TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Giorgi Zeikidze 194437IVSB

Evaluating and Remediating the Performance Impact of Vulnerability Management Software at Pipedrive

Bachelor Thesis

Supervisor: Kristian Kivimägi
MsC
Co-Supervisor: Kieren Niĉolas Lovell
Lt CDR

Tallinn 2022

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Giorgi Zeikidze 194437IVSB

Pipedrive turvanõrkuste haldustarkvara jõudluse mõju hindamine ning parandamine

Bakalaureusetöö

Juhendaja: Kristian Kivimägi
MsC
Kaasjuhendaja: Kieren Niĉolas Lovell
Lt CDR

Tallinn 2022

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Giorgi Zeikidze

16.05.2022

Abstract

Vulnerability management is a continues process, to keep up with the demands of modern cyber-space the process requires improvement with each iteration. Given work is an in depth analysis of yet another improvement of vulnerability management at Pipedrive. Granted that existing vulnerability detection solution is limited in functionality, implementation of hybrid model of vulnerability assessment was proposed. This required deployment of Nessus agent, in-house solution of Tenable.io, the provider of company's vulnerability management software. In the scope of the thesis Nessus agent installation automation was developed and performance of the agent was tested. Initial tests provided unacceptable results in terms of memory use, creating the need of workaround in form of memory limitation using systemd. Proof of concept was designed and deployed in one of infrastructure regions. Trail on the region scale was unsuccessful, the effective way to limit the performance impact remains to be discovered.

List of abbreviations and terms

API	Application Programming Interface
CPU	Central Processing Unit
IDE	Integrated Development Environment
IOT	Internet Of Things
VM	Virtual Machine
CLI	Command Line Interface
CRM	Customer Relationship Management
K8s	Kubernetes
IaaC	Infrastructure as a Code
LTS	Long Term Support
IaaS	Infrastructure as a Service
SOC	Service Organization Controls
SLA	Service Level Agreement
OOM	Out off Memory

Table of Contents

List of Figures

1 Introduction

Many of the recent happenings in the world have proven that Cyber-attacks pose a severe threat to organizations. It has become especially simple to carry out those due to various publicly and privately available tools with incredibly low complexity of usage. Cyber-attacks can have many different consequences, including financial losses, reputational damage, and loss of customer trust. In some cases, they might also lead to regulatory investigations and fines. One of the many reasons for cyber-attacks is unmitigated vulnerabilities. The word vulnerability can have many different meanings. By the broadest definition, vulnerability refers to the state of being exposed. From a security perspective, vulnerability resembles something that can be exploited by outside forces or actors to sidestep our defenses, be it a breach in stronghold walls allowing the enemy in or a door left unlocked allowing a criminal inside. While in the field of information technology, vulnerability means "A flaw or weakness in a computer system, its security procedures, internal controls, or design and implementation, which could be exploited to violate the system security policy, as defined by Park Foreman[1]."

Almost any business today is either directly or indirectly affected by these vulnerabilities that carry a risk for normal day-to-day operations. One of the recent studies suggests that in 2019, 60 percent of data breaches could have been prevented by timely patching of vulnerabilities. [2] Due to the constantly increasing number of vulnerabilities nowadays, it has become impossible to patch all of them. To address this issue, a separate field of vulnerability management has been developed. Vulnerability management is not a standalone technology, it is a combination of processes and practices aiming to identify, classify and mitigate risks connected to existing vulnerabilities. Due to the complexity of this approach, it is mainly aimed at organizations and government bodies. [1]

The broader objective of this paper is to improve vulnerability management inside Pipedrive. Pipedrive is an IT company founded in 2010 in Estonia. The company follows the software as a service monetization model, offering a CRM solution. The main focus of the software has always been simplicity and ease of use. Its intuitive design and kanban-based approach to sales make Pipedrive a very compelling product for small to medium businesses and startups. Currently, the company serves more than 100 000 different customers worldwide. Each subscribed customer can have many users (or seats as Pipedrive calls it). Most of the clientele are other businesses that have some or all employees using the tool, making the total number of software users more than a million.[3]

It should be no surprise that a company with this amount of user data takes the risk of vulnerabilities seriously. Therefore it constantly introduces new advancements to the existing vulnerability management program. Therefore, the primary goal of the thesis is to improve the current state of vulnerability management at Pipedrive, achieving a higher level of security. Principles and solutions provided in the research can be used to implement similar solutions in companies with modern infrastructure architecture that focus on observability and reliability.

The main objectives throughout the given work will be evaluating the performance of vulnerability checking solutions and their impact on mission-critical infrastructure as well as mitigating this impact. The author assumes at least a limited understating of Linux system administration topics and general knowledge of Unix/Linux concepts. This research aims to outline the factors that need to be accounted for, before implementing vulnerability management software in a business-critical environment. It also provides a guide to alleviating some of the negative effects.

2 Background Information

2.1 IT Infrastructure in Pipedrive

To understand the solutions offered in this paper it is of vital importance for readers to understand the inner workings of Pipedrive's IT infrastructure. One of the major reasons for successful growth is the company's persistent and sustainable scaling. The infrastructure department is the organizational body directly responsible for keeping the services' performance consistent during the growth of the user base and the features that the software offers. Different teams under the infra umbrella work on various internal services to keep the end product running smoothly.

2.1.1 Infrastructure Engineering Practices

Growth and scaling of the product is directly enabled by cloud based architecture, but not every cloud infrastructure is built equally, being cloud based does not automatically guarantee scalability or sustainability of growth. To keep the network latency low in different geographic areas, ensure high availability of the product and the company operates multi-cloud, multi-region infrastructure. Number of these regions is ever increasing, there are currently more than twenty different regions, including four live ones currently running the main application and several more are being built. To keep up with the non-trivial size of the infrastructure below mentioned tools and practices are used.

Cloud Services Providers

As mentioned above infrastructure of Pipedrive is deployed on cloud, mainly on two providers: AWS and Rackspace. Rackspace is where the bigger part of the regions of Pipedrive's infrastructure reside. Resources there are managed using Openstack, an open source cloud computing platform developed by Rackspace and NASA[4]. Amazon's is another IaaS provider for Pipedrive. While not as many regions are deployed there, it still houses some of our mission critical environments. AWS has its own property cloud management system accessible via different APIs and tools.

Infrastructure as a Code Approach

Infrastructure as a code is an approach to infrastructure automation that makes use of software development principals. In code desired state of infrastructure, condition of machine or configuration of system is described in declarative format. Then the code is rolled out through unattended processes whilst being validated.[5] IaaC approach plays crucial role in many of the modern practices of development, including DevOps, CI/CD, cloud computing, microservice based architecture and etc.

This approach brings many benefits:[5]

- Automating processes, especially complex ones lowers the chance of human error being introduced in system.
- Time of the engineers is no longer wasted on manual procedures, making the size of the infrastructure non-issue.
- By lowering the cost of change to almost nonexistent, changes become routine, testing becomes easy, this in turn lowers the chance of error and improves speed of improving mistakes whenever they may occur.
- Prototyping is easier so less risks are taken and less time is spent on planning meetings.
- Having state of infrastructure mostly or fully visible as code.

Infrastructure as a Code Tools

Terraform is an infrastructure provisioning tool created by Hashicorp. It has integration for most cloud computing platform APIs including above-mentioned Openstack and AWS. As the allocation process of computing resources is not related to the thesis, a deeper analysis of the software will be omitted.

Chef-infra, is a configuration automation solution[6] and the main tool currently used for infrastructure management at Pipedrive. Chef-infra is the oldest one of many tools offered by its respective company and is commonly referred as just Chef[7]. The tool follows server-client based model where the configuration code is uploaded from the chef-server to chef-clients on the managed machines.[6]

Ansible is another configuration management tool. It is also a runner up for the role of go to solution in Pipedrive's infrastructure, there is also an ongoing migration happening of

infra-team's code-base from Chef to Ansible but the project is still at early stages so as of now both tools are being used. There are also certain parts of the infrastructure managed solely with Ansible.

Container orchestration engine Kubernetes is another IaaC tool and crucial part of Pipedrive's infrastructure, but containerization technologies are outside the scope of the research, reason for this will be addressed in upcoming chapters.

Operating Systems

For simplicity and consistency all of the company's managed infrastructure runs on Debian based systems, predominantly Ubuntu LTS. Thus all the solutions described further in this work should be assumed to be running on Ubuntu 20.04. The solutions provided here in terms of code work on this OS, in case of other Linux distributions or other Unix like operating systems some discrepancies might be experienced. Even still general principals and practices remain applicable.

2.1.2 Networking Infrastructure

Another major part of the IT infrastructure is the networking, including both, our office networks and networks of our deployment. These networks are managed by our networking team, including DevNetOps and Network Engineers. While network analysis is a major part of vulnerability management process, the current state of the network awareness is above satisfactory. Thus it falls outside the scope of the work being done here. It is also worth to consider that network analysis falls outside of the authors abilities and responsibilities.

2.2 Standards and Frameworks

In the field of cyber-security it is general understood that being security and proving that you are secure at two different challenges, with equally different aims. While achieving security is not a small challenge by itself, proving it is on another level of complexity and with the complexity arises the need for structure. Such structures come in the form of standards and frameworks.

2.2.1 ISO27001

To tackle complex issue of prove of security Pipedrive, like many other organizations opted for ISO27001 certification[8]. ISO 27001 is the international standard that provides the

specification for an information security management system or ISMS. This is a systematic approach consisting of people processes and technology that helps you protect and manage all the organization's information through risk management. An ISMS, particularly one that conforms to ISO 27001 can help organizations comply with a host of laws including the high profile general data protection regulation commonly referred to as the GPPR and the network and information systems regulations also known as the MIS regulations. [9]

ISO 27001 focuses on protecting three key aspects of information, also known as CIA triad:

- Confidentiality.
- Integrity.
- Availability.

Confidentiality means that the information is not available or disclosed to unauthorized people entities or processes. Integrity means that the information is complete and accurate and protected from corruption. Availability means that the information is accessible and usable as and when authorized users require it.[9]

ISO 27001 is one of the most popular information security standards in the world, with the number of certifications growing by more than four hundred and fifty percent in the past ten years. T he standard has been designed to help organizations manage their security practices consistently and cost-effectively. its technology and vendor neutral and is applicable to all organizations, irrespective of their size type or nature. ISO 27001 is the mainstay of the ISO 27000 series a family of mutually supporting information security standards that together provide a globally recognized framework for best practice information security management. these standards help organizations keep their information assets secure by offering a set of specifications codes of conduct and best practice guidelines to ensure strong information security management.[9]

2.2.2 SOC

Other certification that the company has is SOC 2 and SOC 3. [10]

SOC was developed in 2011, by the American Institute of Certified Public Accountants, as a replacement for Statement on Auditing Standards. SOC 2 defines the criteria for managing customers data based on the five service principles:[11]

- Security.
- Availability.

- Processing integrity.
- Confidentiality.
- Privacy.

SOC 2 and SOC 3 reports are unique per body being certified and are in line with specific business practices. Each organization designs its own controls to comply with one or more of the trust principles. The First principle, security refers to the protection of system resources against unauthorized access. Access controls help prevent potential system abuse theft or unauthorized removal of data, misuse of software, and improper alteration or disclosure of information. IT security tools such as network and web application firewalls, two-factor authentication and intrusion detection software are useful in preventing security breaches that can lead to unauthorized access of systems and data.[11]

The availability principle refers to the accessibility of the managed service as stipulated by the SLAs. As such, the minimum acceptable performance level for systems availability is set by both parties. This principle doesn't address system functionality and usability but does involve security related criteria that may affect availability, monitoring network performance and availability, site fail-over, security incidents handling are critical in this context. [11]

The processing integrity principle addresses whether the system achieves its purpose: does it deliver the right data, at the right price, at the right time? Accordingly, data processing must be complete, valid, accurate, timely and authorized. However, processing integrity does not necessarily imply data integrity. If data contains error prior to being input into the system, detecting them is not usually the responsibility of the processing entity. Monitoring of data processing coupled with quality assurance procedures can help ensure processing integrity. [11]

Confidentiality - data is considered confidential if access and disclosure is restricted. This may include data that's only intended for company personnel, or its business plans, or intellectual property, it might be financial data. Encryption is an important control for protecting confidentiality during transmission. Network and application firewalls together with rigorous access controls can be used to safeguard information being processed or stored on a computer system. [11]

And finally the privacy principle addresses the system's collection, use, retention disclosure and disposal of personal information in conformity with an organization's privacy policy, as well as within the criteria that's set by SOC 2.[11]

Unlike SOC 2 report which is restricted use only SOC 3 is a general use report, usually available on company's site, including at Pipedrive.[10, 11] SOC 3 report does not provide controls and only includes limited control details in description.[11]

2.2.3 CMMI

Improvement of organisational processes is another complicated topic that requires a systematic approach to navigate. While CMMI is not really aimed at security related processes it still provides much needed structure to orient by. More abstracted view of the challenge gives much needed perspective to understand the context of the problem.

According to CMMI when it comes to the continues processes there are 5 different stages of maturity:[12]

1. Initial
2. Managed
3. Defined
4. Quantitatively Managed
5. Optimizing

At initial stage the process is very chaotic, tasks are performed ad hoc, there is no defined structure, no procedures in place. In such environment success depends on the abilities of the team at hand and outcomes vary vastly depending on their competence.[12] This maps nicely to security processes in general, at this stage internal processes for vulnerability management are usually non-existent. These process require vast investment time and resources to be established which is not possible in chaotic environment. Audits at this point are strictly external, performed once a quarter or yearly.

Improving on this the managed stage has planed, policy based approach. Still, most of the processes are manual and reactive. Teams at this stage become more disciplined, less dependant on the individuals skill, leading to better time management.[12] In the scope of security this is where the internal process vulnerability management comes into play.

Defined stage is logical evolution of the second stage, here processes while similar to the stage two become proactive rather then reactive. At this point processes, tools and methods are well established. [12] This is where the automation usually comes into the picture. In case of vulnerability management at this stage of maturity most of the menial processes should be automated. Response procedures should be well organized and understood.

Pipedrive's vulnerability management process falls somewhere between stage two and

three, given thesis being another step towards the third stage. Considering this stages four and five are less relevant to this work, thus they'll be defined more briefly:

- Quantitatively Managed - this is where the mitigation steps and responses get automated.
- Optimizing - the cutting edge of security, the processes of discovery and response are being improved actively.

3 Problem Statement and Assignment

The Transitional period from a start-up to an enterprise organization is associated with a vast amount of challenges along with legal and financial responsibilities. The year 2021 was the beginning of Pipedrive's transition into an enterprise, which led to the restructuring of the information security team. The main result of those changes was a set of new requirements for the information security team.

Pipedrive has already been certified with System and Organization Controls (SOC) 2 and 3 along with ISO27001:2013 [10][8]. Despite the fact that these certifications have proven the security, availability, processing integrity, confidentiality, and privacy of the system[13], their effect is not permanent. These require to be maintained over time, which involves continuous improvement of the information security process. In addition, as these certifications do not necessarily fulfill their requirements, some refinements were requested by third parties, such as cyber insurance. One of the subjects of the discussion has been SLA for vulnerability remediation. Due to these circumstances, there has been a great emphasis on this topic from the stakeholders' side. Therefore, currently, Pipedrive's information security team is actively working on increasing the efficiency of the vulnerability management system.

The ultimate goal of this paper is to contribute toward increasing the maturity of the vulnerability management system within Pipedrive. This will be achieved by researching and implementing the best practices of venerability detection. The main flaws of the current system have been discussed with the information security team and the criteria for the success have been established.

The current vulnerability detection relies on a Nessus scanner with limited capabilities. For patch auditing the scanner uses a local user, access to this user is currently provided via ssh (asymmetric key authentication), this provides ssh access from the Nessus machine to any asset that is being scanned, and due to the obvious security problems with this approach the dedicated Nessus user has only limited permissions. This limitation in access often provides false positives and leaves a reasonable suspicion that some vulnerabilities remain undetected.

To further increase the effectiveness of vulnerability scanning and address above mentioned security risk, the implementation of a hybrid model of vulnerability detection was suggested

by the information security team. The resulting system will use the Nessus scanner and Nessus agent for assessing different types of vulnerabilities. Scanners are already implemented in the company's mission-critical systems, but when it comes to the agents there are several problems that remain to be addressed:

The footprint of the agent software has not been studied. The impact of the security audit on the systems needs to be measured and considerations need to be made so that running these checks does not impact end-users in any noticeable way. Pipedrive's main objective is to provide its clients with satiable service.

The focus of this work is to improve on the existing approach for auditing by examining the viability of the hybrid architecture. The tasks undergone are:

- Creation of installation and configuration automation for Nessus agent.
- Estimation of the baseline resource use of the agent.
- Investigation of the likely impact on mission-critical services.
- Addressing the performance problems detected to achieve the robustness of the approach.

4 Methodology

4.1 Vulnerability Assessment Approaches

When it comes to the architecture of the Vulnerability assessment there are five principal approaches[1]:

- Scanning
- Passive assessment
- Agents
- Hybrid
- Inference

Of these five the inference and the passive assessment are beyond this thesis's scope, still the brief description is in order. Inference method employs data analysis techniques on data obtained through means outside of the security measures[1]. Snyk currently used at Pipedrive can be considered as such a tool. Passive assessment also relies on something that already existing, in this case notwork security appliances, this appliances can be used to analyze the traffic to detect vulnerabilities[1]. Other three approaches are important part of this work and as such will be described in more detail below.

4.1.1 Scanner-based Approach

To understand rationale behind hybrid model being preferred over scanning, it is useful to understand the method. Scanner is a software that generates network packets to actively engage the targets in order to detect their presence and vulnerabilities.[1] This emulates the standard behaviour of the malicious actor[1] that got access to the network - Port enumeration, the primary method of reconnaissance. Scanner is deployed on a regular machine connected to the network being scanned.

The existing approach is employing Nessus scanner, the software is provided by Tenable.io vulnerability detection system. This approach does have its benefits: Currently the scanners are already deployed in all of the live regions. It would save human resources, addressing the problem of developing the installation and configuration automation as well as going through the process of the deployment. Remote scans are less intrusive in terms of the impact on the operation of the host being scanned, they are less likely to cause instability

and require no in depth analysis of the process as the commands that are used for the probing of the installed processes are available in history of the user that was used to run them.

Considering this agents might hardly seem like the improvement but scanners come with their own inherent problems to consider: Firstly and most importantly the security risks related to the direct shell access that is required for the scanner based patch auditing. Granting ssh access from a single server to *every* server is a risk that can not easily be overlooked. Especially when the requirements are either a root access or the comparable permissions granted per-file or per-command bases, using chmod/chown[1] and sudoers[2]file respectively. Such permissions carry a risk of privilege escalation attack, therefore such access is better avoided.

Another problem occurs as a result of the above described per-command/per-file workaround. The need raises to automate sudoers and file permissions changes on the machines that are being scanned. While the operating systems are the same across the board with machines serving a very different purpose from one another this configuration becomes almost unmanageable. Furthermore each update of the agent or the introduction of the new plugin or type of scan will require rigours testing to adjust the permissions to the new configuration.

4.1.2 Agent-based approach

Agent is as software ran on the machine that is being scanned, it runs with the sufficient privileges to perform the necessary check, they are linked to the management solution and provide required data for assessing vulnerabilities on the system.[1]

This mode of operating on the machine being scanned completely negates scanners biggest flaw, ssh authentication as the roles are reversed and it's now the client who initiates the connection. With this benefit however come the flaws as well, agent is a service running on the system that is mission critical and its impact on stability of such machines can have a disastrous impact on the business.

Nessus agents also lack the ability to do network scanning, the agent performs rudimentary checkup of open port using net on the machine it's ruining on, while this eliminates the unnecessary open ports issue on the hosting machine, visibility of other hosts on network, for example networking equipment is none. This also relates to another major flaw of the

[1]The commands adjust the ownership of the file in Linux system

[2]This file is used to grant administrative privileges, to the user, it can also grant permissions to run certain commands as root

agents, they have no way to discover rogue hosts - machines that should not be on network but are. This could be a machine not properly taken down by the engineer or an attacker introduced agent used to gain access to internal network. In both cases noticing such a host is of crucial importance.

4.1.3 Hybrid approach

Hybrid means a combination of two things. this could mean any combination of the approaches. So for the clarity's sake, the hybrid model that this thesis addresses is the combination of Nessus agent and Nessus scanner. With this comparison matrix it becomes

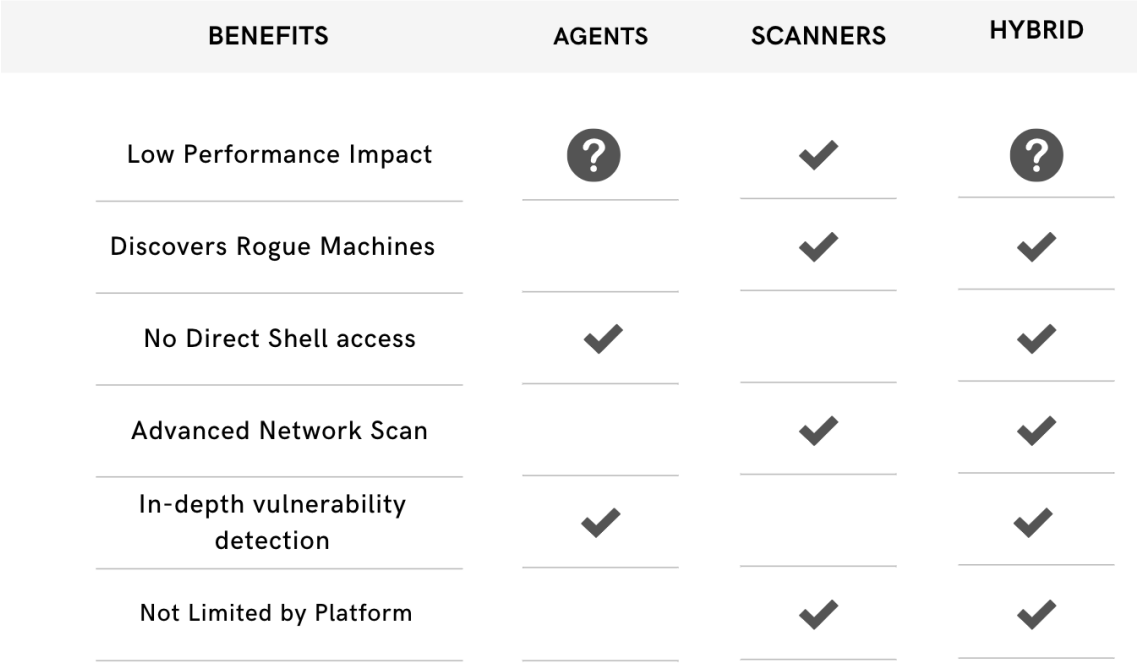

Figure 1. *Agent vs. Scanner vs. Hybrid.*

self evident that the hybrid model is superior compared to two alternatives, but the question of performance impact of the agent still stands.

4.2 Understanding Linux Services

To measure the resource use of the service it is important to understand what service is. Service in operating systems case is generally something that runs in a background, automatically. It is also worth to mention that the definitions here will be in scope of the systemd, which is more or less a standard for modern Linux distributions. systemd is initialization system, it initializes the operating system during boot and continues managing system processes and services afterwards.[14]

Services for systemd exist as service unit file. These files have the extension of ".service" and are typical configuration files. It consists of one or several processes, a service can also be a combination of several other sub-services. Processes under the service are combined in a control group. control group usually referred as cgroup is a process organization mechanism inherent to Linux kernel.[15] While cgroup is not a part of systemd, its functionality still plays an integral role in the process of the service management.[14]

4.3 Testing Ground

The initial idea was to try out the agent in testing environment, but this proved harder than expected. Providing connection from testing environment to live servers is an unnecessary risk. Testing setup requires licence for tenable products and other non-trivial setup and while a trail license can be obtained process is cumbersome. Luckily a new live region, referred as PDX, was recently built. Currently there are some tests being done by different teams to guarantee proper function of all services, only after this testing stage will the region be open to customers. Testing stage usually takes one to two weeks, this provides a suitable environment for testing the agents. Consequently a decision was made to do initial deployment of the Nessus agents there.

4.4 Kubernetes

As already mentioned auditing containers is outside of the scope of this research; Thus the Kubernetes nodes will be treated as typical machines and no work will be done inside Kubernetes clusters. Reasons for this are twofold:

While the pods do in fact operate as a typical Linux machine, evaluating and patching software for a container requires work with the image and a configuration file, not with CLI of an actual machine like it would in a typical Linux VM.

There are already some checks in place for images that are deployed to Kubernetes, mostly done by Snyk software that scans the company's git repositories including docker files and evaluates them, but discussing this solution also falls outside the scope of the research.

5 Practical Implementation

5.1 Installation Process of Nessus Agent

Automating the installation of the Nessus agent is not a trivial process. Unlike typical Linux package, the agent is not available through the package manager. Nessus has no private repository and their binaries are not hosted on public ones due to the licensing. Approach offered from Tenables official site is not quite suitable for a large scale architecture as mentioned in their own documentation [16]

Alternative offered by the Information Security team was a short script2 that makes an API call to Tenable cloud with a specific license key, response is a script that downloads, installs and configures the agent automatically.

```
curl -H \
'X-Key: ${LICENCE_KEY}'\
'https://cloud.tenable.com/install/agent?name=\
${AGENT_NAME}&groups=Servers,${REGION_DOMAIN}'| bash
```

Figure 2. *The installation script for Nessus agent.*

But after some discussion it was decided that running proprietary scripts from a third party, even if this third party is an official provider should still be avoided, thus the decision was made to distribute the package from our own storage solution using chef and ansible (configuration management tools).

Another peculiarity of the installation process is the linking, or how it fails. If the agent was uninstalled form the machine its unique identifiers, defined per machine are not removed form tenable cloud, in such case the registration fails, to avoid

Nessus documentation warns against installing agents on a large scale infrastructure all at the same time, this could cause a network congestion, while they are talking about thousands of machines, it's still worth to consider the effect of all machines upgrading at the same time. [16]

5.2 Measuring Resource Use

There are countless ways to measure the resource use of the process on Linux but deriving the service level measurement from this is somewhat of a complex challenge. Thus to measure with the least amount of complexity and get the result that is easily readable systemd's accounting property was used. Commands: The output was ingested in the bash

```
systemctl set-property nessusagent CPUAccounting=yes
systemctl set-property nessusagent IPAccounting=yes
systemctl set-property nessusagent MemmoryAccounting=yes
```

Figure 3. *Enable performance accounting in systemd.*

script, streaming the time series data to JSON file, which was later visualised using Python three's matplotlib and pandas libraries.

Usually when performance is being measured it is worth to consider the idiosyncrasy of different environments, but with the environment being fully virtual, with the same architecture and the same operating system disparities are not likely. Thus the cloud environment of the tests is not relevant to the outcome.

5.3 Analysis of the Agent

Initial observation on several machines revealed that in certain cases Nessus agents use around 350 megabytes of memory while not actively performing an audit. Impact of such high RAM utilisation can become a major problem for smaller machines, especially ones with 4 gigabytes of memory and is still noticeable with 8 or even 16 gigabytes. This is huge increase compared to 40 megabytes Tenable documentation describes [17]. Memory use during the scan became even more noticeable.

5.4 Limiting the Memory Use

With systemd being as useful as it was with the performance measurement, it was only logical to depend on the service to limit the agents memory use with systemd configuration. For testing purpose the drop-in configuration was used, created via the command below.[18]]

This limits the memory use allowed to the processes in this services cgroup and in case the process exceeds the limit it is killed by the OOM killer.[18]

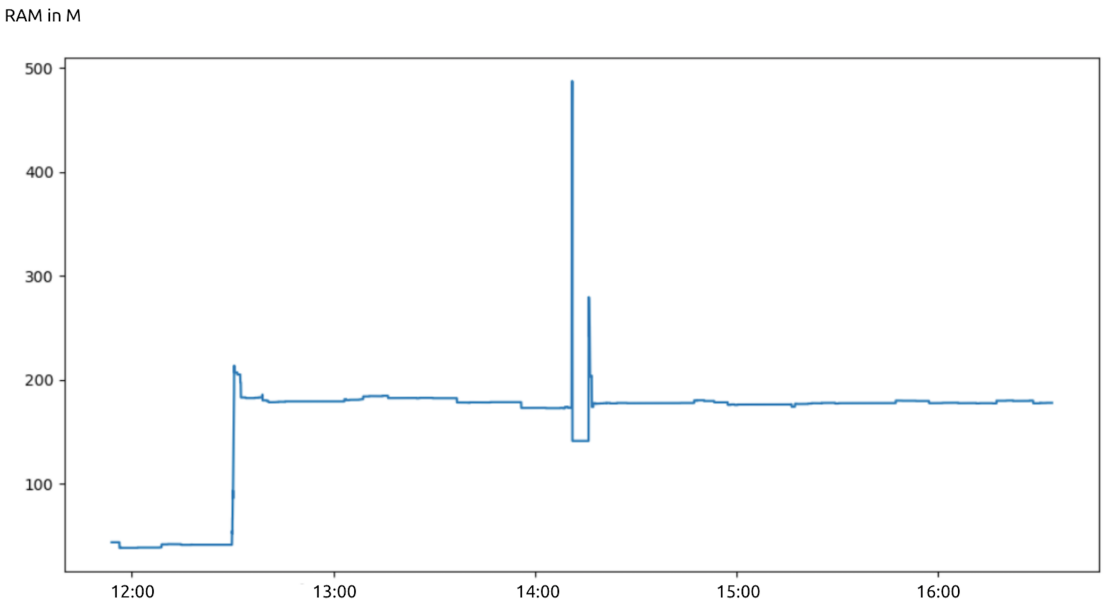

Figure 4. *Memory throughout initial scan.*

```
systemctl set-property nessusagent MemoryMax=50M
```

Figure 5. *Enable performance accounting in systemd.*

The thought process behind this decision was as follows:

- 50M should be enough as the idle service is specified to be using around 40M[17]
- In case the OOM killer kills the service it can be restarted by service config using "restart=on-abort" property it has by default.[14, 18]
- Even if the agent restart fails the audit scan can easily be re-initiated, yet the performance problem will be resolved.

The check using this solution was performed on two different hosts, on three separate instances with all three of them succeeding to report back to Tenable.

5.5 Implementing the Limit on Regional Level

After the success on a single machine the limit was implemented on the region wide scale where it failed miserably. The limit was increased first to 100 and later to 200 megabytes, yet the outcome was not changed. With OOM killer killing the service again and again. This accumulated the errors from several hundred machines which were forwarded to live log aggregating solution, effectively denying company's internal service. As the implication of this incident the trails will no longer continue in PDX or any other live region.

6 Summary

Tasks undertaken throughout the work demonstrate that with the existing state of the affairs Nessus agent is not ready to be deployed in any of the live regions.

Few important takeaways from the undergone processes are:

Nessus documentation while very vast is also somewhat limited and outdated. Agents default configuration does not manage to mitigate the impact on underlying system. This calls for the use of external tools to limit achieve this, in given case systemd did not show it self as such tool.

Testing in live environments is dangerous and can have a disastrous impact, even if the live region is not hosting clients and their data, such approach is always better avoided.

With these considerations the new testing environment needs to be developed to proceed with the testing of the hybrid model.

Bibliography

[1] Park Foreman. *Vulnerability Management*. Auerbach Publications, 2010.

[2] *Costs and Consequences of Gaps in Vulnerability Response*. Tech. rep. Ponemon Institute. URL: https://media.bitpipe.com/io_15x/io_152272/item_2184126/ponemon-state-of-vulnerability-response-.pdf.

[3] "Accessed: 10-05-2022. URL: https://www.pipedrive.com/about.

[4] Juan Angel Lorenzo del Castillo, Kate Mallichan, and Yahya Al-Hazmi. "OpenStack Federation in Experimentation Multi-cloud Testbeds". In: *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*. Vol. 2. 2013, pp. 51–56. DOI: 10.1109/CloudCom.2013.103.

[5] Kief Morris. *Infrastructure as code: managing servers in the cloud*. " O'Reilly Media, Inc.", 2016.

[6] "Accessed:11-05-2022. URL: https://docs.chef.io/chef_overview/.

[7] "Accessed:11-05-2022". URL: https://docs.chef.io/platform_overview/.

[8] "Accessed: 27-04-2022". URL: https://www-cms.pipedriveassets.com/documents/ISO-IEC-27001-2013-Certificate-MRAS-Pipedrive-v12.22.2021-1.pdf.

[9] Ahmad Nurul Fajar, Hendy Christian, and Abba Suganda Girsang. "Evaluation of ISO 27001 implementation towards information security of cloud service customer in PT. IndoDev Niaga Internet". In: *Journal of Physics: Conference Series*. Vol. 1090. 1. IOP Publishing. 2018, p. 012060.

[10] "Accessed: 27-04-2022". URL: https://www-cms.pipedriveassets.com/documents/Pipedrive-2021-SOC-3-Final-Report-1.pdf.

[11] Vickie Choe, David Taylor, and Aleksei Brizhik. "SOC 2 breakdown: a five-part guide to understanding the service organization controls 2 report and its benefits". In: *Internal Auditor* 69.1 (2012), pp. 54–59.

[12] CMMI Product Team. "CMMI for Development, version 1.2". In: (2006).

[13] "Accessed:10-05-2022". URL: https://www.ssae-16.com/soc-2/.

[14] Donald A Tevault. *Linux Service Management Made Easy with systemd*. English. 2022. ISBN: 9781801815031.

[15] "Accessed:12-05-2022". URL: https://www.kernel.org/doc/html/latest/admin-guide/cgroup-v2.html.

[16] *Nessus Agent Large Scale Deployment Guide*. "Last Revised: March 31, 2022", "Accessed: 27-04-2022". URL: https://docs.tenable.com/other/nessusagent/Nessus_Agent_Large_Scale_Deployment_Guide.pdf.

[17] "Accessed: 28-04-2022". URL: https://docs.tenable.com/generalrequirements/Content/NessusAgentHardwareRequirements.htm.

[18] "Accessed:11-05-2022". URL: https://www.freedesktop.org/software/systemd/man/systemd.service.html#.

Appendix 1 – Non-exclusive license for reproduction and publication of a graduation thesis[1]

I, Giorgi Zeikidze

1. Grant Tallinn University of Technology free license (non-exclusive license) for my thesis "Evaluating and Remediating the Performance Impact of Vulnerability Management Software at Pipedrive", supervised by Kristian Kivimägi
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive license.
3. I confirm that granting the non-exclusive license does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

16.05.2022

[1]The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 - Disclaimer

Given work is done in collaboration with Archil Kristinashvili, thus Introduction, background, general problem statement and general summary chapters will be the same.