request for Comments (RFC) 3227 that the data should be gathered in the order of its volatility. According to Brezinski and Killalea, the data collection should be started from registers and cache and continued with network related data such routing tables and Address Resolution Protocol (ARP) tables. In addition, the processes and temporal file systems are to be gathered before the hard disk, logging and monitoring data is acquired. The least volatile data to gather is the systems physical configuration and archive medias. However, the knowledge of what to acquire and in what order is not enough; also the individual features of a target system such as the running operating system (OS) are needed to take into account when data is gathered. (Brezinski & Killalea 2002; Lutggens et al. 2014.)

Live response is executed locally at the same physical location where the target system or systems reside. However, when the number of investigated systems increases and the physical location varies, performing live response becomes challenging. However, remotely performed live response can be used for data acquisition over the networks. Remote live response intends to gather the same volatile data from the target system as live response; however, the investigation can be performed using network connections whether or not the investigator has physical access to the system. (Johansen 2017.)

4 Technical review

4.1 Osquery

4.1.1 Operational overview

Osquery is an open source agent-based tool designed for endpoint instrumentation. Osquery is written in C++ programming language and it is designed and maintained by Facebook. Along with Facebook, also other contributors have participated in the development work of the tool. Osquery was introduced and published to the crowd in June 2014 when the Facebook made the first announcement of the tool that has capability to model and describe the OS in which the tool is deployed as a relational database by making the initial commit to the osquery's Git project repository. By constructing the Structured Query Language (SQL) based queries, osquery can be used to retrieve information about abstract concepts such as established network connections from the underlying OS in an efficient manner. (Chacon & Straub 2014; Osquery documentation n.d; Osquery n.d; Sereyvathana & Reed n.d.)

Osquery consists of two separate instances of which the osquery's interactive shell, osqueryi can be used to retrieve information about the current OS and its changes. The use of osqueryi comes in handy when a new constructed query needs to be evaluated without affecting the existing configurations, since it does not involve the use of daemon in its working and can be considered as a standalone version of the tool. The daemon instance of the tool, osqueryd, gathers information from the endpoints, and it can be queried, logged, scheduled and monitored by changes in a more efficient and scalable manner. The current version of the tool (version 4.0.2, released on 13 September 2019) includes in total 232 different tables that can be used to retrieve a large scale of distinct information about the target system, and it changes by constructing and executing the queries. (Osquery documentation n.d; Schema n.d; Osquery n.d.)

Osquery has been designed to run on every major OS used in enterprises such as Windows, Linux, MacOS, and FreeBSD. Regardless of what the underlying OS is, the same query, utilizing for instance the table that can be used to retrieve the currently logged in users, made on every supported OS should retrieve the report with the same structure. What is worth of noticing is that the current version consists of 40 tables that can be used regardless of what is the underlying OS. The major part of the tables has OS related features and tables are not supported other than in one OS; however, the way how the queries are constructed remains the same. (Osquery documentation n.d; Performant endpoint visibility n.d.)

4.1.2 Query structure and management

The construction of a correct query structure involves using osquery's implementation, a subset of SQLite. Osquery's query construction starts from using the traditional SQLite statements. However, in osquery the only statement with direct effect on queries is the SELECT statement. After SELECT statement, the construction is continued by selecting the desired rows, records from tables, which are selected using FROM clause after the record specification. The specification for retrieved data is performed using clauses such as WHERE. In addition, it is possible to perform the normal use of SQLite wildcards. (About SQLite n.d; Osquery documentation n.d.)

Osquery enables the joining of distinct tables. For instance, if it is desired to acquire those processes executed by a specific user, osquery tables *processes* and *users* can be joined in query to retrieve the desired result as concatenation of two tables. For construction of queries osquery provides a detailed schema documentation, which is useful when monitoring is performed efficiently and in a more intensive manner. (Osquery documentation n.d; Schema n.d.)

As mentioned, the execution of single queries is viable when queries are evaluated, and interactive instance of the tool used. When performing more sustained and robust monitoring, for instance in organization environment, the management of queries can easily become challenging since the number and interval of queries raises inexorably. However, the daemon instance of tool, osqueryd, allows packing distinct queries by default in filesystem plugin, a specific configuration file, which enables growing the number of used queries without losing the tool's controllability. In addition, the scheduling and logging related configuration is added to the configuration file. (Osquery documentation n.d.)

4.1.3 Deployment considerations

Deployment of a single instance of osquery agent is a straightforward process. The project offers distinct installation packages and executables for every supported OS. Alternatively, compiling the tool directly from the source code is possible and is the developer recommended way for the tool deployment. (Osquery documentation n.d.)

Osquery agent is installed and deployed on every endpoint that is desired to be monitored. In small deployments controlling agents can be managed via a remote connection such SSH (Secure Shell). However, when the number of monitored endpoints rises, the manual management of agents becomes more difficult. When the system monitoring is about to be executed in a large scale, for instance on an organizational environment, the use of alternative solutions is required. As the osquery's official documentation (n.d) informs, osquery does not offer by default any centralized fleet manager for the agents. However, few third-party solutions exist and are available for the fleet management, the implementation, suitability, and use of which is left on deployer’s own consideration. (Osquery documentation n.d.)

Managing every endpoint in an organization without centralized access to the system to be monitored is not a lasting and viable solution when speaking of remote live response. One solution for controlling osquery agents in endpoints for the use of remote live response is to use GRR Rapid Response, which allows gathering data and composing results from the osquery agents on endpoints in a centralized way without changing the tool’s default usage, constructing queries. (Osquery documentation n.d.)

4.2 GRR Rapid Response

4.2.1 Operational overview

GRR Rapid Response, later GRR, is an open source incident response framework written in Python 2.7 programming language. The development work of GRR was started in 2011 by Google with an aim to create a state-of-the-art tool that meets the requirements set for a cross-platform and scalable incident response framework focusing on remote live response. Google has committed a long-term support for the GRR and since 2011 it has been continuously developed and maintained by Google's full-day software engineers and other contributors, currently being at version 3.3.0.8 (released on 9 October 2019). (GRR Rapid Response n.d; GRR Rapid Response documentation n.d.)

According to GRR’s official documentation (n.d), the main features of GRR are (only features viable for the research are listed):

- Written in Python 2.7 programming language (Python 3.6 written version was released in the end of the year 2019)
- Consists of two main parts: server and agent
- Allows a powerful investigation to be made remotely to the organization environment over the network
- A cross-platform agent that supports the most common operating systems (Windows, Linux, MacOS)
- A fully functional API (Application Programming Interface) that can be used to manage the incident responders' everyday tasks across the fleet of agents

- The scalability of GRR has been tested on around 50 k client machine environment by Google and in smaller deployments made by other organizations
- Due to open source code, GRR can be implemented and modelled into a different environment with different requirements. (GRR Rapid Response documentation n.d.)

GRR server and agent utilize the traditional client-server model in bilateral communication. GRR server is a centralized server used to launch forensic tasks on the clients over the network. Respectively, GRR agents are installed on those endpoints, or in other words, on clients desired to be examined. GRR utilizes the unique messages for the communication between server and clients, and all message exchange is encrypted. Figure 1 explains GRR's operation mode from the start of a single request-response investigation workflow to the view of results: (Cohen, Bilby, & Caronni 2011; GRR Rapid Response documentation n.d.)

1. Investigator starts a new desired investigation workflow on a GRR server for a desired client or fleet of clients on investigation domain where the GRR agent is successfully installed
2. GRR server serializes, encrypts and signs required workflow and appends it to the client specific message queue
3. GRR agent installed on the client polls the GRR server for assigned message queue periodically for addressed workflows
4. When the addressed workflow is successfully added to the queue, the GRR agent will then request the message from the server to the client in the next time period using a signed HTTP (Hypertext Transfer Protocol) GET request
5. The GRR agent will then decrypt the signed workflow and start the requested investigation workflow on the client
6. After the task is executed, GRR agent will then encrypt the results to a message and send them as a reply to the GRR server with an appropriate signed HTTP POST request
7. GRR server decrypts the serialized message and informs investigator for the results.

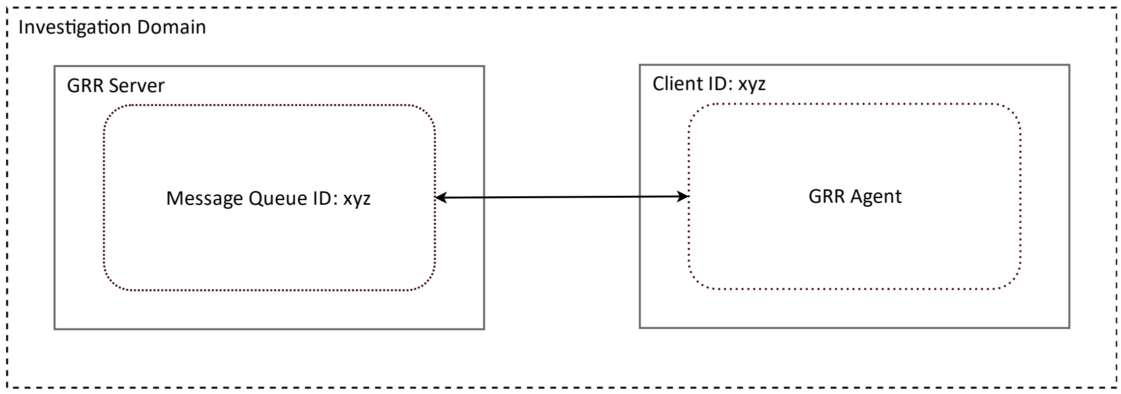

Figure 1. GRR operation mode illustrated (adapted from Cohen et al. 2011)

4.2.2 GRR server and agent

The operation of GRR server has been divided between four distinct logical components: HTTP front-end, worker, user interface consisting of web-based Graphical User Interface (GUI) and API endpoint, and datastore. Every component has their own special purpose; front-end handles the HTTP based communication between GRR server and agent, worker is used for flow processing, user interface to the administrative purposes and investigation creation, launch, management, and automation, and datastore to store the collected results and messages generated during the component communications. (Cohen et al. 2011; GRR Rapid Response documentation n.d.)

Since the scalability is one of the main features of GRR, the server components are designed to be executable as separate processes and multiple instances to meet the requirements set by the environment where the tool is used for and the intensity of incident's responders team actions. By the nature of components, the load between distinct components varies. To avoid a possible bottleneck emergence on those components that fall under heavy load, the reasonable resource targeting and scaling should be considered when components are deployed. Accordingly, those components that fall under light load are not so critical when speaking of bottlenecks, and scaling of which can be considered in situations when redundancy is desired. (GRR Rapid Response documentation n.d.)

For instance, if an organization's incident response team is going to use GRR intensively and investigation domain consists of thousands of clients, it is advisable to consider running distinct HTTP front-end and worker processes as multiple instances and deploying reasonable datastore, which should scale linearly. Accordingly, even if GRR is deployed for the use of an active incident response team, there is no necessary need to run multiple user interface instances as the component is under light load; however, this might be considerable and reasonable if redundancy is needed and investigation team consist of multiple concurrent users. Using reasonable resource targeting, every component can be correctly dimensioned and built to meet investigation and environmentally implemented needs. (GRR Rapid Response documentation n.d.)

As mentioned, the GRR server uses a centralized database to store formalized data. From the version 3.2.4.5 (released on 17 December 2018) onwards the support for MySQL database management system was released, and the support for its predecessor SQLite was deprecated. However, as the developer team announces on GRR's official documentation (n.d), due to the version updates the use of distinct backend system is also possible such as Google's Bigtable datastore. Furthermore, from the version 3.3.0.0 (released on 22 May 2019) onwards the old, default Advanced Forensic Format 4 (AFF4) data storing technology has been replaced with REL_DB data format to bring stability and performance improvements. Even if the REL_DB is not backward compatible with AFF4, the AFF4 can still be used instead of REL_DB if desired; however, this is discouraged by developers for beforementioned reasons. (Cruz, Moser, & Cohen 2015; GRR Rapid Response documentation n.d; Moser & Cohen 2013; What is MySQL? n.d.)

GRR agent is a Python 2.7 written piece of program supporting common enterprise used operating systems: Windows, Linux and MacOS. After the first deployment of the GRR server adding a new client to GRR's investigation domain is a simple and straightforward process. If configured, the GRR server produces the GRR agent installation binary with the corresponding configurations and populates the desired location with generated installation packages. If changes to the GRR server configuration are made, repacking of the installation packages can be made and updates on clients perform. (GRR Rapid Response documentation n.d.)

Updates on agents can be delivered using the specific administrative flow designed for replacing the obsoleted agent with updated or performing agent reinstallation manually. In addition, when the forensic task is created, the GRR agents' resource usage on clients can be managed using flow or hunt specific trace holds. (GRR Rapid Response documentation n.d.)

4.2.3 Client-Server communication

GRR agent and server communication are based on top of HTTP protocol, and all transmitted data between GRR server and agent are serialized. GRR utilizes Google's protocol buffer (protobuf) data serialization mechanism for a data serialization. Protobuf allows serialization of data regardless of the used programming language or platform. In fact, the messages exchanged between GRR agent and server are a protobuf encoded description of the flow execution containing fields such as session identifier, name and arguments as well as request and response identifier. (Developer Guide n.d; GRR Rapid Response documentation n.d.)

As mentioned, GRR encrypts all message exchange between client and server. According to the GRR Rapid Response's source code (2019), encryption is performed using a hybrid cryptosystem constructed from a key encapsulation scheme and data encapsulation scheme. The source code (n.d) indicates that GRR uses symmetric-key cipher algorithm AES (Advanced Encryption System) in CBC (Cipher Block Chaining) mode with 128-bit key length and IV (Initial Vector) as key encapsulation scheme. The RSA (Rivest-Shamir-Adleman) keys are used as a data encapsulation scheme to bring protection against symmetric key delivery problems. (Cramer & Shoup 2003; GRR Rapid Response n.d.)

As the developer of the GRR, Ogaro (2019a) informs, following phases describe how hybrid cryptosystem is implemented on message exchange between server and agent when the GRR server requests a forensic task to be performed on a client:

1. Both GRR server and agent have their own RSA, by default a 2048-bit, public-private key pair used for signing and encrypting, public keys are exchanged
2. Server generates a new random AES 256-bit session key
3. Server encrypts protobuf message which holds the forensic task using the AES key
4. Server encrypts the AES key and signs the message using client's RSA public key
5. Agent fetches the encrypted message and key from the server's message queue
6. Agent decrypts the AES key and verifies the signed message using the client's RSA private key
7. Agent decrypts the message using the decrypted AES key
8. Agent starts the request forensic task on the client and replies to the server with appropriate encrypted message. (Ogaro 2019a.)

The message exchange is started when GRR server requests the GRR agent on client to perform a task such as forensic data acquisition. The composed message is labelled with request identifier as illustrated in Figure 2, which is incremented between distinct flows. After the message has been composed, server sets the message to the client's message queue. When the agent on client has fetched the message from the assigned message queue and resolved the requested task, the agent then constructs and delivers the response message to the server server using HTTP POST request. Before transmission to the server, the client composes and labels the message with corresponsive request identifier and response identifier. (Cohen et al. 2011; GRR Rapid Response documentation n.d.)

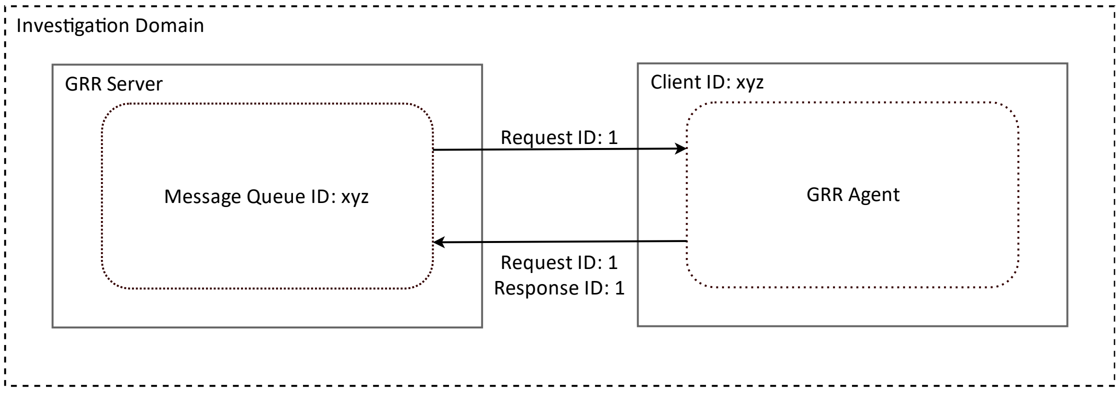

Figure 2. Message exchange illustrated (adapted from Cohen et al. 2011)

If the required task needs multiple client responses, the agent increases the response identifier by one between responses. In such scenarios, GRR server waits the special final response, a status message, which informs the server that the requested task is terminated successfully or with errors. If termination ends with error, the corresponsive traceback is included into the message. Otherwise, the corresponsive results are shown to the user and the message exchange has succeeded. (Cohen et al. 2011; GRR Rapid Response documentation n.d.)

4.2.4 GRR concepts

Flow is a logical task that can be used to perform a forensic or administrative operation on a desired client agent. For example, task specific flow can be used when an incident responder wants to start a new investigation from finding out if a specific file has been seen on a specific client on the environment by listing the target file system. The organizational networks can easily consist of a thousand of clients and launching this kind of search on every client machine would bond unnecessary resources and time if only targeted investigation is desired. As in the beforementioned case, the incident investigation is often started from a certain point of organization environment and then extended to consider the larger fleet of clients to find out the spread of an incident. In fact, flows can be copied to hunt to make the same research on every or specific subset of clients on investigation domain. (Cohen et al. 2011; GRR Rapid Response documentation n.d.)

Configuration and flow management is handled via a web-based GUI or utilizing the API endpoint via CLI (Command Line Interface). GRR includes a large collection of preinstalled flows for the most common forensic tasks. Executing a flow produces results that the investigator can examine directly from the GUI or CLI or exporting them to an archived file using common file formats such CSV (Comma Separated Values). Additionally, investigators can write and import their own Python scripts which can be used via a flow on GRR that is designed for this purpose. (GRR Rapid Response documentation n.d.)

To avoid the excessive tying of recourses, the developers of GRR have designed flow to be an asynchronous task, a state machine. In client-server model this means that the resources are freed from the flow execution on the server-side when the flow is launched on a desired client agent. This enables server resources to be used elsewhere before the client responses and flow execution is continued. (GRR Rapid Response documentation n.d.)

Simply put: hunt is an extended flow. As mentioned, every terminated flow can be copied to a hunt to start the same investigation using the same parameters, for instance on every client or a subset of clients in the environment. If the beforementioned scenario leads to IoC which it is defined to be originated from malicious action, the incident responders' next step in investigation process would be to define all those client machines in the organization's environment where the IoC has been seen and perform a suitable analysis for the acquired artefact. Using hunt, exploring contaminated clients from the environment is efficient and should reduce the time what is taken to conducting contaminated machines from the environment and accelerates the start of further investigation. (GRR Rapid Response documentation n.d.)

As mentioned, hunt can be useful in cases when clarifying the spread of a certain incident needs to be made. Furthermore, GRR's implementation of scheduled jobs allows the use of scheduled hunts. If the investigator wants to perform proactive hunting in an organization's environment, every hunt can be configured to perform a certain activity on desired clients as scheduled in specific periods of time range. (GRR Rapid Response documentation n.d.)

4.2.5 Security considerations

When speaking of security, the GRR's official documentation (n.d) informs that the GRR should be taken as a tool that has security considerations needed to be recognized and resolved before the tool is deployed for production use. By default, without any security consideration, the GRR enables possibilities for harmful actions. (GRR Rapid Response documentation n.d.)

The GRR server can be connected, and a new client added to the investigation domain by using any server generated GRR agent and correct server address. This means that if the adversary gets its hands on the GRR agent installer, it can use the adversary-controlled agent to inspect what the forensic tasks are that are performed by a trusted investigator on investigation domain when an environment-wide hunt is performed. Using this knowledge, the adversary can gather information about the investigated client systems of the organization environment and from the investigation methods. (GRR Rapid Response documentation n.d.)

Investigation information can be useful for the adversary in scenarios, for instance, when the adversary already has access to the target environment and intends to stay under of investigation radar or monitor if the means of the adversary are already resolved by the investigator. What is more, if the adversary has privileged access to the organization environment and clients, the execution of agents can be stopped or disabled. In some scenarios, adversary-controlled agent can be used to send arbitrary messages such as error messages to the server, and if succeeded, raise the possibility of a potential DoS (Denial of Service) attack, the exhaustion of resources on the server so that the server's normal operation aborts. (GRR Rapid Response documentation n.d.)

Other consideration when speaking of security is to resolve how the authentication is handled in GRR when GUI or API endpoint is reached over the network. By default, GRR's administrative GUI can be reached using HTTP protocol and server's IP address and desired web browser, and the API endpoint with developer provided Python library. However, the used HTTP protocol does not provide any encryption for the transmitted data, and all information such as login information is delivered as a plaintext. Adversary using this knowledge can sit between the client and server, called man-in-the-middle (MITM) attack and inspect network traffic for the legitim credentials from the transmitted data, which then can be used to successful login on the GRR's investigation domain. Along with the implementation of secure protocols for the use of authentication, what also should be considered are the correct constraints to and from where the GUI and API endpoint can be connected. (GRR Rapid Response documentation n.d.)

5 Deployment work

5.1 Starting point

The deployment work of the researched tools, GRR Rapid Response and osquery, was executed first to receive an initial and general perception about the installation process and use of the tools in practice. Furthermore, the assignor introduced a list of requirements needed to be resolved before the tools were deployed on the RGCE's organizational environments and implemented to the test cases:

- The deployment process of GRR server should be simple and straightforward, and it should be handled using Docker images (Docker Glossary n.d)
- The containerized GRR should use the latest version of the official GRR image
- The usage of a containerized GRR should also be possible in offline environments
- The containerized GRR should offer good scalability and added security.

The list of requirements focused only on the server-side of GRR since it was observed that the deployment of GRR and osquery agents is a straightforward process. In addition, the agents do not have or produce any restrictions that should have been resolved when implemented on RGCE's organizational environments.

GRR project offers an official Docker image which can be used to deploy the GRR server’s logical components’ HTTP front-end, worker and user interface within a single Docker container. The official image includes also a bundled MySQL database for data storing. However, as the developer team has been noticed on GRR’s official documentation (n.d), the use of the official image is ideal only when the tool is deployed for testing or evaluation purposes. According to the developer of GRR, Ogaro (Ogaro 2019b), this originates from the fact that the component bundling in one container removes the scalability of a single component.

It is worth noticing that GRR server can already be deployed and scaled when the server is compiled directly from a source code or installed from software packages. These installation methods allow the deployment and execution of logical component instances multiple times and on demand in distinct physical servers. However, from the perspective of the research, the beforementioned options were excluded, since the requirements for the tool development set by assignor demanded the use of Docker images. (GRR Rapid Response n.d.)

Initially, the idea was to divide the official image in a way that each logical component of the GRR server can be executed in their own Docker container. The prediction was that if the division of components success, it should offer a better scalability than the official image allowing usage of the containerized GRR also in large organizational networks and the ability to perform investigation more intensively when an orchestration solution such as Docker Swarm or Kubernetes is used for scaling. (Docker Glossary n.d; Production-Grade Container Orchestration – Automated container deployment, scaling, and management n.d.)

5.2 Research of publicly available resources

To get the best result for the GRR official image division, publicly available resources were searched to examine how other developers have deployed GRR in their environments. During the research process Spotify free and open source software (FOSS) team's blog post was researched. Spotify’s FOSS team announced on their engineering and technology blog post how the team has built a Terraform module for a GRR server and implemented it in a Google Compute Engine (GCE) successfully. (Introduction to Terraform n.d; Whacking a Million Moles – Automated Incident Response Infrastructure in GCP 2019.)

Spotify's Git project repository (2019) on their behalf revealed that the team has also built a working testing environment successfully containing Dockerfiles for HTTP front-end, worker, and user interface components of GRR server. In addition, the Dockerfiles use the official GRR image as a base image (Docker Glossary n.d).

Although the work Spotify’s FOSS team have performed seemed promising for a direct deployment, there were features that did not meet the requirements set by the assignor. For instance, Spotify's version used quite an old version of GRR image as a base image (version 3.2.4.7) so there was need for an update. Another disparity was that Spotify's version takes advantage of GCE, which did not meet the requirement for an offline tool. For this reason, it was decided to use Spotify's version as a review base and build a unique containerized GRR on top of the developing work that Spotify’s developer team has made. (Spotify 2019.)

5.3 Division of official GRR Docker image

The division work of official GRR Docker image was started by making the code base review to the GRR Rapid Response’s and Spotify’s Git project repositories. During the review it was seen to be suitable that the initial build environment of divided GRR Docker image should use the same folder structure as Spotify’s FOSS team designed it as illustrated in Figure 3.

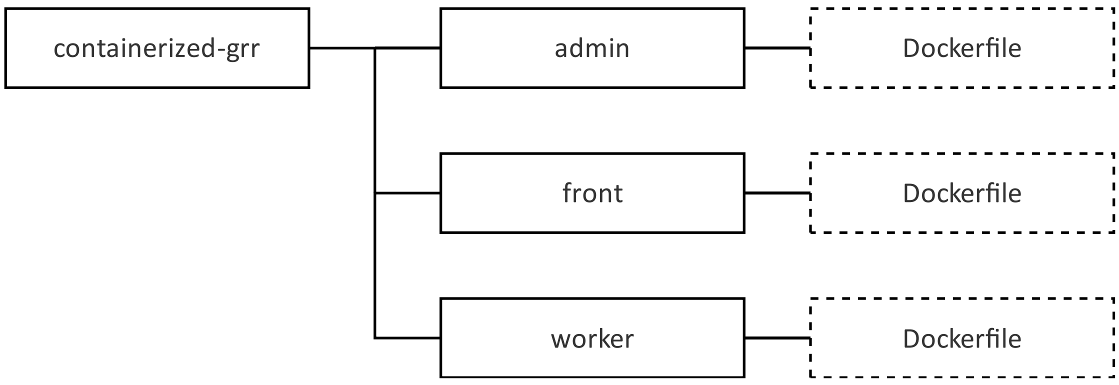

Figure 3. Initial build environment structure for containerized GRR

The use of this kind of folder structure originates from the fact that when the Docker image is built from Dockerfile, Docker uses a special build context for the image. Build context uses by default everything located in the same directory for the build process. Hence, every image to be created from Dockerfile should have only the necessary files and subdirectories for the build process in the same directory to keep the build process efficient. It is worth noticing that excluding additional files and subdirectories from build context is also possible if special .dockerignore file is used; however, on initial build environment the use of .dockerfile was omitted since the directory structure itself reduced and dimensioned the build context correctly. (Dockerfile reference n.d.)

As mentioned, Spotify's Terraform implementation was partly obsolete due to version updates made to GRR. In addition, Spotify's implementation consists of code mainly targeted for Terraform implementation in GCE. For this reason, every file that had source code or configurations related to Terraform or GCE was examined, left out or modified from execution of the containerized GRR build environment. There was also a need to write totally new BASH (Bourne-Again Shell) shell scripts to make the preparation for deployment process of GRR server more convenient. Hence, the folder structure was the only concern seen to be suitably used as it is in Spotify's implementation. Figure 4 illustrates the final working and the tested build environment structure for the containerized GRR version 3.3.0.8, the latest version of GRR (released on 9 Oct 2019).

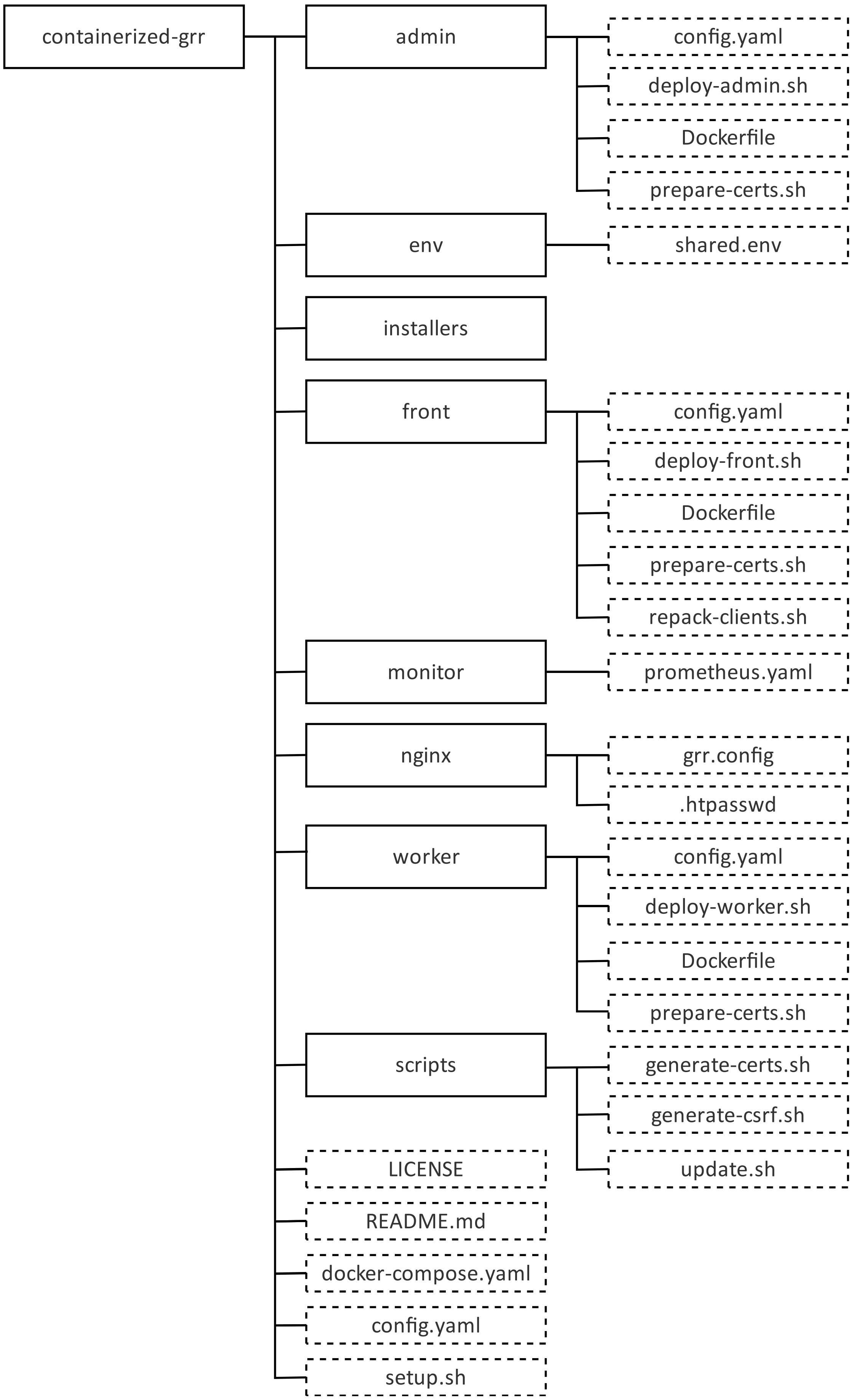

Figure 4. Build environment structure for containerized GRR Rapid Response

The division work of official GRR image happened to be a more challenging process than predicted. A lot of time for division work was needed to be used only for GRR's and Spotify's code base reviews, and it was required to read official documentation to understand the fundamental functionalities of GRR. For example, GRR uses in its working special configuration parameter expansion, filtering the syntax of which is totally unique.

Another challenge was solving the build process for each logical component. As Docker, GRR uses a special build context when components are deployed; however, in the official image the build of components are bundled into one build process. In the end, the use of GRR configuration filters, single component build context, and other issues were solved. After testing, the division work of official GRR Docker image succeeded without errors.

The Spotify's code base is licenced under Apache License, version 2.0 meaning that the licence used for the division work was needed to be the set to the same or stricter license so that the violation of terms of licence does not occur. The license for division work was set to be Apache Licence, version 2.0. (Apache License 2019.)

5.4 Deployment of containerized GRR server

For the deployment of containerized GRR, a special Docker Compose file was composed. Every Dockerfile can be used to build a respective image, which can then be used to launch a running instance of the image, respective Docker container. However, when multi-container applications with different services are created, and the number of images to be built increases, the more convenient way is to use Docker Compose file. The file is composed using YAML (YAML Ain't Markup Language) data serialization standard language, which is a manageable and user-friendly way to describe and manage the multi-container application and its configuration parameters. Every service of multi-container application is described to the Docker Compose file separately; however, by default the run of which is performed simultaneously. (Ben-Kiki, Evans & döt Net 2009; Compose file version 3 reference n.d; Overview of Docker Compose n.d.)

Docker Compose and the constructed Docker Compose file enable deployment of the containerized GRR in two commands. The first command is used to create a suitable subnetwork for the containers allowing data transmission between correct components with limited access to the system and external network. The second command is used for building, deploying, and launching the run of multi-container GRR server from composed Docker Compose file. This was seen to meet the requirement set by the assignor for the simple and straightforward deployment of GRR server.

Figure 5 presents the functionality of containerized GRR server components. As mentioned, when deployed all communication between components is transmitted on its own isolated subnetwork. However, the HTTP front-end and user interface have limited access to the system and external network through the Nginx reverse proxy. Reverse proxy ensures that the communication from GRR agents and investigation workstation to the GRR server is transmitted successfully to the corresponsive components and vice versa. Container name conventions presented in Figure 5 should be comprehended as follows:

- grr-proxy equals to Nginx reverse proxy
- grr-admin equals to user interface component
- grr-front equals to HTTP front-end component
- grr-worker equals to worker component
- grr-mysql equals to data store component.

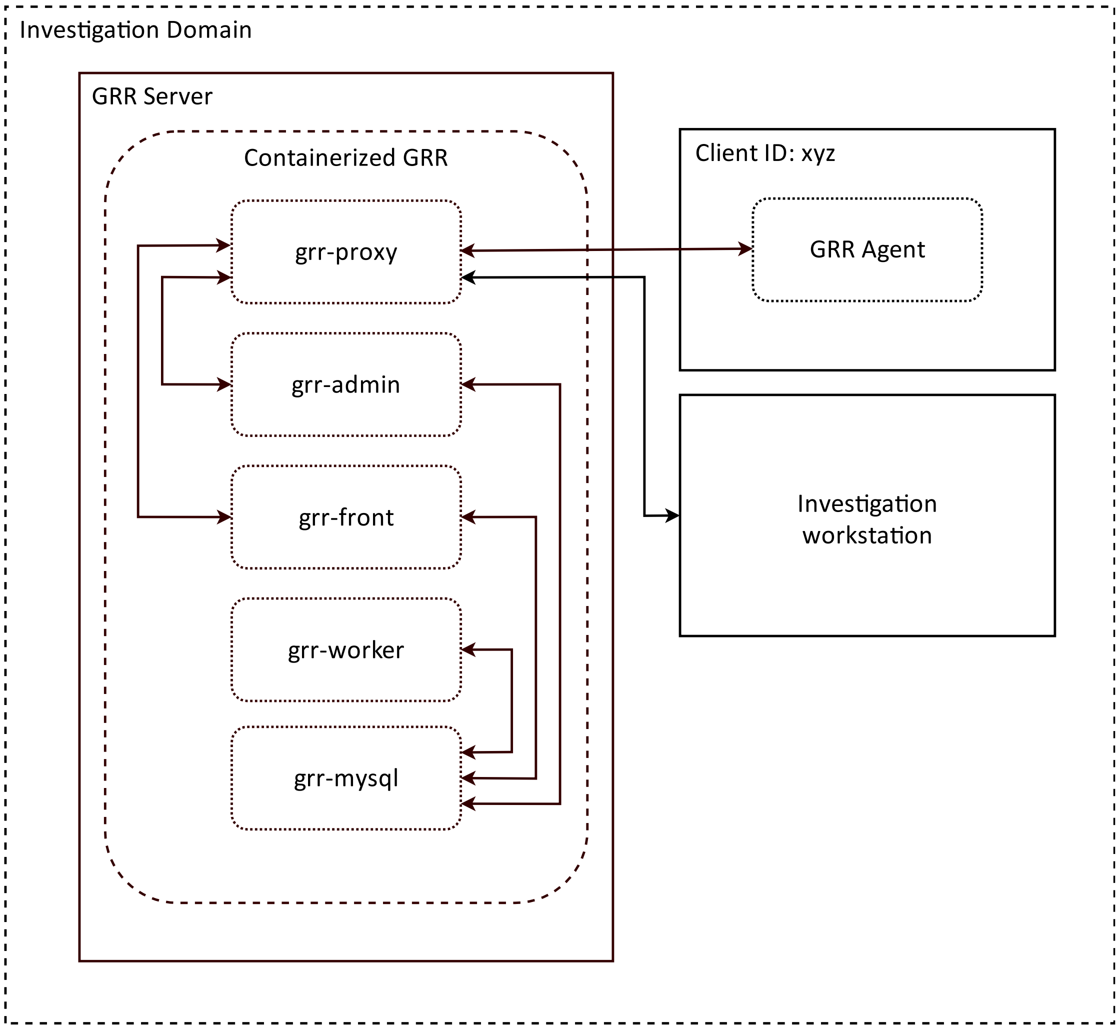

Figure 5. Functionality of containerized GRR

Reverse proxy also provides added security for the data transmission between investigation workstation and GRR server offering TLS (Transport Layer Security) encryption for the transmitted data during TCP (Transmission Control Protocol) connection ergo HTTPS (Hypertext Transport Protocol Secure) when administrative GUI or API endpoint is reached by the investigator. Furthermore, Nginx reverse proxy can be also used for load balancing, which enables division of the same type traffic between multiple similar containers reducing the load directed to the single component. This is a feature needed when component scaling is performed. (Nginx n.d.)

5.5 Deployment of GRR and osquery agents

When the containerized GRR server is deployed, the server populates a local directory from build environment with a respective GRR agent installers for every supported OS. The GRR agents along with osquery agents can then be installed and deployed on desired endpoints. However, the GRR server does not provide installers for the osquery agents directly, and they must be fetched separately from official source.

A convenient way to carry out agent installation on endpoints in organizational environments is to use centralized management such the Group Policies offers in Active Directory environments. Group Policies allow pushing organization-wide agent installation, update, and uninstallation on desired endpoints. However, after the containerized GRR has been deployed, the server can also be used directly to push updates for the GRR agent and if desired, perform agent termination or uninstallation. By default, the containerized GRR can manage osquery agents only when an investigation is made; however, the server allows execution of binaries on clients, the feature of which can be used to deliver e.g. updates on osquery agents.

When the agent installation process is terminated successfully, the GRR agent intends to connect to the containerized GRR and if it succeeds, the respective client is added to the list of known clients on the server with status active. The message exchange between server and active status client agents can then be transmitted and investigation work in investigation domain established.

6 Capability testing

6.1 Overview

Capability testing test case was constructed and executed to produce results for the capability related research question: “How well and reliably can the selected tool be used to identify and contain incidents from the selected organizational environment?” In the test case the RGCE’s organizational environment Funnel was used as a testing environment, which was used to simulate a real-world organization environment on which the deployment and implementation of the researched tools were performed and results for the research question gathered.

The real-world APT level cyber-attack was generated on testing environment which involved the researched tools to investigate attacked environment endpoints on which the realistic adversary techniques were targeted at. The tools were used to gather attack related data, artefacts, from the management workstations. The acquired artefacts were determined based on IoCs that the generated test case attack left on the target systems.

6.2 Testing environment

Testing environment selection was made based on requirements defined together with the assignor in order to create a real-world circumstance for the test case execution. First, the selected environments had to simulate the real-world organization environment as accurately as possible from including public and internal services and endpoints to correct networking solutions. Secondly, the deployment of the researched tools on the environment should be a straightforward process.

RGCE's road tunnel provider Funnel's environment was selected to be used as testing environment. Funnel is a fictional organization, the main objective of which is to provide an undersea road tunnel between Helsinki and Tallinn. Funnel's network is divided into two separate segments: office and management, of which the office segment was selected to be used in the test case. The organization's environment was designed bearing in mind JYVSECTEC's commercial Digital Forensics and Incident Response (DFIR) trainings, and therefore it adapted perfectly to be used also in capability testing. The precise description of Funnel network topology is not relevant for the execution of research and for this reason it is omitted. (JYVSECTEC Cyber Range – RGCE and solutions n.d.)

Figure 6 illustrates the structure of the testing environment in the resolution needed to describe of how the tool deployment was made to the environment. Furthermore, Figure 6 explains how the investigation workflow on the testing environments was resolved: Starting from on how the investigation workstation was managed by investigator to request GRR server to start investigation on client(s) where GRR and osquery agent(s) were installed, which then sent results to GRR server informing the investigation workstation to show generated results to the investigator.

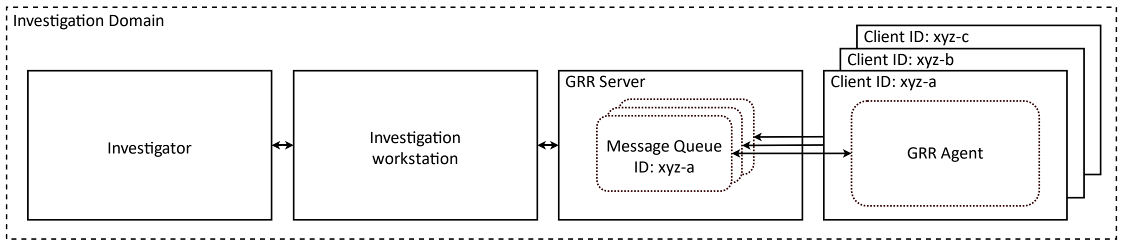

Figure 6. Investigation workflow on testing environment

It is worth noticing that the containerized GRR server could have been made inside or deployed outside of the testing organization environment’s network; however, for the execution of the test case the GRR server was placed inside of the Funnel organization’s management network segment, where it was managed remotely by the investigation workstation. GRR and osquery agents were installed and deployed using centralized Group Policy on ten management workstations running Windows 7 OS.

6.3 Test case attack

6.3.1 Purpose

The test case attack for the capability testing was created on Funnel’s organization environment to involve the researched tools to investigate and to gather artefacts related to the real-world incident from the environment’s management workstation to enable evaluation of the capability of the researched tools. The attack comprised APT level techniques to create a real-world security incident on Funnel’s organization environment’s staff network.

Used techniques in test case attack produced IoCs to the target environment's management workstations. This knowledge was used for the tools to focus on the investigation and collect only the corresponsive artefacts from the systems where the GRR and osquery agents were installed.

6.3.2 Phases

An email message with malicious OpenDocument Text (ODT) attachment was sent to four employees of Funnel roadway tunnel organization. The sender's email address was spoofed to see that it was sent from a person with whom the employees have a trusted relationship. The document included a macro created by Visual Basic for Application (VBA) containing an obfuscated PowerShell script. The obfuscated script was executed on the system when the document was opened, and the run of macros was accepted. (Dent & Blawat 2017; Snover 2016.)

The email attachment was opened and executed in total of four of the ten Funnel management workstations by employees. The obfuscated PowerShell script in malicious macro planted and executed a Cobalt Strike Beacon from the target file system which started immediately to communicate with the Command and Control (C2) server. Beacon communicated with the C2 server in constant intervals using HTTP requests and DNS queries and executed the commands sent from the C2 server on the target environment. (Cobalt Strike documentation n.d.)

Using the Cobalt Strike Beacon an internal recon was executed in the target environment with an intention to find those servers and services from the environment that can be used in lateral movement or contained potentially useful information. The internal recon consisted of actions such as port scanning of various subnets on target environment network and unsuccessful logins attempts to Domain Controller (DC) using a fileless PowerShell script. Furthermore, the organizations network share was explored, which revealed a valuable file. The file contained information about the domain administrator credentials, which were then used to successfully login remotely to the organization's DC using Windows Management Instrumentation (WMI).

The Figure 7 illustrates the different phases of the described test case attack from initial access to the lateral movement.

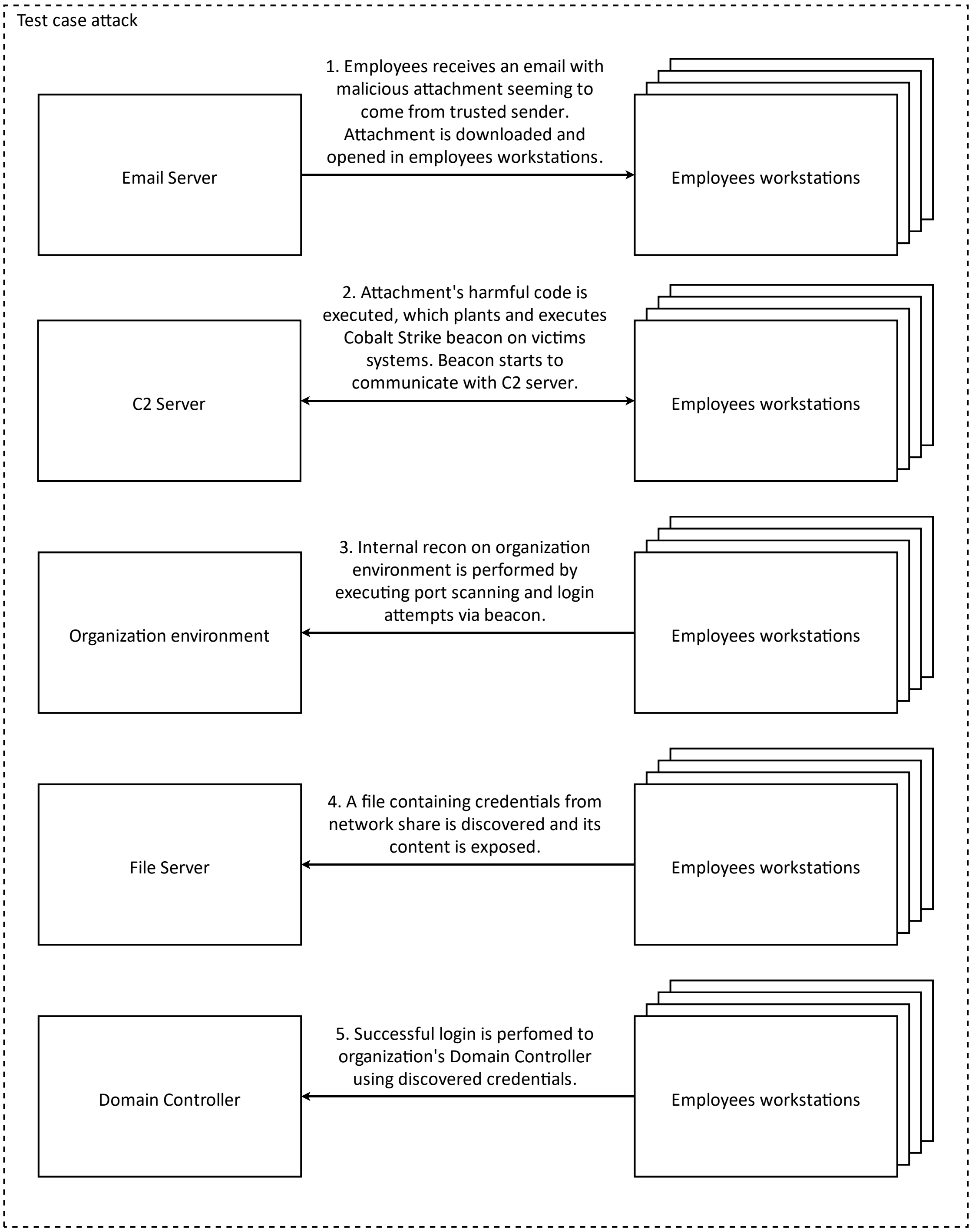
Test case attack
1. Employees receives an email with malicious attachment seeming to come from trusted sender. Attachment is downloaded and opened in employees workstations.
Email Server
Employees workstations
2. Attachment's harmful code is executed, which plants and executes Cobalt Strike beacon on victims systems. Beacon starts to communicate with C2 server.
C2 Server
Employees workstations
3. Internal recon on organization environment is performed by executing port scanning and login attempts via beacon.
Organization environment
Employees workstations
4. A file containing credentials from network share is discovered and its content is exposed.
File Server
Employees workstations
5. Successful login is perfomed to organization's Domain Controller using discovered credentials.
Domain Controller
Employees workstations

Figure 7. Anatomy of test case attack

6.3.3 Mapping the techniques

Mitre ATT&CK Framework is a community-driven knowledge base. The framework’s enterprise version consists of a detailed information about the TTPs that the APT-level actors have observed to be using during their campaigns targeted at organization environments. According to Mitre (n.d), the framework's construction work was started in 2013 by Mitre and on 9 October 2019 the framework consisted of 12 different enterprise tactics, 266 techniques and 41 mitigation methods. (Mitre ATT&CK n.d.)

The usefulness of the framework has been absorbed widely by the different actors in the domain of cyber security such as software vendors, government agencies as well as private sector and service community. A large number of different actors use the framework when referring to APT actors and their capabilities in their products to share information to the end-users. For instance, Microsoft refers to Mitre's knowledge base in Windows Defender Advanced Threat Protection (WDATP) service. Furthermore, different tools have added features and modules to enable referencing in Mitre's knowledge base. One of the well-known Microsoft Sysinternal tools, system monitoring tool Sysmon, has optional configuration developed by Olaf Hartong enabling the mapping of malicious events to the corresponsive technique in Mitre's knowledge base. (Mitre ATT&CK n.d; Sysmon modular n.d.)

Mitre ATT&CK Framework's technique navigator can be used to illustrate (Figure 8) how the used TTPs in different phases of the executed test case attack projects to the Mitre ATT&CK Framework matrix.

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement
Spearphishing Attachment	Command-Line Interface	New Service	New Service	Scripting	Brute Force	File and Directory Discovery	Windows Remote Management
Trusted Relationship	PowerShell	Valid Accounts	Valid Accounts	Valid Accounts	Credentials in Files	Network Service Scanning	
Valid Accounts	Scripting					Network Share Discovery	
	Service Execution						
	User Execution						
	Windows Management Instrumentation						
	Windows Remote Management						

Figure 8. Mitre ATT&CK Framework – Technique navigator (filtered)

The illustrated matrix shows that the used TTPs in the test case attack are the same as the APT actors have been used in their campaigns. It is worth noticing that the used techniques cover only the phases from the initial access to the lateral movement on a matrix. However, in this research the scope of the attack was seen to be reliable to gather accurate results, and the execution of other attack phases was omitted.

In this research the mapping of the test case attack techniques to the Mitre's matrix was used only for the verification purposes; however, if mitigation methods, requirements or additional information about the used techniques and adversaries who are known to be using these techniques should be needed, the mapped fields of the matrix should have led directly to the corresponding source of Mitre's knowledge base.

7 Performance requirement testing

7.1 Overview

Performance requirement testing was constructed and executed to produce data that can be used to answer the research question: “What are the performance requirements when the selected tool is used for investigation performed in organizational environment that consists of hundreds of endpoints?”.

To generate accurate results the researched tools were used normally, meaning that the tools were tested as they should be used in everyday investigation in real-world circumstances. It is advisable to notice that the test was not created to benchmark the tools but only for dictating performance requirements needed when the performance recommendations for the cybercrime prevention environment is made by assignor.

Performance requirement testing focused on the server-side of GRR, and testing for the GRR and osquery agents was excluded from the research. As the GRR’s developers indicate on GRR’s official documentation (n.d), the certain server-side components are under high load when an investigation is performed. In addition, it was noticed that there was already existing research material available about the GRR agent memory usage considerations and footprint on client (Moser & Cohen 2013). What is more, the deployment and development work performed for the GRR’s server-side, containerized GRR, raised the need to focus performance testing on the server-side of GRR. For beforementioned reasons the scope for the performance requirements testing was set with the assignor to consider only the server-side of GRR.

7.2 Testing environment

As in capability testing, the testing environment selection for performance requirement testing was made based on the requirements defined together with the assignor. In this way it was possible to provide accurate and reliable data for the performance requirement evaluation. First, the organization used for performance requirement testing has had to consist of advanced monitoring software so that the accurate monitoring could be executed and reliable data for the evaluation purposes generated. Second, the organization environment had to include hundreds of endpoints on where the investigation could be performed to create a valuable amount of monitoring data.

The selected environment was totally unique since it was designed to be utilized only in one national cyber security exercise by only specific participating training organization. Any other distinctive information about the training organization or the used testing environment could not be revealed since the delicense reasons. However, the used testing environment fills the predefined requirements. In addition, the investigation workflow in testing environment corresponded to the workflow presented in capability testing.

7.3 Load monitoring of containerized GRR

A suitable event for the performance testing was chosen to be the KYHA2019tv cyber security exercise for Finland's national defensive agencies. The five-day exercise was held in fall 2019 and was planned, organized and executed by JYVSECTEC. In exercise, the tools were used for everyday investigation to identify APT level security incidents from organizational environment. During the exercise a large collection of different adversary techniques was used, which produced an extensive amount of IoCs on the organization environment's endpoints that were investigated from the artefacts gathered using the tools. (Security authorities develop their competence in National Cyber Security Exercise (KYHA19) 2019.)

The GRR server was monitored using PRTG Network Monitoring software. PRTG enables a real-time monitoring as well as collecting and exporting data from sensors in between the specific time frame. PRTG sensors can be used to monitor different loads from the system such CPU (Central Processing Unit) and RAM (Random Access Memory) when protocols such as WMI and SNMP (Simple Network Management Protocol) are used. In addition, PRTG enables gathering data from network bandwidth utilizing industrial softwares and standards such as NetFlow and sFlow. (Monitor Load n.d.)

Port binding and requested certificate files were configured on the host’s Docker daemon unit file, which enabled to add a sensor for each container of containerized GRR server component in PRTG monitoring software. In this way it was possible to monitor and gather load data from distinct components in a centralized way. (PRTG Manual – Docker Container Status Sensor n.d.)

Load monitoring data was used to indicate those components of the GRR server that were under high load during the test case and can be considered as possible bottlenecks if sufficient resource targeting is not executed and investigation is made intensively in large scale. After the test case the accurate data from sensors were exported as CSV data. The results from the exported data were calculated and used to make performance recommendations for the containerized GRR.

Table 1 presents the details of the hardware components used in the test case for the containerized GRR. The hardware consisted of a single dual-core CPU with reasonable amount of memory (RAM), 8 GB. In addition, the hardware included a hard disk with total of 164 GB free space. The assigned resources were seen to be enough so that there is no exhaustion of resources during test case. The host system had CentOS Linux 7 OS with Docker Engine and Docker Compose installed.

Table 1. Used hardware

Component	**Description**
CPU	1 dual-core
Hard disk	164 GB
Memory	8 GB

8 Results

8.1 Containerized GRR

The deployment work of GRR server produced a unique containerized GRR, which was developed to meet the requirements set by the assignor for the tool's deployment on RGCE's organizational environment. Containerized GRR brings added value when deployment of GRR Rapid Response is considered; however, when following features are also required from the tool:

- Simple and straightforward deployment process using Docker images
- Scaling for a single component is possible to perform
- Minimal need for required dependencies
- Added security.

Containerized GRR should be considered as a current work as the member of GRR's developer team, Ogaro (Ogaro 2019b) informed that the containerization of GRR is also one of the team's main priorities in the medium or long run. The development work made for containerized GRR is directly available for the use of GRR developer team and other developers, since the work will be published by the assignor in near future. Appendix 1 presents the deployment manual for the containerized GRR, which can be used as a reference material when the tool is deployed.

It is worth noticing that the testing made for containerized GRR was limited due to time restrictions and for this reason the more intensive testing and the evaluation also by other contributors needs to be performed to test the tool is trustworthy. Furthermore, the scalability of the containerized GRR was not tested during the research, since it was not demanded by the test cases or testing environment circumstances; however, testing the single component scalability opens a need for further research.

In the end, containerized GRR functioned and performed from the test cases without any errors, which shows that the performed development work has been successful and the objective set for the deployment work of the researched tools was reached.

8.2 Capabilities

The produced results involved preselected remote live response tools, GRR and osquery, to gather data about the generated APT level incident targeted in a real-world organization environment.

Table 2 presents the general information about the investigation performed in Funnel's organization environment during the test case execution. Active clients present the number of those management testing environment's management workstations on which GRR and osquery agents were deployed. Respectively, contaminated clients indicate the number of the endpoints found the be contaminated by the test case attack. Used flows and hunts present those features of GRR that were used to gather artefacts from endpoints using GRR and osquery agents.

Table 2. Investigation details

Information	**Value**
Active clients	10
Contaminated clients	4
Used flows and hunts	MultiGetFile, ListDirectory, ArtifactCollectorFlow, OsqueryFlow, Client Side File Finder, Netstat, ListProcesses

Table 3 consists of descriptive information about IoCs that the test case attack produced to the Funnel organization environment's management workstations and corresponsive artefacts acquired during the test case using the researched tools. The workstations with artefacts consisting of test case attack related IoCs were considered as contaminated and the containment for the workstations from the testing environment was performed.

Table 3. Gathered artefacts

IoC description	**Gathered artefacts**

Malicious email attachment	File system, Prefetch files, Running processes
Obfuscated PowerShell script	Windows Event log
Cobalt Strike Beacon	File system, Running processes, Outbound network traffic, Prefetch files
Fileless PowerShell script	Windows Event Log, Localhost network traffic

Table 4 presents the capability of the researched tools in a summary. The purpose is to adduce features from both tools that were able to be used to gather artefacts consisting those IoCs that tie the client in question to the test case attack. In addition, the results show those features of the tools that overlapped with each other, and possible caveats when data was collected.

Table 4. Artefact acquiring capabilities

Artefact	**GRR Rapid Response**	**Osquery**
File system	Audit / Collect	Audit
Localhost network traffic	Audit / Collect	Audit
Outbound network traffic	Audit / Collect	Audit
Prefetch files	(Audit) / Collect	(Audit)
Running processes	Audit / Collect	Audit
Windows Event log	(Audit) / Collect	Audit

Audit points those artefacts that were able to gather without extracting them separately from the target system and collect in turn those artefacts that could be extracted and pulled from the target system using the researched tools. Audit with parentheses indicates that the tool was able to audit the desired artefact; however, the used mean does not suffice when the precise investigation is executed.

The capability of the tool depended on how well and reliably the tool was able to gather incident related artefacts from the clients. Presented results do not take into account which tool is preferable or has a better feature to perform a specific task on the client during investigation, since this was not in the scope of research. In addition, the results evaluate the capability of the tools when they are used for investigation in similar circumstances.

As the results indicate, the tools overlap with each other when speaking of data auditing; however, only GRR Rapid Response was able to extract and collect all the desired artefacts from contaminated workstations. Osquery's ability to only audit data originates from the fact that the osquery is designed to describe the system and it changes. For this reason, lack of data extraction feature in osquery should not be considered as a shortage.

It should also be noted that the GRR is able to audit Windows Event log and Prefetch files in file system level; however, the tool's text and hex views do not describe the contents of the binary formatted files in human-readable format and for this reason the lack of audit possibility was considered as a caveat. However, the possibility to audit files' contents in beforementioned ways is a desired feature when other types of binary and text files are examined. Furthermore, osquery was able to audit the contents of Windows Event log files; however, auditing the Prefetch files consisted of only the file system level auditing as in GRR.

The GRR consisted of a large collection of flows for different artefact auditing and collection to be used via user interfaces, such as listing and acquiring binaries for running processes; however, some artefacts such as specific Windows Event logs were needed to be collected separately using flows that allow direct file system exploration and data extraction. It is worth noticing that some of the flows also had similarities with each other and explanations about the differences were frail. One explanation for multiple similar flows was found to be related to the fact that some of the flows are just a more robust and efficient implementation of its predecessor; however, use of the correct flows is left to the investigator, and lack of precise references exists.

The osquery was used by GRR mainly via specific collector flow assigned for the use of osquery. The flow allowed use of osquery in a similar way as it functions without GRR, i.e. performing database queries. Constructing and performing queries for the use of test case artefact auditing was straightforward and efficient. After the corresponsive query was constructed and executed, the results were directly available to be examined from the GRR's user interfaces. In spite of overlap between similar features, the GRR's support for osquery was considered to be a desired feature, since it enabled efficient and human-friendly data auditing after the syntax of the queries was absorbed.

As mentioned earlier, without the GRR's ability to operate as osquery's fleet manager, the use of osquery agents remotely by default should not have been possible and implemention of centralized manager for osquery should have to be solved and implemented separately. In this research the researched tools were used to extend each other, not excluding others' necessity.

In the end, the tools have caveats and features that overlap with each other; however, when these concerns are taken in account the both of the researched tools, GRR Rapid Response and osquery were able to perform remote live response well and reliably and have a strong capability when security incident related artefact data is audited and collected from endpoints on organizational environment during security incident investigation.

8.3 Performance requirements

The produced results involved using the preselected remote live response tools, GRR Rapid Response and osquery for security incident investigation in cyber security exercise event in five-day duration so that it was possible to gather a reasonable amount of data for evaluation from PRTG Network Monitoring software sensors.

Table 5 presents the general information about the test case execution. The number of installed clients presents those endpoints on testing environment on which GRR and osquery agents were installed and deployed. Furthermore, the number of PRTG sensors equals the number of monitored GRR server component containers.

Table 5. General information

Information	**Value**

Clients	173
Monitoring software	PRTG Network Monitor software
PRTG sensors	4
Test case duration (days)	5

Investigation related data gathered from GRR server is composed in Table 6. The number of terminated flows is directional, and precise value lands somewhere between 5,800 and 6,000 terminated flows. In addition, the number of active clients indicates the highest coincidental number of clients that were in active state during the test case. Due to the nature of the test case event, the number of active clients varied between days; however, remaining over 160 clients on each day of test case.

Table 6. Investigation related information

Information	**Value**
Active clients	173
Active cron jobs	6
Executed hunts	22
Terminated flows	> 5800

Figures 9–12 present the monitored load for each containerized GRR container acquired from PRTG Network Monitoring software sensors during test case. The amount of investigation presented in Table 6 was enough to produce a reliable load on containers. As the results indicate the load could have been notably larger and still the exhaustion of resources would have taken place; however, by the nature of the test case event, the load of investigation was not controllable during the test case. The Figures 9–12 presents the following calculated average loads from five-day duration to each container:

- CPU usage
- Memory (RAM) usage
- Packet speed (Sum of input and output speed)
- Total packet count (Sum of input and output count).

In Figures 9–12 the average load by container is presented in thick black lines in blue boxes. Blue boxes indicate the range in which most of the measured values landed, and the T values the highest and lowest measured values during test case execution. Container name conventions presented in Figures 9–12 should be comprehended as follows:

- admin equals to user interface component
- front equals to HTTP front-end component
- worker equals to worker component
- mysql equals to data store component.

Figure 9 presents the memory usage of each containerized GRR container. The results show that the HTTP front-end has on average the highest memory usage, around 9 %, as well as that the worker container had the least memory usage, only 2–3 %. The biggest variance on memory usage in five-day time frame was on MySQL database container (4–10 %).

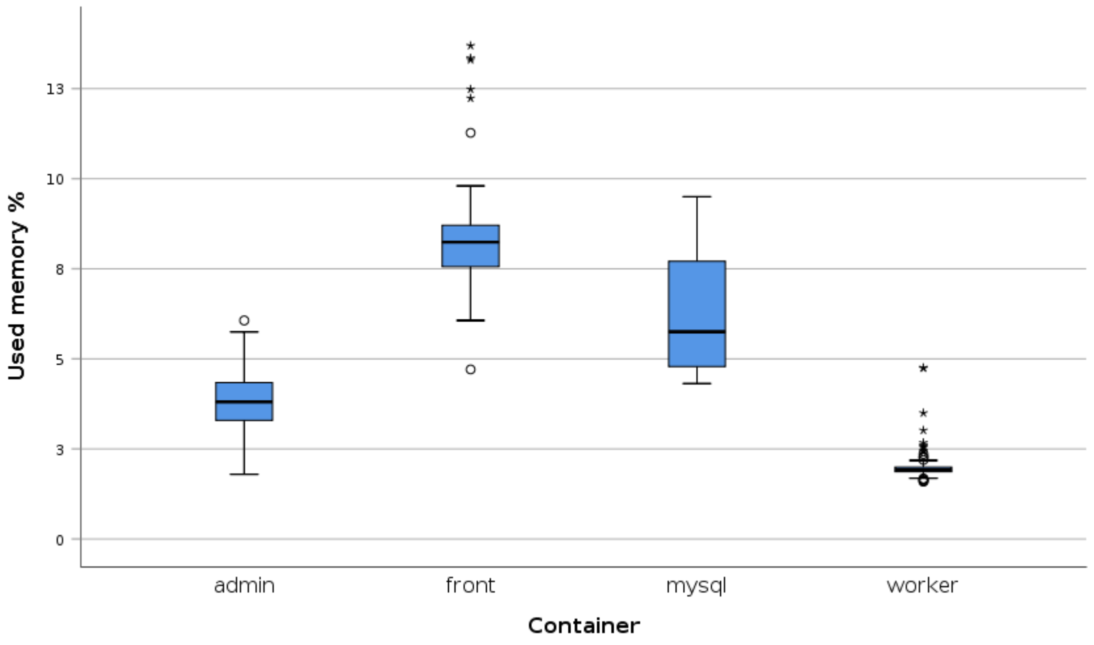

Figure 9. Memory usage by container

Figure 10 presents the CPU usage of each containerized GRR container. As in memory usage, the HTTP front-end container’s CPU usage average was the largest, being around 20 %. The second highest CPU usage average was measured by MySQL database container sensor; however, the difference between HTTP front-end and MySQL database is significant, around 15 %. The produced results indicate that the user interface and worker container's CPU usage is quite moderate and does not charge resources from the underlying hardware.

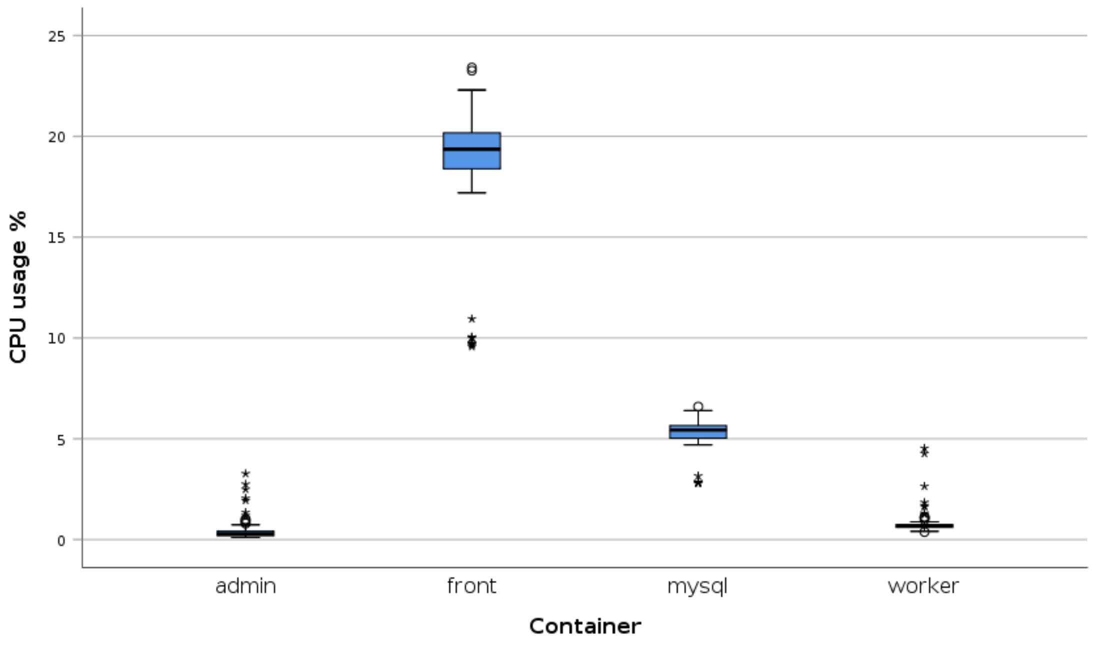

Figure 10. CPU usage by container

Figure 11 presents the overall of inbound and outbound packet speed by containerized GRR container. The results indicate that the HTTP front-end and MySQL database containers had the highest packet speed average during the test case as well as that the user interface and worker container's overall packet speed is quite moderate. Furthermore, the results indicate that the HTTP front-end and MySQL database containers manage the major part of the data transmission.

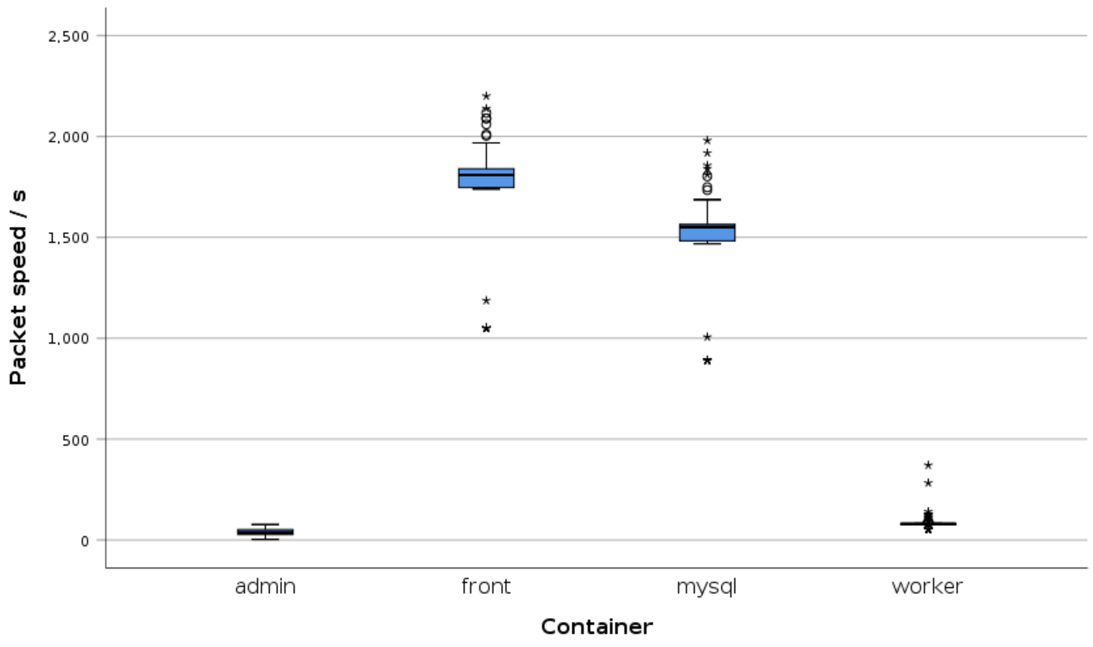

Figure 11. Packet speed by container

Figure 12 presents the packet count by containerized GRR container. As in packet speed, the packet count results indicate that the containerized GRR's data transmission relies largely on two components, HTTP front-end and MySQL database.

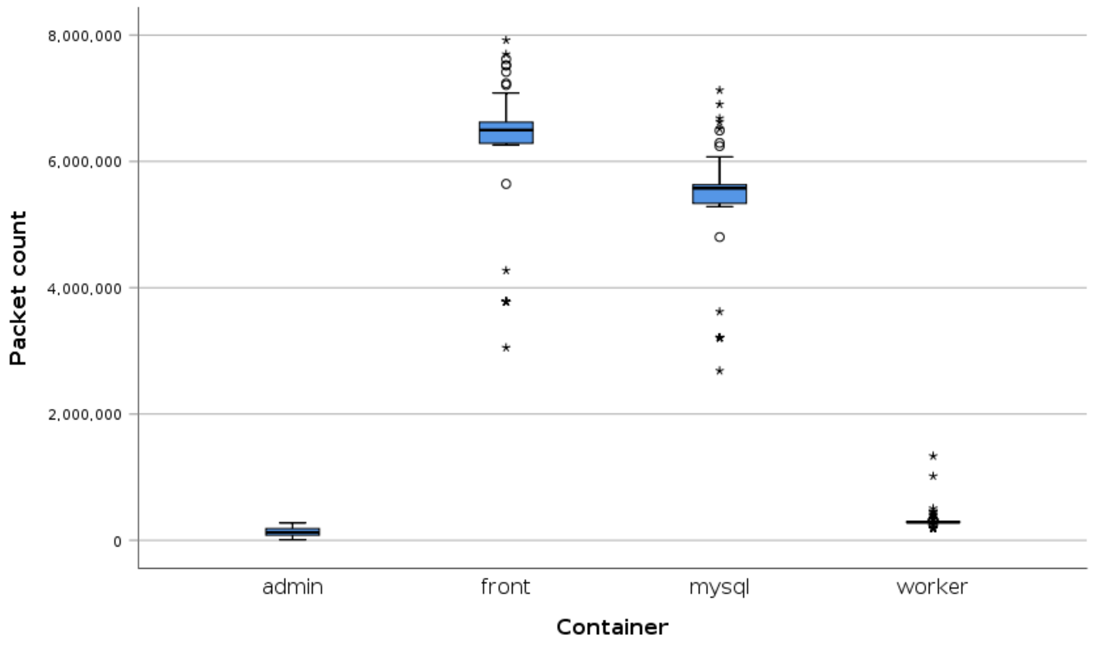

Figure 12. Packet count by container

Table 7 presents the containerized GRR's image disk space usage, the minimum space which is needed when the initial deployment of containerized GRR is made.

Table 7. Images space usage

Docker image	**Size (GB)**
grr-admin	1.8
grr-front	1.8
grr-mysql	0.5
grr-proxy	0.2
grr-worker	1.8
Total	6.1

Table 8 presents the hard disk usage by each container and the total hard disk usage of containerized GRR in five-day duration. It is advisable to notice that the values presented in Table 8 do not take into account the size of collected artefacts, which reduced the investigation workstation's hard disk space, not the host's where the containerized GRR was deployed.

Table 8. Containers space usage

Docker container	**Size (GB)**
grr-admin	0.9
grr-front	34.6
grr-mysql (mounted)	0.6
grr-proxy	0.3
grr-worker	2.9
Total	39.3

The results show that the used hardware during test case was suitable enough; however, if investigation duration had been longer, the available hard disk space would have been reduced and shredded. The reason for excessive disk usage originates from the fact that the verbose logging feature was enabled during the test case execution being the largest single factor for the high disk usage. The log files consumed around 36.7 GB, which is over 90 % of total container's hard disk space usage. Without having verbose logging enabled, the disk usage would have been quite moderate. In addition, at no point of test case does any container consume over 25% CPU or memory, which proved that the selected hardware for the test case was reasonable; however, not being extravagant.

The results also show that the assumptions stated by developers in GRR's official documentation were correct: HTTP front-end component was under higher load than other components, and is a possible bottleneck if resources are not correctly dimensioned. In addition, the user interface component load was moderate.

Table 9 presents the hardware recommendations which are composed by the executed test case. Recommendations cover the hardware requirements when the containerized GRR is used for remote live response on environments that consist of hundreds of endpoints, and the investigation is performed in similar circumstances. The recommendations should cover also the requirements when a single component scaling for the containerized GRR is performed; however, scaling and its impact should be tested and verified separately.

Table 9. Recommended hardware

Component	**Description**
CPU	1 dual-core
Memory	8 GB
Hard disk	32 GB

It should be noticed that the suggested hard disk recommendation do not cover verbose logging. The logging of containerized GRR should be handled using distinct centralized logging system and have verbose logging feature enabled only when the testing for the tool is performed.

9 Conclusions

9.1 Discussion

The main scope of the research focused on two different test cases involving the remote live response tools, GRR Rapid Response and osquery, preselected by the assignor to gather data, artefacts, related to occurred incidents from real-world organization environment's endpoints that indicated both the tools' capability and performance requirements. The research also involved using a significant part of the time for practical deployment work of the tools to the selected environments in RGCE, solving and taking into account the requirements set by the assignor first.

The deployment process of the tools should be considered as a straightforward process and the developers' official documentations consult the user in a sufficient manner if universal tool is deployed. When the deployment work for the tools is started in environments with special restrictions or if special requirements are set for the tools, as in this research, the deployment work should be considered to reverse time, expertise, and additional resources. In such cases, implementing the tool is a development process itself and no easy solutions exist, since every requirement have to be examined, evaluated, and solved separately when the tool's deployment process is started or even considered. In addition, such deployment work needs testing, which also bonds resources and time.

It should also be noted that even if there are pre-existing available solutions made by other developers, they consist inexorably of caveats and should not be considered to fit directly for the deployer's needs without a precise evaluation. However, as the research proved, other developers' pre-made solutions can contain valuable information and a possible starting point for the development work. Furthermore, the implementation of the tool with special needs is easier to perform when the tool and solutions are open sourced, and the specification for the requirements is set and well known.

The capability test case results indicate that even when the remote live response tools have maturity, strong developer background, and are designed to be able to perform investigation in a large scale and an intensive manner, tools have necessarily caveats or overlap with other similar tools designed for the same kind of purpose. For this reason, the “one fits for all” out of the box open source tool for remote live response should be not considered to exist. However, the researched tools fit perfectly to the investigator’s toolkit when the tool’s individual constricts and deficiencies are recognised and evaluated, and when the overlap between tools is noticed and accepted.

If the researched tools are used separately, they do not give as much for the investigator as the GRR’s capability to function as an osquery’s fleet manager is utilized. If GRR is used as an osquery fleet manager, it gives an investigator a combination of two tools that extend the investigator’s ability to investigate the target systems more extensively and efficiently. In addition, since the osquery does not include a centralized way to handle agents in remote locations by default, the GRR should be considered as being a saving when deployment time that is required to be spent when the centralized fleet manager for the osquery is implemented. Implementation of fleet manager is necessary if the remote live response is wanted to be executed by using osquery in large organizational environments. As mentioned, the features that overlap between tools are also easier to perceive when the tools are used as one in investigation workflow. By this way the best features from both tools can be recognized and used.

Performance test case results indicate the performance requirements of the containerized GRR are moderate when investigation in similar circumstances is performed. However, this is not always the case and if the investigation made on environments consisting of tens of thousands of endpoints, the tool will consume resources in a different manner. The recommendations made should, however, cover the most common investigation use cases performed in similar organization environments and circumstances.

The usage of remote live response tools requires expertise from the user. The tools are developed with a special purpose in mind, and the user must understand and master the purpose and proper use of the tools to get the most out of them. However, the usage of the tools is not always unambiguous, and developers often describe the most common use cases in documentation, omitting the specific needs of environments to be deployed and the complex features that are usually left to the user's own concern.

Cooperation with the assignor of the thesis was a success. Information, expertise, administration, and consultation was shared during the research to get the best possible results for the research work. The research work was transparent between parties, which enabled solving the raised issues during the research such RGCE environmental questions.

Selected research methodology, applied research, applied to the research well. By the nature of the research, it focused largely on tools and their deployments as well as test cases which also were largely practical work. In addition, the assignor of the research is satisfied with the produced results, which indicates that the research methodology used was selected well, and that the research was able to cover the research questions set for the research correctly. Furthermore, all the considerations and requirements that the assignor set for the research were achieved.

The accuracy of research was dimensioned correctly; however, correct dimensioning required omitting the Mitre ATT&CK Framework's phases: collection, command and control, exfiltration, and impact from the test case attack, which should have been executed in capability testing test case. In addition, the GRR Rapid Response and osquery agent testing in the performance requirement test case was excluded from the research. However, the accuracy of the research did not jeopardize the results and answers to the research questions; vice versa, dimensioning produced more accurate results.

9.2 Reliability of research

The results of research can be considered reliable. In different phases of research criticism and additional time was used for observation. In reference material gathering source criticism was performed. During the development work of the researched tools and the test case construction the focus was on producing accurate results for research and any unambiguousness was omitted.

The results indicate that the use of containerized GRR can be considered as reliable as the use of official Docker image directly; however, as mentioned, excessive testing needs to be executed to verify the reliability. The direction of a development work; however, has been correct, since the GRR Rapid Response's development team's desire is also to bring the containerization feature to the GRR in future.

9.3 Further research

The research results proved that the two test cases were suitable enough to indicate the tools' capability and performance requirements. In addition, the research verified the possible caveat and overlap that the tools have when they are used simultaneously and as parts of the same investigation workflow. The results are suitable when further research is considered; however, during the research also a need for further research emerged.

The research pointed out that the containerized GRR can be used to the investigation also in larger organizational environments; however, the research executed did not involve testing the containerized GRR's component scalability, since the testing environments used in test cases consisted of only at the best hundreds of endpoints. The scalability in larger deployments should be researched using similar real-world circumstances and suitable test cases as in this research, which evaluates the containerized GRR's ability to perform investigation in environments consist of thousands of endpoints so the scalability of the tool can be verified.

Furthermore, the research addressed a need to examine how the researched remote live response tools, GRR Rapid Response and osquery, change the target system's state and what the tool's impact is when used for investigation and data acquisition. Further research should cover log gathering and analysis, target machine's state analysis as well as a descriptive guide for the investigator on how to perform the investigation when using the tools, and automated use cases with corresponsive tests.

References

5 miljoonaa euroa ammattikorkeakoulujen soveltavaan tutkimukseen ja innovaatioihin. 2018. Page on Finnish Government’s website. Accessed on 31 July 2019. Retrieved from https://valtioneuvosto.fi/artikkeli/-/asset_publisher/1410845/5-miljoonaa-euroa-ammattikorkeakoulujen-soveltavaan-tutkimukseen-ja-innovaatioihin.

About SQLite. N.d. Page on SQLite's website. Accessed on 16 November 2019. Retrieved from https://www.sqlite.org/about.html.

Anttila, A. 2018. Kyberrikokset – Kirjaaminen ja alkutoimet tietotekniikkarikoksissa. Bachelor's thesis. Police University College. Accessed on 14 November 2019. Retrieved from http://urn.fi/URN:NBN:fi:amk-2018060612820.

Apache License. 2019. Apache License, Version 2.0. Accessed on 6 October 2019. Retrieved from https://www.apache.org/licenses/LICENSE-2.0.

Baimyrzaeva, M. 2018. Beginners’ Guide for Applied Research Process – What Is It, and Why and How to Do It?. Occasional report. University of Central Asia. Accessed on 25 January 2020. Retrieved from https://www.ucentralasia.org/Resources/Item/1661/EN.

Ben-Kiki, O., Evans, C., & döt Net, I. 2009. YAML Ain’t Markup Language (YAML Trademark) Version 1.2. 3rd ed. Accessed on 11 November 2019. Retrieved from https://yaml.org/spec.

Bourgeois, D. 2019. Information Systems for Business and Beyond. 2nd ed. Accessed on 5 September 2019. Retrieved from https://opentextbook.site/exports/ISBB-2019.pdf.

Brezinski, D., & Killalea, T. 2002. Request for Comments 3227–Guidelines for Evidence Collection and Archiving. Accessed on 18 October 2019. Retrieved from https://www.ietf.org/rfc/rfc3227.txt.

Chacon, S., & Straub, B. 2014. Pro Git. 2nd ed. Accessed on 17 December 2019. Retrieved from https://git-scm.com/book/en/v2.

Cobalt Strike documentation. 2019. Page on Cobalt Strike’s website. Accessed on 23 November 2019. Retrieved from https://www.cobaltstrike.com/help-beacon.

Cohen M.I., Bilby D., & Caronni, G. 2011. Distributed forensics and incident response in the enterprise. Accessed on 8 August 2019. Retrieved from https://doi.org/10.1016/j.diin.2011.05.012.

Compose file version 3 reference. N.d. Page on Docker's website. Accessed on 8 October 2019. Retrieved from https://docs.docker.com/compose/compose-file.

Cramer, R., & Shoup, V. 2003. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing, 33, 1, 167–60. Accessed on 15 October 2019. Retrieved from http://dx.doi.org/10.1137/S0097539702403773.

Cruz, F., Moser, A., & Cohen, M. 2015. A scalable file based data store for forensic analysis. Accessed on 30 August 2019. Retrieved from https://doi.org/10.1016/j.diin.2015.01.016.

CYBERDI – Kansallista & Kansainvälistä kyberosaamista kasvattamassa. N.d. Page on JAMK University of Applied Sciences Jyväskylä's website. Accessed on 24 September 2019. Retrieved from https://www.jamk.fi/fi/Tutkimus-ja-kehitys/projektit/CYBERDI/Projektiesittely.

CYBERDI. N.d. Page on JYVSECTEC's website. Accessed on 24 September 2019. Retrieved from https://jyvsectec.fi/2018/10/cyberdi.

Dent, C., & Blawat B.J.W. 2017. Mastering Windows PowerShell Scripting–One-stop guide to automating administrative tasks. 2nd ed. Packt Publishing.

Developer Guide. N.d. Page on Google's website. Accessed on 7 November 2019. Retrieved from https://developers.google.com/protocol-buffers/docs/overview.

Docker Glossary. N.d. Page on Docker's website. Accessed on 7 December 2019. Retrieved from https://docs.docker.com/glossary.

Dockerfile reference. N.d. Page on Docker's website. Accessed on 7 December 2019. Retrieved from https://docs.docker.com/engine/reference/builder.

Finnish cyber security expertise in Singapore – JAMK University of Applied Sciences and Singapore Polytechnic collaborating in Centre of Excellence in Applied Cyber Security. 2019. Page on JYVSECTEC's website. Accessed on 31 August 2019. Retrieved from https://jyvsectec.fi/2019/01/finnish-cyber-security-expertise-in-singapore-jamk-university-of-applied-sciences-and-singapore-polytechnic-collaborating-in-centre-of-excellence-in-applied-cyber-security.

Graves, M. W. 2013. Digital archaeology – The Art and Science of Digital Forensics. Pearson Education.

GRR Rapid Response documentation. N.d. Page on GRR Rapid Response's website. Accessed on 31 August 2019. Retrieved from https://grr-doc.readthedocs.io/en/v3.3.0.

GRR Rapid Response. N.d. Page on Github's website. Accessed on 22 October 2019. Retrieved from https://github.com/google/grr/tree/v3.3.0.8.

Hirsjärvi, S., Remes, P., & Sajavaara, P. 2008. Tutki ja kirjoita. 14th ed. Helsinki: Tammi.

Introduction to Terraform. N.d. Page on HashiCorp's website. Accessed on 20 December 2019. Retrieved from https://www.terraform.io/intro/index.html.

Johansen, G. 2017. Digital Forensics and Incident Response – An intelligent way to respond to attacks. Packt Publishing.

JYVSECTEC Cyber Range – RGCE and solutions. N.d. PDF document on JYVSECTEC's website. Accessed on 29 August 2019. Retrieved from https://jyvsectec.fi/wp-content/uploads/2018/10/JYVSECTEC-cyber-range.pdf.

JYVSECTEC organization. 2019. Page on Github’s website. Accessed on 28 December 2019. Retrieved from https://github.com/jyvsectec.

JYVSECTEC overview. N.d. Page on JYVSECTEC’s website. Accessed on 4 November 2019. Retrieved from https://jyvsectec.fi/about/overview.

JYVSECTEC services. N.d. Page on JYVSECTEC’s website. Accessed on 4 November 2019. Retrieved from https://jyvsectec.fi/services.

Luttgens, J. T., Pepe, M., & Mandia, K. 2014. Incident Response & Computer Forensics, 3rd ed. McGraw-Hill Education.

Marcella, J., & Menendez, D. 2008. Cyber Forensics – A Field Manual for Collecting, Examining, and Preserving Evidence of Computer Crimes. 2nd ed. Auerbach Publications.

Merriam-Webster dictionary. N.d. Page on Merriam-Webster's website. Accessed on 28 December 2019. Retrieved from https://www.merriam-webster.com/dictionary/forensics.

Mitre ATT&CK. N.d. Page on Mitre ATT&CK’s website. Accessed on 2 January 2020. Retrieved from https://attack.mitre.org.

Monitor Load. N.d. Page on PRTG Network Monitoring Software's website. Accessed on 1 November 2019. Retrieved from https://www.paessler.com/monitor_load.

Moser, A., & Cohen, M. I. 2013. Hunting in the enterprise – Forensic triage and incident response. Accessed on 31 August 2019. Retrieved from https://doi.org/10.1016/j.diin.2013.03.003.

National Institute of Justice. 2008. Electronic Crime Scene Investigation – A Guide for First Responders. 2nd ed. Accessed on 13 January 2020. Retrieved from https://www.ncjrs.gov/pdffiles1/nij/219941.pdf.

Nginx. N.d. Page on Nginx's website. Accessed on 19 January 2020. Retrieved from https://nginx.org/en.

NIST – National Institute of Standards and Technology. 2012. Computer Security Incident Handling Guide – Recommendations of the National Institute of Standard and Technology, Special Publication, 800–61. Rev. ed. Accessed on 7 January 2020. Retrieved from http://dx.doi.org/10.6028/NIST.SP.800-61r2.

Ogaro, D. 2019. Hybrid cryptosystem in GRR Rapid Response. Email on 23 July 2019. Receiver Ahonen, Joni (GRR Users mailing list).

Ogaro, D. 2019. GRR Rapid Response's future work. Email on 6 December 2019. Receiver Ahonen, Joni (GRR Users mailing list).

Osquery documentation. N.d. Page on Osquery's website. Accessed on 17 October 2019. Retrieved from https://osquery.readthedocs.io/en/4.0.2.

Osquery. N.d. Page on Github’s website. Accessed on 7 September 2019. Retrieved from https://github.com/osquery/osquery.

Overview of Docker Compose. N.d. Page on Docker's website. Accessed on 7 December 2019. Retrieved from https://docs.docker.com/compose.

Performant endpoint visibility. N.d. Page on Osquery's website. Accessed on 9 September 2019. Retrieved from https://osquery.io.

PHR model. 2019. Page on Github’s website. Accessed on 18 December 2019. Retrieved from https://github.com/JYVSECTEC/PHR-model.

Production-Grade Container Orchestration – Automated container deployment, scaling, and management. N.d. Page on Kubernetes’ website. Accessed on 17 January 2020. Retrieved from https://kubernetes.io.

PRTG Manual – Docker Container Status Sensor. N.d. Page on PRTG Network Monitoring Software's website. Accessed on 1 November 2019. Retrieved from https://www.paessler.com/manuals/prtg/docker_container_status_sensor.

Reith, M., Carr, C., & Gunsch, G. 2002. An Examination of Digital Forensic Models. International Journal of Digital Evidence, 1, 3. Accessed on 18 January 2020. Retrieved from http://www.just.edu.jo/~Tawalbeh/nyit/incs712/digital_forensic.pdf.

Schema. N.d. Page on Osquery's website. Accessed on 12 November 2019. Retrieved from https://osquery.io/schema/4.0.2.

Security authorities develop their competence in National Cyber Security Exercise (KYHA19). 2019. Page on JYVSECTEC's website. Accessed on 5 December 2019. Retrieved from https://jyvsectec.fi/2019/11/security-authorities-develop-their-competence-in-national-cyber-security-exercise-kyha19.

Sereyvathana T., & Reed, T. N.d. Osquery – Cross-platform Lightweight Performant Host Visibility. Presentation at Open Source Digital Forensics Conference, 26 October 2016. Accessed on 3 January 2020. http://www.osdfcon.org/presentations/2016/Facebook-osquery.pdf.

Snover, J. 2016. PowerShell is open sourced and is available on Linux. Blog post on Microsoft Azure blog, 18 August 2016. Accessed on 6 January 2020. Retrieved from https://azure.microsoft.com/en-us/blog/powershell-is-open-sourced-and-is-available-on-linux.

Spotify. 2019. Page on Github’s website. Accessed on 6 September 2019. Retrieved from https://github.com/spotify/terraform-google-grr.

Sysmon modular. N.d. Page on Github’s website. Accessed on 28 January 2020. Retrieved from https://github.com/olafhartong/sysmon-modular.

Tietoverkkorikollisuuden torjuntaa koskeva selvitys. 2017. Publication of Ministry of the Interior of Finland. Accessed on 6 September 2019. Retrieved from http://urn.fi/URN:ISBN:978-952-324-136-7.

Valkama, S. 2019. Kyberrikoksista ilmoittaminen poliisille – kaupunkien ja poliisin välinen yhteistyö. Master’s thesis. University of Jyväskylä. Accessed on 6 September 2019. Retrieved from http://urn.fi/URN:NBN:fi:jyu-201904252266.

Whacking a Million Moles – Automated Incident Response Infrastructure in GCP. 2019. Page on Spotify’s website. Accessed on 31 August 2019. Retrieved from https://labs.spotify.com/2019/04/04/whacking-a-million-moles-automated-incident-response-infrastructure-in-gcp.

What is MySQL?. N.d. Page on MySQL's website. Accessed on 19 January 2020. Retrieved from https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html.

Appendices

Appendix 1. Deployment manual for containerized GRR server

# Prerequisites

Containerized GRR server is successfully tested on following operating systems:

Ubuntu 18.04 Bionic (1 dual-core CPU, 8 GB RAM) with following packages and their respectively versions:

- Docker Engine and Client v19.03.1
- Docker Compose v1.24.1

CentOS 7 (1 dual-core CPU, 8 GB RAM) with following packages and their respectively versions:

- Docker Engine and Client v18.09.6
- Docker Compose v1.24.0

Docker images used during deployment process (hosted on Docker Hub):

- grrdocker/grr:v3.3.0.8
- mysql:5.7
- nginx:latest
- prom/prometheus:latest

GRR agents are successfully tested on following operating systems and versions:

- Windows Workstation 7, 10
- Windows Server 2008, 2012
- CentOS 7
- Ubuntu 18.04

# Quick deployment

## GRR server

Deployment of containerized GRR server is designed to be a straightforward process. After you have executed the requirements shown in prerequisites section you should be able to deploy GRR server using following commands:

- git clone https://github.com/JYVSECTEC/containerized-grr
- cd ./containerized-grr
- bash setup.sh
- docker network create --driver=bridge --subnet=<SUBNETWORK> static
- docker-compose up --build --detach

If no errors occur command “docker-compose ps” should inform that there are now six containers up and running:

- grr-admin
- grr-front
- grr-proxy
- grr-mysql
- grr-worker
- grr-prometheus

Now you can access the administrator GUI by browsing by Nginx IP address or URL and use the credentials provided during deployment process (default: admin/grr). However, you must install GRR agents on those clients that you want to examine before you can really execute any forensic tasks on endpoints. Consult the "GRR agent installation on clients" section for additional information.

## GRR agent installation on clients

After GRR Server is successfully deployed, client installers can be examined. GRR Server populates “./containerized-grr/installers” directory with installer packages for both 32-bit and 64-bit operating systems:

- dbg_GRR_x.x.x.x_amd64.exe
- dbg_GRR_x.x.x.x_i386.exe
- grr_x.x.x.x_amd64.changes
- grr_x.x.x.x_amd64.deb
- GRR_x.x.x.x_amd64.exe
- grr_x.x.x.x_amd64.pkg
- grr_x.x.x.x_amd64.rpm
- grr_x.x.x.x_i386.changes
- grr_x.x.x.x_i386.deb
- GRR_x.x.x.x_i386.exe
- grr_x.x.x.x_i386.rpm

Depending on the client operating system, you can push the correct installer package to the client and execute it, or install agent using respective package manager:

- # On Red Hat based Linux distros
- yum install grr_x.x.x.x_amd64.rpm
- # On Debian based Linux distros
- dpkg --install grr_x.x.x.x_amd64.deb
- # On Windows operating systems
- .\GRR_x.x.x.x_amd64.exe
- # On MacOS
- sudo installer -pkg grr_x.x.x.x_amd64.pkg -target /

# Configuration explained

## Authentication

In future containerized GRR will support various authentication methods, but currently there are two tested methods available (defaults to Remote Authentication):

- Basic Authentication – Username and password are generated during setup process of GRR server and stored on the database
- Remote Authentication – GRR server trusts authentication that the Nginx reverse proxy handles.

## Database

MySQL database files are mounted to host side of system to prevent any data loss if the container execution terminates.

## Monitoring

Containerized GRR includes Prometheus monitoring system which enables investigator to observe the status of each GRR sever component and query monitoring data for occurred changes. However, it should be noticed that the monitoring system is only implemented to bring additional feature for the server execution. Any extensively testing of Prometheus is not made.

## Proxy

Containerized GRR utilizes Nginx proxy which handles the traffic between GRR agents and GRR HTTP front-end. In addition, administrative user interface is accessed via proxy, and by default it handles the user authentication.

Creating a new self-signed certificate for the proxy:

- cd ./containerized-grr
- openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout ./nginx/cert.key -out ./nginx/cert.crt

Creating a new username and password for Remote Authentication:

- cd ./containerized-grr
- sh -c "echo -n '<USERNAME>:' >> ./nginx/.htpasswd" && sh -c "openssl passwd -apr1 >> ./nginx/.htpasswd"

## Osquery

Containerized GRR supports also centralized management of osquery agents. GRR is configured to search osquery binary on clients from its default installation path, so it is advisable to keep binaries on their default installation location. Osquery is successfully tested in containerized GRR using the osquery version 4.0.2.