Professional thesis report

DE MIGUEL ARAMBURU, Ander

September 2020

VULNERABILITY MANAGEMENT IN ORGANIZATIONS

Company : SES Astra S.A.

Company Supervisor : John MCHALE

Academic Supervisor : Prof. Marc DACIER

Non-confidential

EIT Digital Master in Cyber Security

EURECOM

Rapport de Thèse Professionnelle

DE MIGUEL ARAMBURU, Ander

Septembre 2020

GESTION DES VULNERABILITES DANS L'ENTREPRISE

Société : SES Astra S.A.

Encadrant dans l'entreprise : John MCHALE

Encadrant académique : Prof. Marc DACIER

Non-confidentiel

EIT Digital Master in Cyber Security

EURECOM

ABSTRACT

The continuous increase of cyber-attacks taking advantage of unpatched vulnerabilities, pushes the organizations to implement a vulnerability management framework to stay protected. This thesis addresses the issue of a correct vulnerability management, studying the problematic in different phases of the vulnerability lifecycle and providing best practices to reduce vulnerabilities in each of them. Moreover, a real-life case study is presented through an internship in SES's Space Operations department. During the 6-month internship a semi-automated patch management system is configured and used to contribute to the reduction of vulnerabilities discovered in the department's systems.

RESUME

L'augmentation continue des cyber-attaques profitant de vulnérabilités non corrigées, pousse les organisations à mettre en place un cadre de gestion des vulnérabilités pour rester protégées. Cette thèse aborde la question d'une gestion correcte des vulnérabilités, en étudiant la problématique dans les différentes phases de son cycle de vie et en fournissant les meilleures pratiques pour réduire les vulnérabilités dans chacune d'elles. De plus, une étude de cas concrète est présentée à travers un stage dans le département des opérations spatiales de SES. Pendant le stage de 6 mois, un système de gestion des correctifs semi-automatisé est configuré et utilisé pour contribuer à la réduction des vulnérabilités découvertes dans les systèmes du département.

CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABREVIATIONS

BSIMM	Building Security in Maturity Model
BTZ	Betzdorf
CLI	Command Line Interface
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
DEV	Development
DMZ	De-Militarized Zone
ENG	Engineering
FQDN	Fully Qualified Domain Name
GEO	Geostationary Equatorial Orbit
IP	Internet Protocol
IT	Information Technology
LEO	Low Earth Orbit
MEO	Medium Earth Orbit
NVD	National Vulnerability Database
OPS	Operations
OS	Operating System
OSSA	Open Stack Security Advisories
OSVDB	Open Source Vulnerability Database
OWASP	Open Web Application Security Project
PRC	Princeton
RHEL	Red Hat Enterprise Linux

SAMM	Software Assurance Maturity Model
SecSDLC	Secure Software Development Lifecycle
SIM	Simulation
SLES	SUSE Linux Enterprise Server
SOC	Space Operations Control
SP	Service Pack
SSH	Secure Shell
SSL	Secure Sockets Layer
SUMA	SUSE Manager
VAL	Validation
WebUI	Web User Interface

1. INTRODUCTION

Year after year new exploitations of known software vulnerabilities are making it to the news. These attacks are becoming so recurrent that the question is no longer if a vulnerability will be exploited, but when will it be exploited.

In this context, companies and organizations should put more effort than ever to try to stay on top of these exposures, keeping their systems updated and having clear procedures on vulnerability detection and remediation.

However, although many studies show improvements on the vulnerability management process of organizations, there are still many companies which still lack formal processes and procedures in place to systematically reduce vulnerabilities.

Moreover, among the organizations which already have these processes implemented, there is still room for improvement, as a gap remains in many cases, between the identification of vulnerable applications and the time when they get fixed. This gap gives attackers a window to launch malicious actions against them.

It is important to remark that most of the cyber-attacks could have been avoided with a timely update of the systems. This is the case of some of the most famous cyber-attacks in the recent years, such as WannaCry and NotPetya ransomwares in 2017, which exploited EternalBlue vulnerability, being a patch available already one year before.

This fact is supported by the numbers presented in Flexera Vulnerability Review 2020 [6]. This report shows that from the total of 13,319 new vulnerabilities discovered in 2019, 83.9% of them had a patch available on the day of disclosure. In addition, only 20 of those vulnerabilities were zero-day vulnerabilities, meaning that exploitation of vulnerabilities before its public disclosure remains a rare phenomenon.

Taking into consideration the figures above, vulnerabilities are still a big problem for the security of organizations, but also an issue that can be in a great extent solved if correct policies and processes are followed.

In this thesis, an overview of the challenges that vulnerabilities bring is presented, alongside some of the key practices to follow in order to minimize them.

The document is structured as follows:

In the first half of the thesis a complete overview of vulnerabilities through their lifecycle is presented, remarking the problems and risks involved and the possible solutions or key practices in each of the lifecycle's stages.

In the second half of the thesis, the theoretical concepts introduced in the first half are applied in a real case study, through an internship performed in the vulnerability management of systems of the Space Operations department of SES.

The main contributions of this thesis are:

- An overview of the main issues and key practices of vulnerability management.
- Implementation of key vulnerability management practices in a real-life scenario:
    - Through the configuration of a semi-automated patch management system.
    - Reducing the vulnerabilities present in the company's systems.

2. VULNERABILITY MANAGEMENT

Vulnerability management is the cyclical practice of identifying, classifying, remediating and mitigating vulnerabilities [7]. In this section, the concepts of vulnerability and the possible security mechanisms that can be implemented are introduced, to both avoid them and to solve them when they show up.

2.1 VULNERABILITY LIFECYCLE

Before jumping into the field of vulnerability management, it is important to define the concept of vulnerability and describe all its lifecycle from creation to elimination.

[17] defines a vulnerability as “a flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy”.

From its initial creation and discovery until its final elimination, vulnerabilities undergo several stages described in the vulnerability lifecycle in [8]. The following figure shows these different stages and the risks associated to each of them:

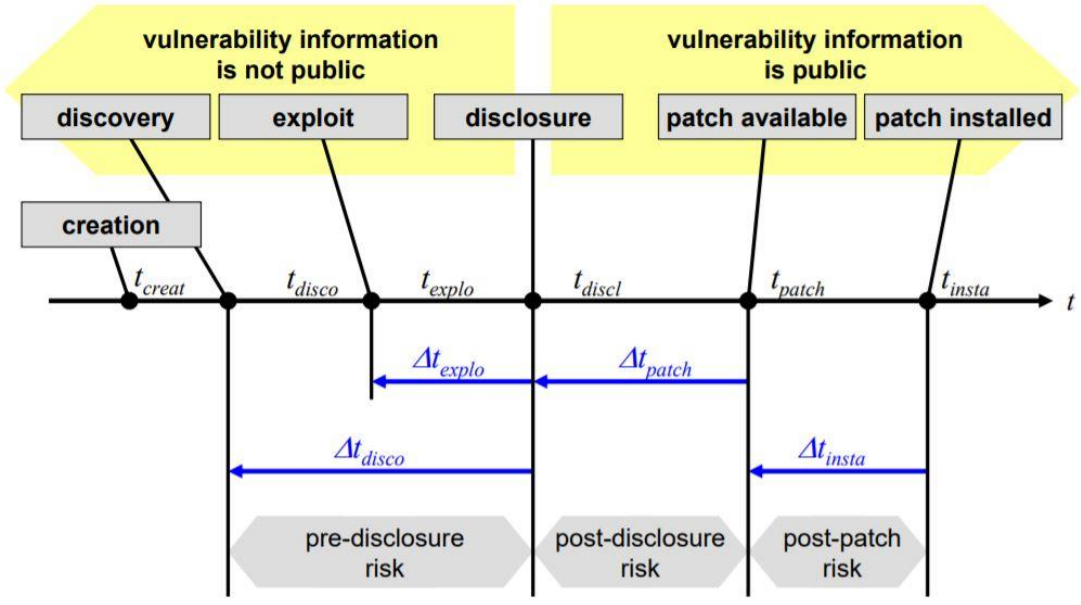

Figure 1: Vulnerability Lifecycle [8]

2.1.1 Events

Creation Time

This is the time when the mistake was done during software development. The exact date of this event is unknown, as the vulnerability has not been discovered yet. There could be an exception where a malicious actor creates a vulnerability on purpose, in this case the creation and the discovery of the vulnerability happen at the same time.

Discovery Time

This is the first time a vulnerability is found in a software or system. The vulnerability can be discovered by the developers or vendors themselves, by malicious actors or security researchers.

Exploit Time

This is the first time when an exploit that takes advantage of that vulnerability is available. From this moment of the lifecycle, the vulnerable software is exposed to a security risk, sometimes without the knowledge of the code owner.

Disclosure Time

This is the time when a vulnerability is publicly disclosed by a trusted and independent channel, after it has been evaluated by experts. Common channels for such disclosures include OSSA, NVD and OSVDB.

Patch Available Time

This is the time when a vendor releases a patch that fixes the vulnerability. This can happen both before or after the public disclosure time. Measures that could mitigate the vulnerability such as anti-virus tools or intrusion prevention systems are not considered patches, because they do not eliminate the root cause.

Patch Installation Time

This is the time when the user installs the patch, ending with the vulnerability and therefore its lifecycle. This date can vary a lot depending on the priority of the patch in terms of the risk it poses or the patching policy of an organization.

2.1.2 Risk Exposure Phases

Pre-Disclosure Phase

This phase happens between the discovery time and the disclosure time. In this period, only a limited number of actors have access to the vulnerability, this could be from cyber-criminals to security researchers and vendors who are already trying to find a patch to solve the vulnerability. Anyway, the risk in this phase is high, because malicious actors can damage an organization without it being aware that their systems are vulnerable.

Post-Disclosure Phase

This phase happens between disclosure time and patch time. In this period, the public is aware of the risk and the organizations can evaluate it to assess what actions to take, although the patch is not yet available. At this point, the risk is still high, because there is no fix to solve the root problem, however, organizations can take some measures which could lower the risk until the official patch is disclosed by the vendors.

Post-Patch Phase

This phase happens between the release of the patch and the installation in the system. This period is determined by the patch implementation speed of the organization and the risk is linked to this speed.

There are some important considerations to take into account from both vulnerability stage's and its associated risk's point of view.

First of all, the 6 events described above could happen in a different order than the sequence described. Sometimes, and this would be the ideal case, the patches are

released before the disclosure time, or even better, the vulnerabilities are discovered by security researchers instead of malicious actors. In the first case, the organizations are vulnerable in the pre-disclosure period only and in the second case, there is only risk in the post-patch phase or no risk at all if the patch is installed before its public disclosure.

Regarding exposure levels, it is important to note that both pre-disclosure phase and post-disclosure phase happen in periods where the organization or the software user has no influence, until the patch is released, they cannot solve the root cause of the vulnerability. However, they can try to get into their post-disclosure period faster, by keeping informed about the latest vulnerabilities affecting their systems and performing a risk assessment once in that stage.

Finally, the post-patch phase is totally under the influence of the affected organization and the time and risk level that they are exposed to depends on their vulnerability patching management processes, which will be further studied in this thesis.

2.2 SECURE SOFTWARE DEVELOPMENT LIFECYCLE (SECSDLC)

The best way to deal with a vulnerability is not to create it in the first place. For that it is really important to bring the security scope as early as the development process.

In [4] the importance of bringing security to the early stages of software development is stressed, to avoid the costly problem of dealing with vulnerabilities when the software is released. Through the study it is concluded that the later in the lifecycle is the security scope tackled, the higher is the cost of dealing with the vulnerabilities.

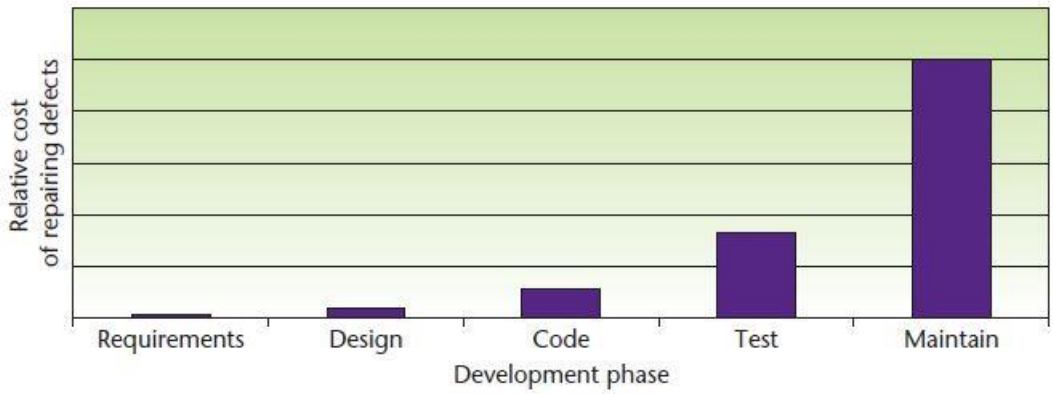

Figure 2: Relative cost of repairing software defects [4]

Many different models have been proposed to improve the security in the software development lifecycle, for instance BSIMM or OWASPs SAMM. One of the most popular processes is though the Microsoft Security Development Lifecycle [11]. In the next figure, the different security practices for each of the stages of the lifecycle are depicted.

Figure 3: SecSDLC practices

Training

The first security practice, even before starting the actual software development life cycle is to provide training in security concepts to the developers. Security is everyone's job and although not everyone needs to be an expert in the field some basic concepts such as understanding the attacker's perspective can contribute in building secure software.

Requirements

When defining the functional requirement of any software or system, it is really important to also define the security requirements. This requirement definition should be ideally done in the initial design and planning stages to minimize possible disruptions later in the

development process. Some of the recommended actions in this phase are to create quality gates or bug bars and to perform a security and privacy risk assessment.

Design

Through the design phase, developers will choose which security features such as cryptography, authentication or logging they rely on in order to implement a secure solution. However, a bad design choice can result in vulnerabilities. Therefore, it is of vital importance to analyze the attack surface and build a threat model for each particular system.

Implementation

When starting to implement the solution, the right choice of the tools to be used is really important from a security perspective. It is always essential to use approved tools with the latest versions installed and to be aware of and manage the risk of using third-party components. Moreover, performing a static analysis of the source code before compilation is also a recommended action to take in order to reduce the possible vulnerabilities.

Verification

Once the software is developed, verification or testing is needed also from a security point of view. Some of the recommended actions at this stage are to perform dynamic analysis of the code to discover bugs such as memory corruptions and fuzz testing to discover unexpected behaviors. It is also a good idea to review the attack surface at this point and perform some penetration testing to unveil coding and configuration errors which could generate vulnerabilities.

Release

Just before the software is released there are some essential actions to perform. These include to establish and test an incident response plan and to perform a final security review.

Response

This stage only happens if there is an incident with the developed system. Here the main action is to execute the incident response plan drawn in the previous step to address it.

The implementation of the practices described above in the SecSDLC can make a great contribution to reduce the number of vulnerabilities that have to be patched later. However, no matter how good the security approach in the development lifecycle is, sometimes vulnerabilities keep appearing and patches have to be released to deal with them.

2.3 VULNERABILITY DETECTION

After the development and deployment of the systems a periodical and constant effort should be made to detect vulnerabilities that could show up both because of the environment where they are deployed, or for design and coding mistakes that were not detected in the development phase.

In [15] some of the most common security testing techniques are presented, including network discovery, port and service identification, wireless scanning and vulnerability scanning. Although, there are multiple tools and solutions to specifically perform the different techniques, most commercial vulnerability scanners offer all these techniques as a unique product or service.

A vulnerability scanner is a tool used to identify vulnerabilities in a computer, network or application. Alike network and port identification services, it can identify hosts and their attributes in a network, but it goes further, interpreting those scanning results and giving possible vulnerabilities as an output.

Shortly, vulnerability scanners are able to:

Check the application usage and compliance with security policies. This can be done matching the information obtained through the scan with vulnerability databases.

Provide information on targets for penetration testing. Some vulnerability scanners can even offer additional tools to automate the testing to check that those vulnerabilities can be exploited.

Provide mitigation information on the discovered vulnerabilities. In this case also, some scanners can be linked with patch management tools which can automate the process of patching the vulnerabilities discovered by the scanner.

We can categorize scans depending on which systems they target and how they do it. This classification is summarized in the next table.

Scan type	Assistance/root access	Summary
External network	None	- Reflects capabilities of external attackers - Less precise, less vulnerabilities discovered
	Firewall assistance through port forwarding	- Increased insight into the network - Do not accurately reflect external attackers' capabilities
Internal network	None, user/admin credentials	- Reflects capabilities of malicious insider, or attacker with internal network access - Different vulnerabilities than with external scan discovered
Local host	None, user/root credentials	- Reflects capabilities of an attacker with different types of access to local hosts - Highest accuracy on vulnerabilities and misconfigurations

Table 1: Vulnerability Scan Categorization

External Network Scan

This type of scan is performed to analyse the network from an external point of view and it encompasses some of the techniques mentioned before, such as network and host discovery and analysis of port state and related vulnerabilities.

When performing an external scan, assessors must deal with the security perimeter of the network and other challenges such as the use of Network Address Translation and host-based firewalls. For an increased insight on the network, port forwarding can be enabled in specific IP addresses or local firewalls can be configured to allow the scan.

If those techniques are utilized more extensive scan results could be outputted. On the other hand, if no assistance is provided to the scanning, the process will more accurately reflect the capabilities of an external attacker.

Internal Network Scan

This type of scan can be performed to test the capabilities of a malicious insider or an external attacker which has already access to the internal network.

Different credential levels can be provided to the scanner to reflect different type of situations, which could uncover distinct vulnerabilities.

Local Host Scan

In this type of scan, normally the scanner is installed in each host. A higher level of vulnerabilities and misconfigurations can be discovered with this method. Depending on the type of credentials given to the scanner, different scenarios can be tested.

As described above, each type of scan can reflect different threat scenarios and can output different vulnerabilities. Therefore, it is really important to do all type of scans as often as possible to assure that all the vulnerabilities are discovered on a timely basis.

2.4 VULNERABILITY IDENTIFICATION AND SCORING

Once the vulnerabilities are discovered, they should be uniquely identified and classified taking into account the risk that they could pose on the systems. For this purpose, multiple standards have been developed by different associations. CVE [12] and CVSS [5] are two of the most popular standards.

2.4.1 Common Vulnerabilities and Exposures (CVE)

CVE is an open standard that gives globally unique identifiers to each different vulnerability. It is currently maintained by Mitre Corporation (US-based not-for-profit organization) and it was created in 1999 to deal with the problem of not having a common naming standard to identify vulnerabilities. At the moment, there are over 130.000 vulnerability entries registered in the database.

CVE identifiers consist of three parts: CVE prefix, year of application for the identifier and a unique number which is reset every year. An example of a well-known vulnerability is Heartbleed, a security vulnerability in the OpenSSL library, which is identified as described before as CVE-2014-0160.

2.4.2 Common Vulnerability Scoring System (CVSS)

CVSS is a system that provides a way to capture the principal characteristics of a vulnerability and produce a numerical score reflecting its severity [5]. It is currently maintained by the Forum of Incident Response and Security Teams and it was created in 2005 to deal with the problem of not having a structured way of rating the severity of known vulnerabilities [10].

In the current version, CVSS 3.1 (released in July 2019), the final result is a score and a vector calculated from three different metrics: base metrics, temporal metrics and environmental metrics. The final score is a number which can range between 0.0 and 10.0, lowest score meaning that there is no vulnerability and highest meaning extremely severe vulnerability.

Base metrics

The base metric is a common score which is defined when the vulnerability is first registered, and it normally stays unchanged, unless new ways to exploit it are discovered which could affect the severity. Base metrics are mandatory to get a CVSS score and they are completely inherent to the vulnerability itself (they do not depend on the time or environment where the vulnerability can be exploited).

Base metrics are divided into three groups:

Exploitability: Attack vector needed to exploit it, attack complexity, privileges required, user interaction needed.

Scope: Whether other components could be affected if the vulnerability is exploited.

Impact metric group: Confidentiality, Integrity and Availability impacts.

Each of the metric presented above, can have different predefined values. For example when assessing the attack vector, the possible options are Network (AV:N), Adjacent Network (AV:A), Local (AV:L) and Physical (AV:P). Depending on the choice the final score will be higher or lower (a vulnerability is considered less severe if it can only be exploited physically than if it can be exploited over the network).

Once all the metrics are defined a vector is formed with all the answers and a score is calculated following some formulas which give each attribute a weight. Online calculators are available to automate this process [13].

Following the example of vulnerability given in CVE section, HeartBleed vulnerability (CVE-2014-0160), gets the score: 7.5 (High) and the vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N.

Temporal metrics

This is an optional metric group which contains metrics that change over time. Those metrics are:

Exploit code maturity: How developed is the exploitation technique (only theoretical, autonomously exploitable…).

Remediation level: It describes the current patch status (not available, official fix…).

Report confidence: Credibility of the existence of the vulnerability and published technical details.

As this metrics can change over time, they are optional when calculating the CVSS score of a known vulnerability and it is used to modify the base score.

Environmental metrics

This one is also an optional metric group which should give different results in each organization. The metrics assessed here are the same as in the base metric, but they are used to customize the severity of a vulnerability for each organization or environment.

For example, the confidentiality impact is higher in a company dealing with personal data, so this will change the assessment of the confidentiality impact metric, making the overall CVSS score higher.

CVSS is a really important standard used when trying to solve vulnerabilities in the patch management phase, as it is a metric used to decide if the patch has to be implemented immediately or it can follow the standard patching calendar.

2.5 PATCH MANAGEMENT

The last step in the process of vulnerability management, once the vulnerability has been identified and classified, is to eliminate it applying a patch.

However, in order to be effective, it is essential to define a formal process which should be followed to overcome implementation barriers and to be able to perform it periodically in a structured way.

In the following lines, the issues that patch management usually faces and key practices for a correct implementation within an organization will be presented.

2.5.1 Implementation Barriers

A correct management of the security of the systems involves having an accurate knowledge of the specific applications, components and OSs running in them, as patches affecting them are published in a daily basis. An effective patch management procedure should take into account various factors and specific particularities of the systems involved and has to overcome some implementation barriers.

In [9] the authors identify what they call “Pain Points” or barriers that a correct patch management procedure should overcome in order to be effective:

How to detect that a new patch is available

A fast awareness of available patches is fundamental to reduce the exposure window. Generally, there are two ways to retrieve this information. The passive one, is to be subscribed to a notification channel where the vendor can directly inform about new patches for their products. The active one, is to access the dedicated web page where normally vendors publish the vulnerabilities of their products with their patches.

Which patch is applicable to which specific system

Not all the systems are affected by the same vulnerabilities, so it is essential to identify which of the available patches fit in each of the systems. Moreover, some patches could affect some specific functionality of a system and in some cases some patches should be waived until the issue is resolved.

Patches need to be tested in non-production environments

A patch cannot be applied in a production environment, because of the collateral effects or even the system failure that the patch could generate. Therefore, a testing environment which mirrors the functionality of the production environment should be patched first to avoid those issues.

How patches can be properly applied

The last mile of the process, the actual implementation of the patch, should be thorough and accurate, as a mistake could leave the system vulnerable while giving the administrator a false security thinking that the system is correctly patched and safe. The other factor to take into account is the downtime of the system to be patched. Sometimes, patching requires a downtime of the system which should be planned, especially when dealing with critical systems.

2.5.2 Key Practices

As stated before, patch management is a process where several aspects should be considered and with some barriers to overcome in order to be effective. There are many

publications such as [2] and [21] were some best patch management practices are presented. In the following lines a patching procedure will be presented, which includes some of the best industry practices:

Establish policies, procedures and responsibilities

Patch management should be a formal part of the organization's information security management system. The different policies and the procedures and who in the organization will be responsible should be clearly defined.

Some of the responsibilities in this area include vulnerability monitoring, alert response and asset tracking. The frequency of the vulnerability scans and the patching efforts should also be established beforehand, and ideally, the efficiency and effectiveness of the patching process should be monitored.

Maintain an inventory of all the assets of the IT infrastructure

Maintaining a complete and accurate inventory of the assets of the IT infrastructure of the organization is an essential step of a correct patch management process.

Each of the assets should be classified by type (physical, virtual, work station, etc.), location in the network (internal network, DMZ, etc.), the OS version and software that are running. In addition, having a record of the states of updates and the vulnerabilities associated is also strongly recommended. Finally, the staff responsible for each of the assets should be also identified, as they are going to be the ones affected if something goes wrong.

Monitor vulnerability alerts

The organizations should be aware for new vulnerabilities affecting their systems. For that purpose, they should be subscribed to vulnerability alert channels from vendors, security organizations or other third-party services.

Vulnerability scanning tools are really useful too in order to be aware of new vulnerabilities and will precisely highlight which systems and applications are suffering from a known vulnerability.

The use of the CVE notation will facilitate the gathering of the information from different sources and the identification of the issue concerning each of the systems.

Assess and respond to vulnerability alerts

Once the vulnerabilities and the affected systems have been identified, the priority of the vulnerability should be assessed. The most accurate way to do that is to use the CVSS scale, but other less accurate scales such as low, moderate, high, critical are also widely used.

The next step after assessing the risk level of the vulnerability, is to identify if there is any patch available and if it is applicable to the systems (there could be cases when a patch cannot be applied because it would affect some essential functionality of the system and it has to be waived until the problem is solved).

If a patch is directly not available yet, some prevention measures could be taken in the meanwhile. These could range from an increase of monitoring of vulnerable systems to a temporary shutdown. Of course, the application and severity of the prevention measures will depend on the risk and the impact of an exploit of the systems.

Prepare backups

The application of certain patches could have unexpected results in the systems. Therefore, it is essential to prepare a backup of the systems to be patched, to be able to rollback in case something goes wrong.

Test and evaluate patches

As stated before, patches could have unexpected results when applied in the systems. This could be a big problem if they have to be installed in production environments or critical systems. The best practice to solve this issue is to have a replica of the production environment and test the deployment of the patch first there.

Once the patch is installed in the test environment an evaluation should be performed to see if any of the components of the system has been affected. If the system is functioning correctly, then the risk of implementing it in the production environment is much lower.

Install patches

Compromise between the urgency of implementation of patches and the minimization of the business interruption is a key aspect to consider for many organizations. Taking this into account, patches should be applied, if possible, out of business or production hours.

This issue is even a bigger problem for systems which have to be up all the time (in some sectors systems could be required to be working more than 99.999% of the time (less than 6 minutes down time per year for any reason) [21]. This means, that the patches should be precisely scheduled, or backup systems should be used during the patching time.

Assess the patch deployment

Once the patches have been installed, it is the time to evaluate the impact they had in the systems. First, logs should be reviewed to check if there has been any problem during the process and if all the patches have been deployed correctly. If there is any issue it should be addressed and in the worst case, the system can be rolled back to its last working state.

Finally, the whole patching process should be regularly evaluated to measure its effectivity and introduce improvements in next iterations.

The implementation of an effective patch management procedure can be really tedious and time consuming, especially for organizations that have to deal with a large number of systems. Luckily for people in charge of this labour, there are many tools in the market which can help automate the process.

These tools can be used to implement most of the key practices introduced before, such as keeping a detailed inventory of all the IT assets, monitoring vulnerability alerts and matching them with the adequate systems, assessing the risk of the vulnerabilities with different scales and finally installing the patches.

3. VULNERABILITY MANAGEMENT IN THE SPACE OPERATIONS DEPARTMENT IN SES

In this section, my experience as a cyber security intern in SES in its headquarters in Betzdorf, Luxembourg, will be presented.

During the 6 months of duration of the internship in SES's Space Operations department and under the guidance of the Space Operations Chief Security Engineer, many of the vulnerability management concepts presented before were applied in a real-world environment.

3.1 COMPANY AND BUSINESS SECTOR

3.1.1 Business sector

The satellite market is an important part of many economies for the development of infrastructure for commercial companies, government agencies, telecom and space industry. Its market size was valued at $126.5 billion in 2018 and is expected to reach $144.5 billion by 2026 [1].

Artificial satellites are objects placed in the earth's orbit to act as cell towers in the sky to transmit data between different locations of the planet. Satellites can vary between orbit, frequency and mission, offering different purposes and services such as telecommunication, navigation and military.

The business is segmented based on the type of satellite (size of satellite normally depending on the type of orbit, GEO, MEO or LEO), end user (commercial, civil, government) and application (communication, research, earth observation…).

Some of the market's key players include Intelsat, SES, Eutelsat and other smaller companies.

3.1.2 SES

SES is a satellite and terrestrial telecommunications network provider, which supplies data and video connectivity worldwide to different industries, institutions and governments [16].

SES's network, formed by 54 GEO and 20 MEO satellites, covers the 99% of the world's population with more than 367 million TV home served in 2019. In 2021 a new GEO satellite and 7 new MEO satellites are planned to be launched.

In 2019 it reported a revenue of almost 2 billion dollars and the company's stock list is listed on the Luxembourg Stock Exchange and Euronext Paris.

SES divides its business in 2 units, SES Video and SES Networks, counting for 61% and 39% of the total revenue respectively. SES Video delivers both video distribution and services in different platforms and formats for a total of more than 1 billion people globally. SES Networks provides connectivity services to various industries and governments around the world.

With its headquarters in Betzdorf (Luxembourg), it is a highly international company with 2185 employees from 81 nationalities in 41 working locations worldwide.

3.1.3 Space Operations department

Space Operations is one of the 13 departments that form SES's Technology group. Its main purposes are:

- Lead the 24/7 operation of the GEO/MEO satellite fleet ensuring all planned activities are executed in a timely and safe manner.
- Perform the orbital control and co-location of the SES fleet.
- Develop all the satellite Ground Control Software (GCS) and tools needed to operate fleet.
- Continuously improve the safety and efficiency of satellite operations by automating operations with appropriate situational awareness systems and controller's training/certification.

Spacecraft Control Systems Security is one of the sub-departments of Space Operations, in charge of ensuring the security of the department's systems with a close partnership with SES Technology group's IT Security department.

As a Cyber Security intern, I assisted the chief of Spacecraft Control Systems Security to achieve this department's goals.

3.2 INTERNSHIP GOALS

The main goal of the internship is to incorporate as a Cyber Security student in the Space Operations department of SES, putting into practice the skills and competences acquired during the master's degree.

Two main technical objectives were established in the beginning of the internship:

- Set up a semi-automated patch management system using SUSE Manager software, in order to register and maintain Space Operation's department's systems in it's different network enclaves.
- Contribute to the vulnerability mitigation of the department's systems through all the vulnerability lifecycle, from the discovery using Nexpose-Rapid7 vulnerability scanner, to its mitigation using SUSE Manager for the patching process.

3.3 DEVELOPMENT

Through the completion of this internship several tasks have been carried out in order to set up a patch management system to support the vulnerability management of the department. In the following lines, the processes involved in the development of the tasks and the main tools utilized will be described, along the challenges that have been faced during this development.

3.3.1 SUSE Manager setup

SUSE Manager is the main tool which has been used throughout all the internship to set up a centralized semi-automated patch management system to manage and deploy patches to the different systems of the department.

SUSE Manager is an open source infrastructure management solution for software defined infrastructures. It is a useful tool to regain the control of the IT assets of a company and to

manage Linux systems across different hardware architectures, hypervisors, containers and IoT and cloud platforms [19].

The solution allows both automation of provisioning, patching and configuration of Linux servers and IoT devices and monitoring, tracking, auditing and reporting of the systems across different environments (development, test, production…). All this happens while being able to ensure compliance with internal and external security policies.

The typical architecture of a SUSE Manager implementation is shown in the following figure.

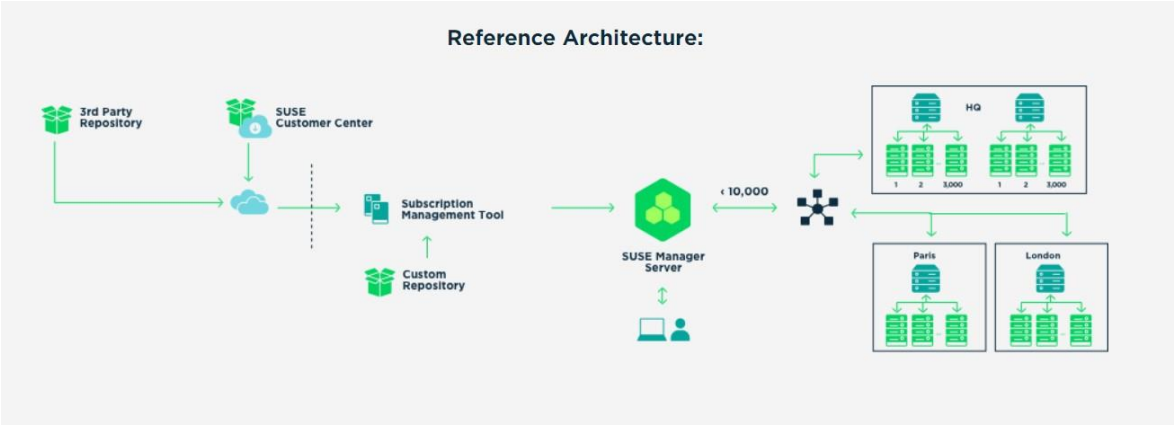

Figure 4: SUMA Reference Architecture [19]

The core of the architecture is SUSE Manager Server, which can be accessed both via user and command line interface. Connected to it there can be a few proxy servers each of them normally grouping several clients belonging to different locations or business units. Hosts can also be connected directly to SUSE Manager server without need of deploying a proxy server between them.

SUSE Manager server is at the same time also connected to a Subscription Management Tool which manages the different repositories and channels to later feed SUSE Manager with the adequate packages and patches for each system. The repositories can be custom repositories created by the own organizations, official repositories from SUSE Customer Center or repositories from 3rd party organizations. All of them end up in the Subscription Management Tool for the later organization and delivery of content to the server and finally the hosts.

The key functionalities of SUSE Manager along the developments made making use of them will be presented in the following lines:

Asset registration and management

SUMA allows a fast way to bootstrap, which means provisioning and configuration of new hosts both using the WebUI and the CLI.

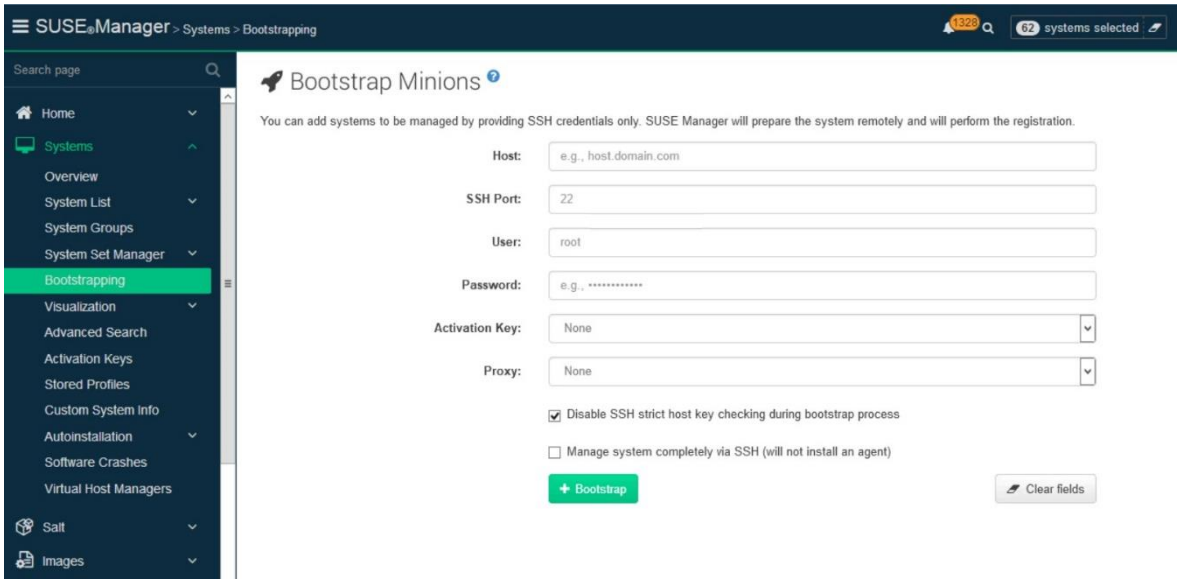

Figure 5: Bootstrapping in the WebUI

The WebUI offers the most intuitive way to register new hosts into SUMA. Filling in the information required shown in Figure 5, SUMA automatically contacts the host to be bootstrapped and the configuration to be deployed is indicated introducing an activation key.

However, this way to register hosts only works for systems which support Salt. Moreover, this system is not suitable for a bulk registration of hosts running the same configuration, as it would be time consuming to manually introduce this data via the WebUI. For this case, the best solution is to run a script through the CLI.

To bootstrap using the CLI, a bootstrap script needs to be generated and configured to apply the same configuration to a group of hosts in bulk. Then, the hosts should be contacted via SSH connection.

Once the bootstrap has been executed successfully the hosts will appear in the system list of the WebUI.

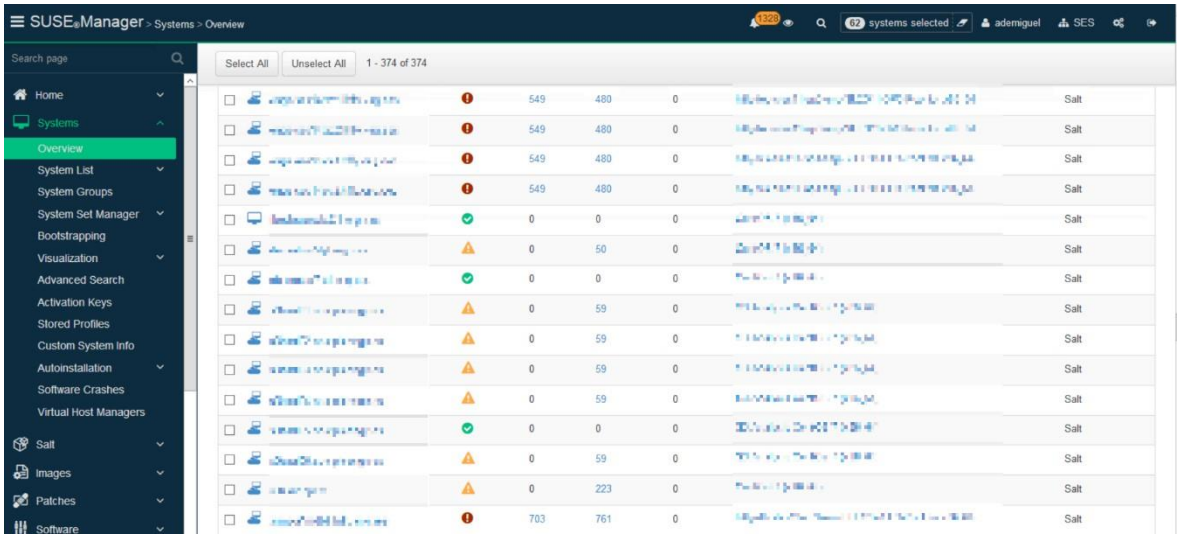

Figure 6: SUMA System Overview (system names blurred for confidentiality)

In the system list, we can see the list of the hosts which are registered to SUMA. The first column indicates the FQDN of the host. The second, the patching status, with green meaning system is up to date, orange patches available and red critical updates available. The next columns represent the patches and packages available for that system subsequently. Then, the number of available configuration files, the name of the software channel to which the system is installed and the system type (if the system supports Salt or not) are indicated.

During the internship several hosts have been gradually registered to SUMA in order to start managing them in a centralized fashion.

Software Channels

All the hosts registered to SUMA are connected to a base software channel which provides them with the adequate patches and packages. SUSE provides its own base software channels for SLES OS and its different Service Packs.

However, the Space Operation department has hosts running different OS such as CentOS and RHEL, SUSE Manager 4.0 does not directly provide packages for those OSs. Therefore, custom channels synced to the correct repositories have been created to be able to provide the suitable patches for those hosts.

Moreover, custom channels have been also created for each environment of systems and system type running SLES. Those channels are clones of the official channels provided by SUSE; however, they are customized to serve the specific needs of each system group, following a patch management lifecycle process.

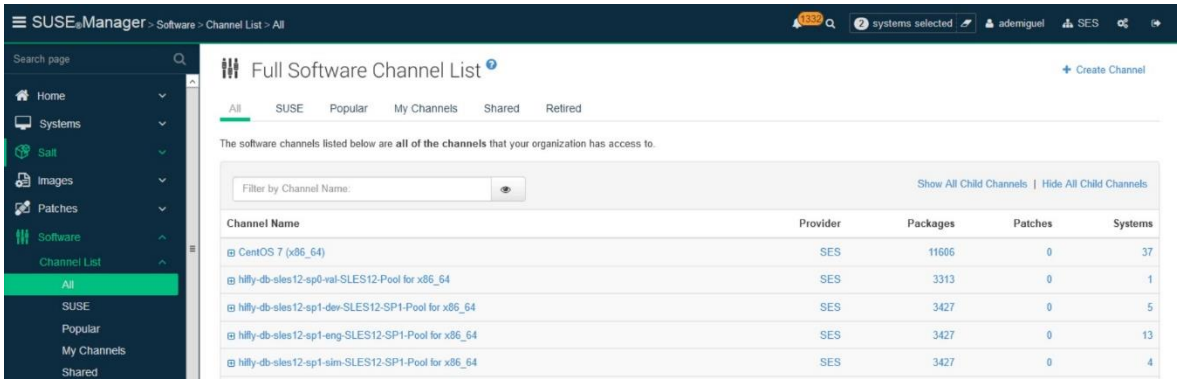

Figure 7: SUMA Software Channels

Patch Management Lifecycle

Content Lifecycle Management is a feature that SUMA offers to customize and test packages before updating production systems [20]. This is a really useful tool for the systems in the Space Operations department, as satellite control systems are critical, and all the patching and package updates should be first deployed and tested first in non-production systems.

With the aid of this feature a Patch Management Lifecycle process has been followed to make sure that patches are applied safely in each of the departments systems. The following figure depicts this process:

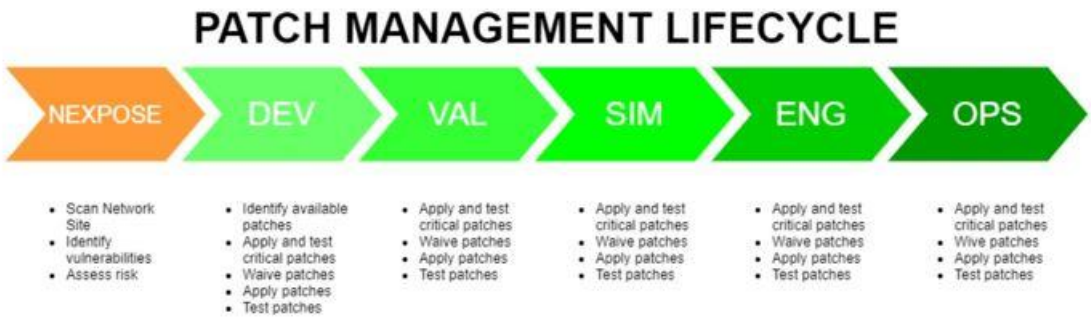

Figure 8: Patch Management Lifecycle

The first step consists in performing a scan of all the assets of the network with Nexpose. This results on a list with all the systems and vulnerabilities affecting them. Each vulnerability has a CVSS risk score associated and possible solutions to solve it.

Once the list of vulnerabilities is known, the software channels in SUMA which are connected to the vendor's repositories are checked to see if patches to the discovered vulnerabilities are available. At this moment a process that is repeated in each of the department's environments (DEV, VAL, SIM, ENG and OPS) is initiated.

First of all, if there is any vulnerability which has to be solved immediately, the patch that solves it is applied and tested in each of the environment as fast as possible. This is achieved in SUMA, setting up a filter which allows a single patch or package to be promoted to each higher environment instantly.

Then, the available patches should be checked to see if all of them can be applied in the systems without causing problems in them. If there are patches that cannot be applied at that moment, they are waived using again a filter that denies the promotion of those patches in SUMA. As soon as the condition to apply the waived patches is fulfilled, the filter should be disabled, and the patches applied.

Once the critical and waived patches are filtered, the other available patches can be applied in the environment. After the patching process has been troubleshooted, the system owner tests that everything is working correctly, and those patches can be promoted to the next environment.

This process is repeated in each of the environment until OPS (the production environment) is patched. The cycle should be started from the beginning as many times as it is necessary (or enforced by internal or external security policies) to keep the systems up to date and free of vulnerabilities.

Proxy Setup

SUSE Manager Server acts as a centralized node which keeps control of multiple systems on a network. However, in case the number of systems under SUMA's control increases or in case that the systems to be managed lie in different network enclaves, proxy servers should be used to act as local SUMA servers for those systems. In SES's Space Operations

department's case, systems are divided in different networks depending on their purpose and location.

In order to manage all the systems in the different networks centrally, 3 proxies are planned to be set up. The network architecture with the different department networks and the proxies is shown in the next figure:

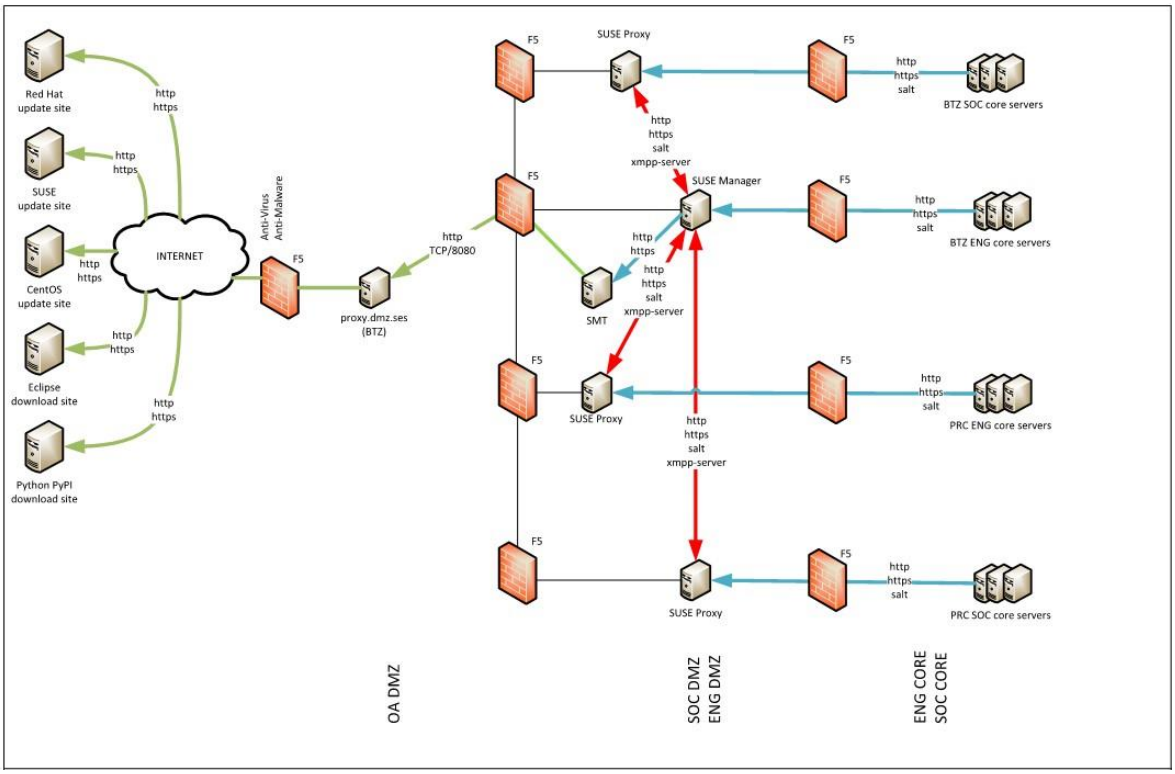

Figure 9: SUMA network architecture in SES

The main SUMA server is located in ENG DMZ of Betzdorf location and it directly manages the systems inside this network enclave. There are 3 proxy servers connected to it, managing systems of the other network enclaves (Betzdorf SOC, Princeton ENG and Princeton SOC). Finally, SUMA server is connected to the DMZ Proxy which acts as the gateway to the internet, to access the repositories containing patches and new packages to be installed by SUMA on the end point systems in each of the network enclaves.

3.3.2 Nexpose

Nexpose is the tool which have been used during the graduation project to identify the vulnerabilities affecting SES's Space Operations department's systems vulnerabilities.

Nexpose is a vulnerability management software that allows reducing threat exposure, allowing a real time assessment and response of changes in the monitored environment, prioritizing risk across vulnerabilities, configurations and controls [14].

The results of the vulnerability scanner are the starting point of the patch management process, as shown in Figure 8.

Nexpose is configured to perform weekly scans of the different network enclaves which contain the department's systems. This way, a weekly snapshot of the vulnerability state of the systems is available and patching can be planned accordingly.

Other interesting features of Nexpose are the risk score and number of vulnerabilities of the assets over time. With these two parameters a nice visual representation of the success of the patching efforts can be studied. In the next figure, the evolution of the vulnerabilities of one of the systems can be observed over one year, until it finally gets patched.

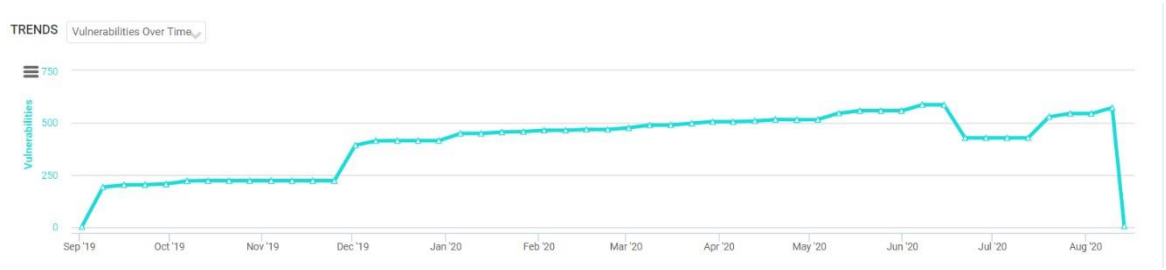

Figure 10: Example of vulnerabilities over time on a Hifly host

We can observe vulnerabilities go increasing over time as new ways of exploiting the system appear. The vulnerability count falls once all the patches are applied and only the configuration vulnerabilities remain. Those vulnerabilities should be studied case by case taking into account the operational needs of the systems.

3.3.3 Patching process

During the internship, once the main SUMA server was set up, some patching efforts have been initiated to reduce vulnerabilities affecting Hifly hosts of DEV and VAL environments.

Following the schema presented in Figure 8, the patching process has first been initiated in DEV environment and once a successful result has been achieved, it has been extended to VAL environment.

This process starts with the identification of the vulnerabilities present in Nexpose analysis and the patches available for each of the systems in SUMA. Once the systems and the patches to apply have been identified, a snapshot of the systems should be taken, and a disruption window should be programmed by the system administrator.

During this time frame the software engineers and system operators are aware that they could suffer some disruption or that the systems could go down for a short time if the type of patch to be applied requires a reboot of the system. Normally, the windows are scheduled during the night and for the first few hours of the day, to troubleshoot the systems in case some patches fail.

The morning after the patches have been applied, during the maintenance window yet, the correct application of the patches is checked, and the failure cases are analysed checking the logs. If the problem can be solved in time, the patch is applied and if any problem is found the system can be rolled backed using a snapshot of the system taken just before applying the patches.

Once the patches are correctly applied for the systems of one environment and have been tested, the process can be applied to the next environments in an iterative way.

3.3.4 OS Lifecycle

Software usually get security support in form of patches or updates which allow to correct vulnerabilities that may be discovered. However, products are constantly evolving over time and vendors only provide support for their software during a window of time or lifecycle.

This is a particularly important issue when dealing with the vulnerability management of machines running different type of OSs like the ones managed in SUMA.

Vendors release new versions of their OSs periodically and they support them only for a limited time frame. Once the general support period has ended the products stops receiving security patches and other enhancements, so the machines become exposed to the latest vulnerabilities. Often, vendors extend the support window with Long Term Services and Supports subscriptions which allow companies to keep working with old versions of the product until they are prepared to migrate their systems to the newest version.

One of the tasks carried out during the internship has been to identify the different OS-SP versions where the systems managed by SUMA running in order to ensure that they are receiving the latest security updates.

Information about the dates concerning the support windows can be found in the vendor's website for each of the different running OSs. In our case, the dates concerning SLES OS and CentOS are shown in Figure 11 and Figure 12.

End of Lifetime (EOL) Dates			
	CentOS-6	CentOS-7	CentOS-8
Full Updates[1]	May 10th, 2017	Q4 2020	May 2024
Maintenance Updates[2]	November 30th, 2020	June 30th, 2024	May 31st, 2029

Figure 11: CentOS Lifecycle [3]

Service Pack Release	FCS Date	General Ends	LTSS Ends
SUSE Linux Enterprise Server 11	24 Mar 2009	31 Dec 2010	N/A
SUSE Linux Enterprise Server 11 SP1	02 Jun 2010	31 Aug 2012	30 Aug 2015
SUSE Linux Enterprise Server 11 SP2	29 Feb 2012	31 Jan 2014	30 Jan 2017
SUSE Linux Enterprise Server 11 SP3	01 Jul 2013	31 Jan 2016	30 Jan 2019
SUSE Linux Enterprise Server 11 SP4	15 Jul 2015	31 Mar 2019	31 Mar 2022
Service Pack Release	**FCS Date**	**General Ends**	**LTSS Ends**
SUSE Linux Enterprise Server 12	27 Oct 2014	30 June 2016	01 July 2019
SUSE Linux Enterprise Server 12 SP1	15 Dec 2015	31 May 2017	31 May 2020
SUSE Linux Enterprise Server 12 SP2	08 Nov 2016	31 Mar 2018	31 Mar 2021
SUSE Linux Enterprise Server 12 SP3	07 Sep 2017	30 Jun 2019	30 Jun 2022
SUSE Linux Enterprise Server 12 SP4	12 Dec 2018	30 Jun 2020	30 Jun 2023
SUSE Linux Enterprise Server 12 SP5	09 Dec 2019	31 Oct 2024	31 Oct 2027

Figure 12: SLES Lifecycle [18]

Once the different running OS-SP versions have been identified a report has been given to the management, for them to make a decision for the following years.

For systems running CentOS7 there is no necessity to take action in a short term, as the OS will be receiving maintenance updates until June 30th, 2024.

However, for systems running various SPs of SLES 11 or SLES 12 some actions should be taken to ensure they receive the latest security updates. Two different alternatives have been presented:

Upgrade SLES 11 systems to SLES 11 SP4 and acquire LTSS for SLES 11. Upgrade SLES 12 systems to SLES 12 SP5.

This option would involve the lowest effort, as upgrades of the same OS version can be automated using SUMA. However, this would be just a patch until the systems are migrated to a newer OS version and acquiring LTSS subscriptions incurs extra costs.

Upgrade all systems to SLES 12 SP5

This option would involve greater effort, as upgrading systems from SLES 11 to SLES 12 involves an offline installation of the new OS version and can't be automated through SUMA. However, this would not incur any extra cost, as the subscriptions for SLES 12 already exist.

After considering the two alternatives, the second one was chosen by the management. Therefore, a gradual upgrade of all the systems which need it has been planned, following the same schema presented in the patch management lifecycle process (gradual upgrade from least operational environment to most operational environment).

3.3.5 Challenges

During the internship, several challenges have been faced on the way to achieve the desired results. In the following lines, some of these problems will be presented along with the decisions or actions taken in order to overcome them.

Network enclave interconnection

Following a secure network architecture, systems that belong to Space Operations department are deployed in different network enclaves with different type of access both to the internet and other systems in the company. While such an architecture makes a lot of

sense from a security perspective, it makes more complicated to deploy a system to centrally manage and keep up to date all the hosts in the network.

SUSE Manager is an excellent solution for patch deployment by setting up a centralized server to store the packages and configurations downloaded from the vendor's repositories. This way, the department's systems can simply fetch the adequate packages and patches from the server without having to connect to the repositories on the internet.

However, as stated before, systems are isolated in different enclaves and are not allowed to contact the central SUMA server. To overcome that problem, 3 different proxies were planned to be set up, to act as middlemen between the systems on those networks and the central SUMA server.

Moreover, new firewall rules had to be enforced to allow some services running both in SUMA central server and the proxies communicate. This way, the goal of having a centralized semi-automated management of the systems is achieved, while keeping the enclave interconnection controlled and to its minimum.

Patch deployment

The main goal of this graduation project is to configure a system that facilitates the patching process of the departments systems automating the process as much as possible. This is achieved with the use of SUMA, to install patches in bulk to systems which are grouped by environment and function.

However, when a bulk patch process is scheduled, it normally does not succeed in all the systems and many of them remain unpatched. There are multiple reasons which could cause these fails.

The first reason could be that SUMA server cannot reach the client host. This SUMA deployment, is configured to use "Salt" open-source configuration management software, to connect to most of the hosts. It could happen that when the server is trying to push the new patches to the clients, it does not receive a response, consequently the patch deployment will fail and a "minion seems to be down" error will be prompted. Normally it is just a synchronization error, or a problem on the client side and can be solved rebooting it or restarting the Salt service.

Another reason for an error in the patching process, could be a conflict between the patches or packages trying to get installed or updated and the ones already present in the machine. Normally, when there is a conflict with a patch in one machine, all the scheduled patches for that machine will fail, this can be a problem specially when many patches have to be installed in a short maintenance window.

In this case, the best option is to try to identify first which patches may fail, with the information acquired patching previous environments. If the patch still fails, logs should be analysed to find what is causing the conflict. Normally, as many hosts are running the same programs and configurations, it is enough to troubleshoot one host and to apply the correction in bulk to the other systems.

3.4 CONTRIBUTIONS

Throughout the 6 months spent working as a Cyber Security intern in the Space Operations department of SES, several task and activities have been carried out in order to achieve the goals set up in the beginning of it. The results of this efforts will be presented in the following lines.

Configuration of SUSE Manager

With the configuration of SUSE Manager Server and SUSE Manager Proxy in the Betzdorf ENG and Betzdorf OPS network enclaves of the department, the backbone for the automated patch management of all the devices of those networks have been installed.

Registration of systems into SUSE Manager

Hundreds of systems have been registered and configured to be managed centrally using SUMA.

Patch Management Lifecycle

A patch management lifecycle has been designed and applied to manage the gradual patching of the different systems of Hifly fleet.

System OS Lifecycle planification

The different OS versions run by the systems in the department have been analysed and an upgrade plan has been sketched to ensure the security support of the systems for the next years.

Vulnerability reduction

Vulnerabilities in Hifly DEV and VAL environments have been reduced through the upgrade of its database systems to the newest supported OS and through the application of the latest available patches.

3.5 FUTURE WORK

Vulnerability management is an effort that has to be applied constantly. Therefore, there are many things that the Space Operations department could do to improve the management of the security of their systems. In the following lines some necessary future task and recommendations are presented.

Deployment of proxies in Princeton ENG and SOC enclaves

Once SUMA is configured in Betzdorf ENG and SOC network enclaves, the configuration of the proxy in BTZ SOC should be copied into PRC ENG and PRC SOC enclaves, to be able to centralize and automate the patch management of the whole Space Operations departments systems.

OS upgrade of systems

All the hosts should be upgraded to the newest supported available version to ensure the reception of the latest security patches.

Patch Management schedule

Once the first full patching cycle is ended, a periodic patching schedule, involving the different system's environments should be planned. This way, following the patch management lifecycle presented before, the window of exposure to vulnerabilities could be controlled and reduced.

4. CONCLUSION

Vulnerability management is an essential process in the cyber security of any company or organization. The ever growing threat of cyber criminals taking advantage of unpatched vulnerabilities emphasizes the need of an effective procedure to administrate those exposures.

Moreover, the fact that most exploited vulnerabilities have already a patch available at the time of the attack, is an undeniable reason to keep all the systems up to date.

Throughout this thesis report, the problematic of a correct vulnerability management has been addressed through all the phases of its lifecycle, presenting the challenges involved and some of the best practices to deal with them in an effective way.

The practical application of some of the recommendations presented in the first part of the report in a global organization like SES, has proved the feasibility of applying the policies described in the literature.

However, it has also emphasized the complexity of managing the vulnerabilities in environments with hundreds or thousands of systems. Furthermore, if the systems belong to a production environment another extra difficulty is added to the process.

In conclusion, managing vulnerabilities in an organizational environment is a fundamental effort that if applied following an adequate procedure will effectively reduce the exposure to cyber-attacks.

5. REFERENCES

1. Allied Market Research. (2020, August 25). *Satellite Market*. Retrieved from https://www.alliedmarketresearch.com/satellite-market
2. Brykczynski, B., & Small, R. (2003). Reducing internet-based intrusions: Effective security patch management. *IEEE software*, 50-57.
3. CentOS. (2020, 08 28). Retrieved from CentOS Product Specifications: https://wiki.centos.org/About/Product
4. Diamant, J. (2011). Resilient security architecture: A complementary approach to reducing vulnerabilities. *IEEE Security & Privacy 9.4*, 80-84.
5. FIRST. (2020, August 25). *Common Vulnerability Scoring System*. Retrieved from https://www.first.org/cvss/
6. Flexera. (2020). *Flexera Vulnerability Review 2020.*
7. Foreman, P. (2019). *Vulnerability management.* CRC Press.
8. Frei, S. (2009). *Security Econometrics The Dynamics of (In)Security.* Zurich, Switzerland: ETH Zurich.
9. Gauci, A., Michelin, S., & Salles, M. (2017). Addressing the challenge of cyber security maintenance through patch management. *CIRED-Open Access Proceedings Journal*, 2599-2601.
10. Infosec Handbook. (2020, August 25). *CVSS, CVE, CWE, CAPEC – common standards security professionals should know*. Retrieved from https://infosec-handbook.eu/blog/cvss-cve-cwe-capec/
11. Microsoft. (2020, August 25). *Microsoft Security Development Lifecycle*. Retrieved from https://www.microsoft.com/en-us/securityengineering/sdl
12. MITRE Corporation. (2020, August 25). *About CVE*. Retrieved from https://cve.mitre.org/about/index.html

13. National Institute of Standards and Technology. (2020, August 25). *Common Vulnerability Scoring System Calculator*. Retrieved from https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator

14. Rapid7. (2020, August 25). *Nexpose*. Retrieved from https://www.rapid7.com/products/nexpose/

15. Scarfone, K., Souppaya, M., Cody, A., & Orebaugh, A. (2008). *NIST special publication 800-115: Technical guide to information security testing and assessment.* National Institute of Standards and Technology.

16. SES. (2020, August 25). *About us: SES*. Retrieved from https://www.ses.com/about-us

17. Shirey, R. (2007). *Internet Security Glossary, Version 2, RFC 4949 (Informational).* IETF.

18. SUSE. (2020, 08 28). Retrieved from Product Support Lifecycle: https://www.suse.com/lifecycle/

19. SUSE. (2020, August 25). *SUSE Manager*. Retrieved from https://www.suse.com/products/suse-manager/

20. SUSE. (2020, August 25). *SUSE Manager 4 Content Lifecycle Management Deep Dive*. Retrieved from https://www.suse.com/c/suse-manager-4-content-lifecycle-management-deep-dive/

21. Tom, S., Christiansen, D., & Berrett, D. (2008). *Recommended practice for patch management of control systems.* US Department of Homeland Security.