Implementing Patch Management Process

Pasi Hassani

Bachelor’s thesis
May 2020
School of Technology
Degree Programme in Information and Communication Technology

jamk.fi

Description

Author Hassani, Pasi	Type of publication Bachelor's thesis	Date May 2020
		Language of publication: English
	Number of pages 55	Permission for web publication: x

Title of publication

Implementing Patch Management Process

Degree programme
Information and Communication Technology

Supervisor(s)
Sampo Kotikoski, Jari Hautamäki

Assigned by
Landis+Gyr

Abstract

Landis+Gyr needed a patch management process to simplify and manage the patching more efficiently.

Different publications and methods created by several different professionals and organizations were utilized. The task was to implement a patch management process which suits Landis+Gyr's organizational requirements while the objective was to bring more security to the field of energy management through patch management and vulnerability assessments.

The theory and basis for the new patch management process was composed by utilizing some methods and steps from the prior process and complementing them with the new methods and steps gathered by researching various sources. New tools for assessing vulnerabilities are introduced as well. The practical implementation of the process was carried out with Oracle products due to their complexity and variety of versions.

Implementing the new process while recycling the existing methods allowed for a less intense approach to adopting the new process. As a result, the requirements placed for the new process were fulfilled. The process was simplified with clear steps and rules to follow while the theory clarifies why things work the way they do.

There is no such thing as a perfect patch management process. It is a process that requires constant maintenance and validation to keep it relevant. This work provides a solid foundation for the process which can be further developed in the future with new applications, better solutions and methods.

Keywords/tags (subjects)
Patching, Testing, Cyber Security, Energy Sector, Vulnerabilities

jamk.fi

Kuvailulehti

Tekijä Hassani, Pasi	Julkaisun laji Opinnäytetyö, AMK	Päivämäärä Toukokuu 2020
		Julkaisun kieli: Englanti
	Sivumäärä 55	Verkkojulkaisulupa myönnetty: x

Työn nimi

Implementing Patch Management Process

Tutkinto-ohjelma
Tieto- ja Viestintätekniikka

Työn ohjaajat(t)
Sampo Kotikoski, Jari Hautamäki

Toimeksiantaja
Landis+Gyr

Tiivistelmä

Landis+Gyrillä oli tarve korjauspäivitysten hallintaprosessille (engl. patch management process) jolla voidaan yksinkertaistaa sekä toteuttaa korjauspäivitysten hallinta tehokkaammin.

Tehtävänä oli implementoida korjauspäivitysten hallintaprosessi, joka noudattaa Landis+Gyrin vaatimuksia. Tavoite oli edistää energiasektorin kyberturvallisuutta korjauspäivitysten hallinnan sekä haavoittuvuusarvioinnin keinoin.

Korjauspäivitysten hallintaprosessin teoria ja pohja koostettiin hyödyntämällä olemassa olevan prosessin hyväksy koettuja metodeja ja vaiheita samalla kun niitä täydennettiin uusilla metodeilla sekä vaiheilla, jotka kerättiin tutkimalla laaja-alaisesti eri lähteitä. Työssä esitetään uusia työkaluja haavoittuvuusarviointia varten. Käytännön toteutus prosessista tehtiin Oraclen tuotteille niiden kompleksisuuden sekä eri versioiden moninaisuuden takia.

Uuden prosessin implementointi kierrättäen olemassa olevia metodeja mahdollistaa vähemmän intensiivisen lähestymisen uuden prosessin adoptointiin. Vaatimukset, jotka uudelle prosessille asetettiin, saavutettiin. Prosessi yksinkertaistettiin ja vaiheet sekä säännöt, joita tulee noudattaa ovat selkeät.

Täydellistä korjauspäivitysten hallintaprosessia ei ole olemassa. Se on prosessi, joka vaatii jatkuvaa ylläpitoa sekä validointia, jotta se pysyy ajan tasalla. Työ tarjoaa vahvan pohjan prosessille, jota voidaan jatkokehittää tulevaisuudessa uusilla applikaatioilla, paremmilla ratkaisuilla sekä metodeilla.

Avainsanat (asiasanat)
Korjauspäivitys, Testaus, Kyberturvallisuus, Energiasektori, Haavoittuvuus

Contents

Figures

Tables

Terminology

AMM	Advanced Meter Management
API	Application Interface
CPU	Critical Patch Update
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
DB	Database
DBSAT	Database Security Assessment Tool
EE	Enterprise Edition
EMEA	Europe, the Middle East and Africa
GPL	General Public License
HES	Head End System
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
JDK	Java Development Kit
JRE	Java Runtime Environment
JVM	Java Virtual Machine
LDAP	Lightweight Directory Access Protocol
ME	Micro Edition
NoSQL	Non-SQL
OS	Operating System
OWASP	Open Web Application Security Project
RDBMS	Relational Database Management System
R&D	Research and Development
SCADA	Supervisory Control And Data Acquisition
SE	Standard Edition
SIG	Special Interest Group
SIT	Solution Integration Testing
SQL	Structured Query Language
XE	Expression Edition
XML	Extensible Markup Language

1 Introduction

1.1 Assigner

The thesis was assigned by Landis+Gyr, Finland. Landis+Gyr is a multinational organization that designs and manufactures smart meters and develops the system software for them. Landis+Gyr is a globally recognized leader in the field of energy management and has operations in over 30 countries and in five different continents and is headquartered at Zug, Switzerland. (About Landis+Gyr 2020.) In Finland the R&D center is located in the city of Jyväskylä and employs roughly 250 professionals working on smart reading systems and smart metering solutions.

1.2 Smart Meters and Smart Grid

Most of the customers of Landis+Gyr for smart meters are energy companies. Smart meters send the meter values to energy companies remotely while “dumb” meters need to have their values read manually. In addition to the basic use case, smart meters have a great amount of functionality. Smart meters often come with displays, features such as prepayment and for example energy companies can remotely cut energy distribution through the meters. Smart meters provide benefits for both the energy companies and households. Because smart meters provide real-time data of the energy usage, energy consumption can be tracked more efficiently, which is good both environmentally and socially wise. (Smart meters explained 2020.)

Electric grid delivers electricity from the power plant to its intended targets such as households or businesses. The actual grid consists of transmission lines, substations and transformers. Smart grids differ from the traditional grids by making two-way communication possible between the utility and the intended customers. Smart grids allow sensing and control along the transmission lines, and the different forms of automation, computers, controls and equipment working together make the grid smart. (What is the Smart Grid? n.d.)

1.3 Objective

The evolving world of remote connections and IoT brings plenty of opportunities; however, also risks. Smart metering solutions provide functionalities that could possibly be used for malicious activity against energy companies, energy distribution and consumers; hence, the cyber risks should be mitigated. Concerns related to cyber security threats in smart energy management have been on the increase all over the world (Cybersecurity of critical energy infrastructure 2019, 1).

The assigner of this research had a need for a practical patch management solution to simplify the process of applying and testing third-party patches. The purpose of this work is to develop a patch management process for Solution Integration Testing (SIT) team to further support patching and testing in a large corporation environment while bringing more security to the field of energy management. The idea of this research is to assure the continuity of stable, secure software and services while increasing the security standards. Preventing possible security threats from escalating due to vulnerabilities in third-party components is the ultimate goal. Efficiently patching and testing the applications will benefit the customers of Landis+Gyr; therefore, this work will bring more security to the field of energy management.

The research starts by identifying and clarifying the theory about vulnerabilities and what kind of threats the applications might encounter. Understanding how the vulnerabilities are reported and how the severities are assessed is a part of the chapter as well. Two real life cyber-attacks related to the topic are demonstrated as well.

The theory about patches and updates is clarified by explaining what they are and why it is difficult to constantly keep software and applications up to date, along with the reasons why it is important to do so. The definition of patch management is demonstrated as well.

The last part of the theory is about the Landis+Gyr AIM product and its third-party components which this work focuses on. The components are demonstrated on a relatively high level.

The actual implementation begins by introducing the testing environment and product versions used in this research. A database security assessment tool is introduced, and its requirements along with the execution phases and results are described. It is followed by the evaluation of the patch management process which consists of information about the Oracle patches, requirements placed by the assigner and theory about the process steps. The last part of the implementation is the practical implementation of the process steps based on the theory composed in patch management process evaluation.

2 Research frame

The problem to be researched is the lack of a clear process to applying and testing software patches in end-to-end testing stage. Some amount of security testing is being carried out by other teams before the product is pushed to end-to-end testing stage. Solution Integration Testing team is responsible for the final verification and testing before the product is released to the customer; thus patch and security testing is important.

2.1 Research Questions

This thesis discovers what kind of difficulties individuals and organizations face when patching software and applications and discusses what kind of threats and risks the lack of patching poses. The theory for the patch management is researched using different publications by reliable authors. The research results combined with the Landis+Gyr's requirements and procedures are used to answer the main research questions which are:

1. What kind of patching process to implement?

The theory for the patching process itself is not enough, and the objective of this thesis was to create a practical process supported by the theory, which was gathered by answering the first research question. The practical part is dependent on the limitations and requirements placed by the assigner. The main question is followed by a sub question:

    1.1. How to implement a patch management process?

Having a theory backed up by different authors and implementing the process based on the theory is a solid foundation. However, making sure everything works after the patching is crucial. The research questions are answered starting from chapter 4.3 and the last question supporting the main question is:

    1.2. How to ensure everything works after patching?

2.2 Research Methods

Qualitative research methods are used to answer the research questions in this thesis. According to Kananen (2019, 26), the criterion for qualitative research is met because the research takes place at the work site, and the idea is to get a better overall view and understanding of the subject. Constructive approach is used to gather information for an existing problem and by creating a new construction of already researched solutions, the existing problem can be solved (Lukka 2001).

2.3 Reliability

The theory of this thesis is based on best practices related to patch management, vulnerabilities and organizational literature

published by professional, official, respected or otherwise trustworthy authors. Differ ent types of utilized publications include but
are not limited to blog posts, documents, websites and articles.

The methods used in this thesis are chosen based on how well they fit the organizational environment, how common they are, how relevant they are and will be in the future. Many of the methods are also modified in order to make them work better in the organization.

3 Theory

3.1 Software Vulnerabilities

3.1.1 Definition

A vulnerability has several different contexts in computing; however, it is often associated with holes or weaknesses in applications and operating systems. They compromise the overall security and are often the result of design flaws or bugs. Vulnerabilities may be used by attackers to cause harm to the stakeholders of the application. The stakeholders of an application include but are not limited to the application owner, users and other entities that rely on the application.
(Vulnerabilities 2020.)

3.1.2 OWASP

The Open Web Application Security Project (OWASP) is a non-profit foundation for improvisation of secure software. OWASP powers community-led open source projects, over 275 local chapters worldwide, tens of thousands of members and industry-leading educational and training conferences. Publications by OWASP are free to use and the foundation operates solely on donations. OWASP was established on 1 December 2001. (About the OWASP Foundation 2020.)

3.1.3 OWASP Top Ten

The OWASP Top 10 is a document for developers about the most common critical security risks to web applications. By mitigating the risks associated in the document, companies and developers are on a path to minimizing security risks in their web applications. The Top 10 risks are presented below: (OWASP Top Ten 2020.)

1. Injection – Injections flaws such as SQL, NoSQL, OS and LDAP injections occur when untrusted data is sent to an interpreter as a part of a command or query. This may result into the interpreter executing unintended commands or even accessing data without proper authorization.

2. Broken Authentication – Incorrectly implemented authentication and session management functions in applications allow attackers to compromise passwords, keys, session tokens and/or to exploit other implementation flaws to hijack users' identities.

3. Sensitive Data Exposure – Some web applications and APIs do not properly protect sensitive data, meaning attackers may steal or modify weakly protected data to do credit card frauds, identity thefts or other similar crimes.

4. XML External Entities – Poorly configured XML processor could parse a reference to an external entity leading to loss of confidential data, denial of service, server-side request forgery, port scanning and/or other system impacts.

5. Broken Access Control – Access control is used to control what users are allowed to do. Often this is not enforced properly, and it can be exploited by hackers to access unauthorized functionality. User accounts and sensitive files including the rights to modify data and access rights are compromised.

6. Security Misconfiguration – Insecure default configurations, wrong ad hoc configurations, open cloud storage, misconfigured HTTP headers and error messages with sensitive information. This is a very common issue which can be avoided with proper configuration.

7. Cross-Site Scripting – XSS allows script execution in browser which may lead to attacker hijacking user sessions, defacing websites or redirecting to malicious websites. Applications sending untrusted data without proper validation are prone to this.

8. Insecure Deserialization – Commonly leads to remote code execution. Will lead to attacks such at replay attacks, injection attacks and privilege attacks.

9. Using Components with Known Vulnerabilities – Any components with sufficient privileges and exposed vulnerabilities may lead to an attack on the system resulting in a possible server takeover or data loss.

10. Insufficient Logging & Monitoring – Lack of logging or monitoring will allow attackers to effectively tamper the system without any attention.

The OWASP Top Ten is mostly beneficial towards people developing web applications; however, CVE and CVSS explained in chapters 3.1.4 and 3.1.5 often contain vulnerability information of the most common security threats in web applications, which are explained in this chapter. Therefore, having at least a high-level understanding of the actual vulnerabilities is beneficial, especially when working with patches and assessing CVSS ratings.

3.1.4 Common Vulnerabilities and Exposures

Common Vulnerabilities and Exposures (CVE) is a list of standardized names for cybersecurity vulnerabilities and other security related exposures. The CVE standard provides a unique identifier for vulnerabilities and exposures and a description of the incident. It is a dictionary type list rather than a database. CVE was launched in 1999 with the aim to standardize the names for all known vulnerabilities and exposures while making data sharing across different databases and tools easier. Before a common standard was implemented most of the cybersecurity tools had their own databases along their own naming standards making it difficult to determine whether another database already referred to a same vulnerability. CVE is now accepted as the industry standard for vulnerability and exposure identifiers. Each CVE entry has the following structure and Figure 1 illustrates what a standard CVE entry looks like: (About CVE 2019.)

- CVE ID number (i.e. CVE-2017-1044)
- Description of the vulnerability or exposure
- References

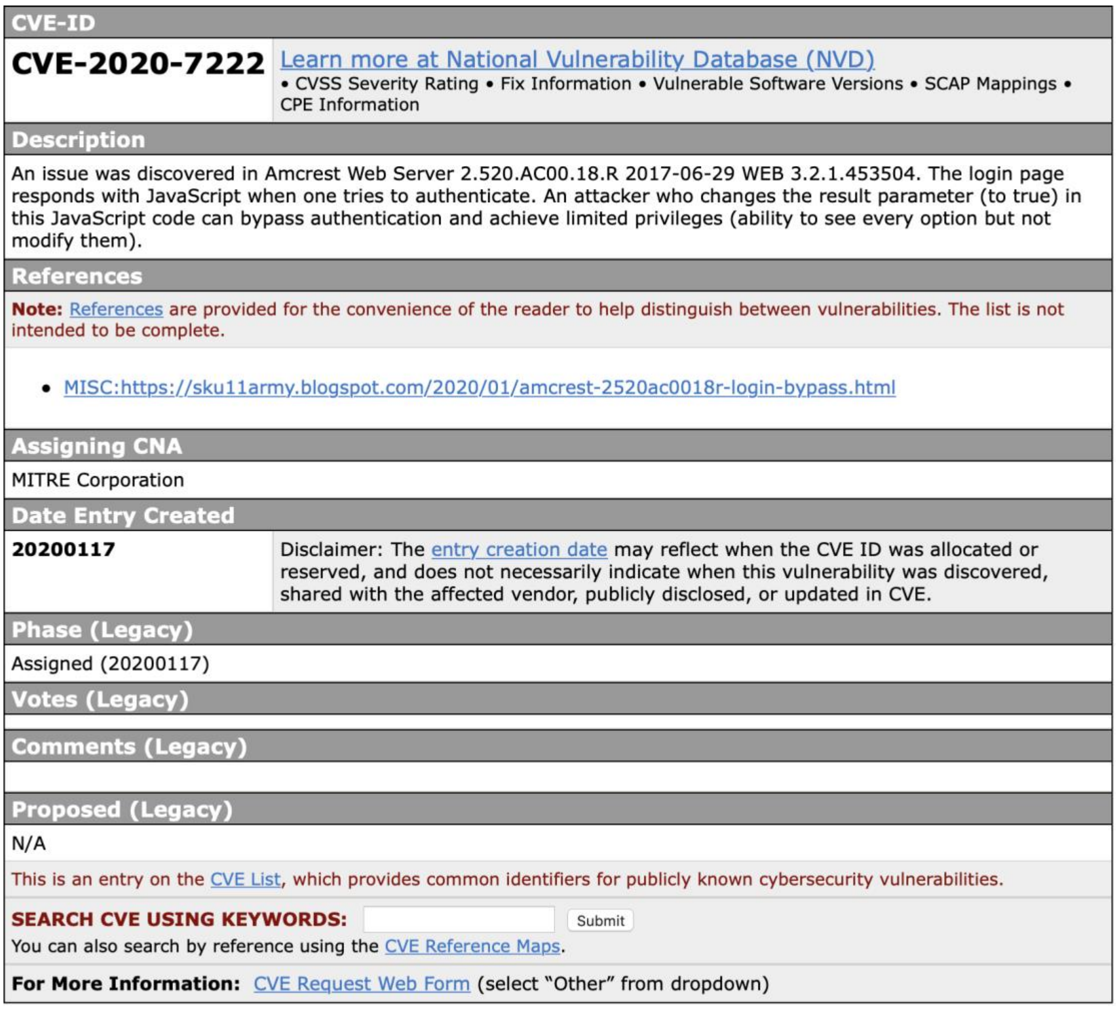

CVE-ID	
CVE-2020-7222	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information
Description	
An issue was discovered in Amcrest Web Server 2.520.AC00.18.R 2017-06-29 WEB 3.2.1.453504. The login page responds with JavaScript when one tries to authenticate. An attacker who changes the result parameter (to true) in this JavaScript code can bypass authentication and achieve limited privileges (ability to see every option but not modify them).	
References	
Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.	
• MISC:https://sku11army.blogspot.com/2020/01/amcrest-2520ac0018r-login-bypass.html	
Assigning CNA	
MITRE Corporation	
Date Entry Created	
20200117	Disclaimer: The entry creation date may reflect when the CVE ID was allocated or reserved, and does not necessarily indicate when this vulnerability was discovered, shared with the affected vendor, publicly disclosed, or updated in CVE.
Phase (Legacy)	
Assigned (20200117)	
Votes (Legacy)	
Comments (Legacy)	
Proposed (Legacy)	
N/A	
This is an entry on the CVE List, which provides common identifiers for publicly known cybersecurity vulnerabilities.	
SEARCH CVE USING KEYWORDS: Submit You can also search by reference using the CVE Reference Maps.	
For More Information: CVE Request Web Form (select "Other" from dropdown)	

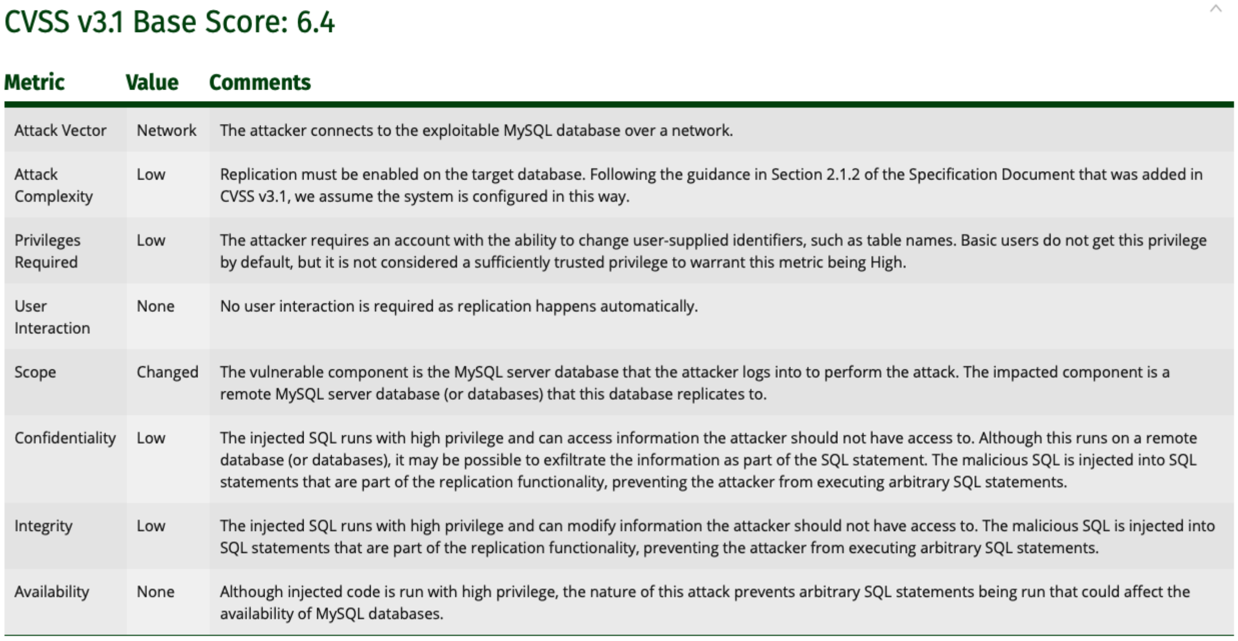
Figure 1. CVE entry CVE-2020-7222

3.1.5 Common Vulnerability Scoring System

Common Vulnerability Scoring System (CVSS) provides the characteristics of a vulnerability and produces a numerical score to reflect its severity. The score can be used to translate the severity to a more readable representation such as low, medium, high and critical, which helps organizations to assess vulnerabilities and prioritize them in their vulnerability management process. CVSS is a standard used by organizations worldwide, and the CVSS Special Interest Group (SIG) is on a mission to improve it. SIG consists of representatives from a broad range of industry sectors from banking and finance to technology and academia. (Common Vulnerability Scoring System SIG 2020.) Below, in Table 1, is an example of a CVSS table of a

MySQL Server database vulnerability reported with the CVE-2013-0375 identifier. (Common Vulnerability Scoring System version 3.1: Specification Document 2020.)

Table 1. CVSS v3.1 table of CVE-2013-0375 vulnerability



CVSS v3.1 Base Score: 6.4

Metric	Value	Comments
Attack Vector	Network	The attacker connects to the exploitable MySQL database over a network.
Attack Complexity	Low	Replication must be enabled on the target database. Following the guidance in Section 2.1.2 of the Specification Document that was added in CVSS v3.1, we assume the system is configured in this way.
Privileges Required	Low	The attacker requires an account with the ability to change user-supplied identifiers, such as table names. Basic users do not get this privilege by default, but it is not considered a sufficiently trusted privilege to warrant this metric being High.
User Interaction	None	No user interaction is required as replication happens automatically.
Scope	Changed	The vulnerable component is the MySQL server database that the attacker logs into to perform the attack. The impacted component is a remote MySQL server database (or databases) that this database replicates to.
Confidentiality	Low	The injected SQL runs with high privilege and can access information the attacker should not have access to. Although this runs on a remote database (or databases), it may be possible to exfiltrate the information as part of the SQL statement. The malicious SQL is injected into SQL statements that are part of the replication functionality, preventing the attacker from executing arbitrary SQL statements.
Integrity	Low	The injected SQL runs with high privilege and can modify information the attacker should not have access to. The malicious SQL is injected into SQL statements that are part of the replication functionality, preventing the attacker from executing arbitrary SQL statements.
Availability	None	Although injected code is run with high privilege, the nature of this attack prevents arbitrary SQL statements being run that could affect the availability of MySQL databases.

3.2 Significant Cyber Attacks

The following incidents are included in this thesis to demonstrate how critical it is to keep applications and software up to date. Both of these attacks were conducted on outdated operating systems and software, and the attacks could have possibly been avoided by having the latest versions and patches, along with proper policies to upkeep the security and patch management. A single weakness in an application may result in a chain of events.

3.2.1 Ukraine 2015 Power Grid Cyber Attack

The first known power outage caused by a cyberattack occurred on 23 December 2015 in Ivano-Frankivsk Oblast in Ukraine. Three power companies were affected by this, Prykarpattyaoblenergo, Chernivtsioblenergo and Kyivoblenergo. These power companies disclosed the incident to be a cyberattack, and other power companies in Ukraine were affected with similar problems as well. These other companies however do not admit to being hacked. The consumers of Prykarpattyaoblenergo energy company were affected most severely. Roughly 30 substations were turned off and 230 000 people were left without power for 1 to 6 hours. (Vijay, Hoikka & Blomqvist n.d, 2.)

It is believed that this attack was conducted by Russian advanced persistent threat group called “Sandworm” and the event occurred during an ongoing military and geopolitical conflict between Ukraine and Russia. The group has recently targeted European Union institutions, American government entities, NATO targets and they have tried to hack European telecommunication companies repeatedly. The group is known to use a hacking tool called BlackEnergy. (ibid., 2.)

Blackenergy is a Trojan malware application designed to perform actions on the host computer. The installer will attempt to bypass user account controls and it will prompt for administrative access if it is not able to bypass them. It exploits a Windows 7 backwards compatibility feature, and the malware solely targets Window operating systems. With the correct privileges it loads plugins over the network which can execute arbitrary tasks on the host computer. (ibid., 3)

Using the malware, the attackers were able to harvest user credentials and passwords stored on the employee workstations. The attackers harvested credentials for virtual private networks used by the employees to access the SCADA control system. As the final step before the actual attack, the attackers installed KillDisk malware on the servers and workstations thus rendering them inoperable. Killdisk malware often overwrite critical system files and in some cases, it overwrites log files such as system logs. (ibid., 4)

Right before initiating the attacks on the substations, the attackers installed custom firmware on the gateway devices to ensure that remote commands cannot be issued to the substations. The switches at the substations had to be manually flipped in order to turn the power back on. The attackers could have done much more damage to the substations; hence, the attack was deemed to be a message of some sort to Ukraine. (Vijay, Hoikka & Blomqvist n.d, 5-7.)

ESET, a Slovakian security firm reported that the systems running on Windows XP were widespread, not all workstations had antivirus software, many systems and software had not been patched, and critical systems were accessible remotely and poorly defended. (ibid., 8.)

3.2.2 WannaCry Ransomware Attack

WannaCry is a crypto ransomware used to extort money. Crypto ransomware works by encrypting files making them unreadable or simply locking the user out of the computer. It does so until the request, a ransom, is paid. Computers running Windows OS were targeted by WannaCry and it demanded the cryptocurrency Bitcoin as the ransom. (What Is WannaCry ransomware? 2020.)

The attacks took place all over the world in May 2017. Wannacry spread through computers running an outdated version of Windows. The incident could have been avoided by having the latest patches released by Microsoft. WannaCry took advantage of a weakness in the Windows operating system, and the hack was allegedly developed by the United States National Security Agency. A security patch mitigating the vulnerability was released roughly two months prior to the attacks by Microsoft. Individuals and organizations that had not updated their systems were left vulnerable. (ibid.)

It is advised not to pay the ransom as there is no guarantee the stolen data will be returned, and future attacks are more likely. In this case, the attackers did not have a way to associate the payment with the victim's computer rendering the payments useless. A great amount of data was lost; however, according to F-Secure some amount of data was restored. (ibid.)

WannaCry affected around 230 000 computers globally. Companies such as Telefónica, a Spanish mobile company and thousands of NHS hospitals and surgeries all across the UK were affected by this attack. One third of the NHS hospitals were affected, which caused devastating results. Ambulances were rerouted leaving people without help, and £92 million was lost after 19 000 appointments were canceled due to the attack. WannaCry spread beyond Europe and crippled computer systems in over 150 countries. It is estimated that WannaCry caused a total of $4 billion in losses all over the world. (What Is WannaCry ransomware? 2020.)

3.3 Software Patches and Updates

3.3.1 Definition

A software patch is generally a chunk of code used to fix, change or update something in the software. Supported programs commonly have their patches released by the original vendor while open-source programs might have several third-party developers releasing them. Single patches often are a temporary solution between version releases. (Rouse n.d.) Patches are not to be confused with updates or upgrades as patches are commonly used to address single issues whereas updates often contain several patches and enhancements, and upgrades often introduce new versions and features. This definition; however, is not an industry standard; nevertheless, it is often acknowledged by the industry. Yantz (2019) states that there are several different types of patches:

- Hotfix – Used to fix a specific issue and unlike normal patches, these are developed and released as soon as possible. Some hotfixes can be applied while the application or system is still running without having to restart or close the program.

- Point release – Minor update intended to fix an error or flaw in a piece of software without adding any features.

- Maintenance Release – Known as incremental update between service packs or versions to fix multiple issues.

- Security Patches – Security patch is a change in an asset in order to correct a weakness caused by a vulnerability. The purpose of a security patch is to prevent exploitation and mitigate threats to an asset.

- Service Pack or Feature Pack – These are major patches and they are a collection of updates, fixes and feature enhancements. They are delivered in a single package and often fix many issues. They often include all the previous patches, hotfixes, maintenance and security patches released before the service pack.

- Unofficial patches – Third parties and user communities often create these patches. Often, It is the result of lack of support from the original developer or the product has reached the end of its life. These patches like the standard patches are designed to address same kinds of issues. Unofficial patches can be malicious.

- Monkey Patches – Used to extend or modify a plugin or software locally without altering the source code.

It must be noted that not all vendors, developers or users have the same naming scheme or release protocol, and the list above is only an example of some common ones. Vendors might have their own naming scheme, e.g. Oracle names its security patch bundles as Critical Patch Updates and single or a small number of security patches as Security Alerts (The Critical Patch Update Program 2020).

3.3.2 Challenges in Patching

Hackers are constantly coming up with new techniques to break software and exploit flaws in order to penetrate security measures. Updating software might seem like a nuisance because when security measures are taken care of, the users do not have a sense of the lurking security threats. According to Boblin (2018), the four most common reasons people delay their updates are:

- Compatibility issues – A major reason for why people don’t regularly update their computer software is due to compatibility issues. People who use specialized software might run into issues with their applications and It is a valid concern for people who need to use their applications without any hinders.

- Lack of knowledge – People might not be aware of why updating is important. This is especially common amongst non tech-savvy people. People might think that updates are a way to make them purchase something.

- Bad experiences – Updates might occasionally break something causing unpleasant experiences.

- Being comfortable – Everything works and anything affecting the environment is seen as a nuisance. People may think that updates bring unnecessary complications.

There are many reasons for not applying updates. The above examples cover only a fraction of the actual reasons. Being aware of the risk associated with running old versions is crucial. Timed and silent updates are utilized by some companies to force patches for the most critical vulnerabilities. Apple is known for deploying silent updates to mitigate security risks (Bohn 2019).

Patching is a challenge and many companies have a hard time keeping up with them. According to Stauffer (2017), the most common reasons why companies are not prioritizing patching higher are the following:

- Too many patches to keep up with
- Patching is manual and time consuming
- Lack of resources
- Applications cannot be patched internally
- End user resistance
- Risk of creating more problems

All these reasons are valid complaints; however, the threats caused by vulnerabilities should weigh more in all instances. Companies often use third-party components in their own products meaning that they are dependent on the vendor to release the patches for any exposed vulnerabilities. This places stress on the companies to apply and test the released patches even more because they have to follow the vendors' release schedule. Some patches are released unexpectedly as well. The whole process and making sure that everything works after applying the patches requires a tremendous amount of resources.

3.3.3 Importance of Patching

Till this day software patching and the lack of it remain as one of the biggest security threats an organization has to deal with. Applications and software running on older versions are easy targets for hackers who want to cause harm to a company. Patching however is a matter of prioritization. (Cihodariu 2019.) Companies should prioritize patching so vulnerabilities in their own assets are mitigated thus providing safety to their customers as well. Having a solid process to patch management is a step to increase the security standards in a company and to optimize resources. The following statistics are based on different surveys done by several different companies and the results are gathered in a single article written by Cihodariu (2019) for Heimdal Security.

- 80% of companies that had a data breach or failed audit could have prevented by patching on time – Voke Media survey (2016)
- Types of malware and ransomware that could have been avoided: WannaCry, NotPetya, SamSam
- 20% of all vulnerabilities caused by unpatched software are classified as High Risk or Critical – Edgescan Stats Report (2018)
- Microsoft reports that most of its customer are breached via vulnerabilities that had patches released years ago – Microsoft's Security Intelligence Report (2015)

Vendors discover and disclose vulnerabilities in their software and then release patches to address these problems. The disclosure of the present vulnerabilities is exposed to adversaries as well. Unfortunately, it might take time for organizations to patch the software instead of doing it as soon as possible. Adversaries are able to analyze and determine how to exploit the weaknesses and the organizations that are yet to patch are vulnerable to the exposed vulnerabilities. ”Patching quickly is essential, as the likelihood of publicly available exploits increases significantly after patches are released.” (Security Vulnerabilities and Patches Explained 2015, 1.)

3.4 Patch Management

Patch management process is a solution to protect enterprises and their products. It is about mitigating the risks related to loss of confidential data and maintaining the integrity of the systems. Patch management, when done properly, can be the most efficient way to mitigate security vulnerabilities and even the most cost-efficient way to do so. (Ruppert 2007, 5.)

It is very unlikely that an application is developed in a way that it does not require corrections, upgrades or modifications in its lifecycle, which is this the reason why a process for patch management is important and required to fix the issues regularly. It is a process that needs to be done routinely in order to be the most effective. One compromised system can lead to other systems being compromised as well. However, it does not mean that a company should treat all of its systems equally but to prioritize its most critical assets instead. Patches should eventually be applied to every system and application though, and not just to the ones most valuable to the company. Multiple security controls, in which patch management is a part of, are an effective way to maintain good enterprise security. Patch management should not be treated as the only solution to avoid security vulnerabilities though. (ibid., 5-7.)

Requirements for a patch management process vary. Variables such as organization size, budget, supported systems, number of employees and systems, location of systems and the number of third-party vendors all influence the process. (ibid., 8.)

3.5 Landis+Gyr AIM

Landis+Gyr’s Gridstream smart metering solution is used to simplify the way energy companies collect and handle their metering data. Smart meters, versatile communication options, advanced software and comprehensive services are included in the solution. AIM is the smart metering system that provides data management capabilities, task flow engine and validation beyond basic HES functions. AIM is capable of efficient multi-energy metering and with a single integrated smart metering software, the utility can manage its AMM infrastructure, enhance business processes and support the network management operations. (Landis+Gyr AIM 2020.)

Support for utility processes: (Landis+Gyr AIM 2020.)

- Business Processes – Billing, balance settlement, customer service, contract management and new business development
- Network Management – Network monitoring, load management and network investment planning
- Smart Metering Operations – Automated data collection and management, device asset management and troubleshooting

AIM is based on modularity and openness and designed for the ever-changing energy market. Its purpose is to help energy companies collect and manage their metering data efficiently, process it in a flexible manner and transfer the information between different parties effortlessly. AIM offers a single source for metering data that can be then transferred to other systems. The architecture of AIM has a high level of openness and interoperability across multi-utility infrastructures, and it enables bi-directional communication with hardware and software platforms developed by other manufacturers. It means that AIM is flexible enough to be integrated with other systems. AIM also offers a fair range of value-adding applications such as deployment and reporting. As of today, Landis+Gyr AIM is running in over 2 million metering points in EMEA region thus proving its reliability and scalability in smart metering operations. (ibid.)

Landis+Gyr AIM uses three main components developed by third-party vendors: Java, Oracle Database and Oracle WebLogic. Oracle Database and WebLogic are the main components covered by the patch management process. AIM uses Oracle DB

Enterprise edition and Java SE 8; however, migration to OpenJDK is underway. This is because Oracle has changed its licensing terms, and patches are not free of charge after January 2019 for commercial use. Free to use Java version with a long-term support for security fixes is required and OpenJDK provides this.

3.5.1 Java

Java is both an object-oriented, class-based programming language and a platform developed by Sun Microsystems in 1995 and later acquired by Oracle. Java is widely used in many applications and different domains such as: (Johari 2020.)

- Banking – Used for transaction management
- Retail – Billing applications written in Java
- Information Technology – Used to solve implementation dependencies
- Android – Java or Java API is used to write Android applications
- Financial services – Used for server-side applications
- Stock market – Algorithms written in Java

The list above does not cover all the use cases of Java because it is really widespread and used in over three billion devices. (Johari 2020.) The different Java platforms are: (Java vs Java EE 2020.)

- Java Standard Edition (SE) – Provides basic functionality and is mostly used to develop APIs for desktop applications.

- Java Enterprise Edition (EE) – Is built on top of SE and provides APIs for running large scale applications. EE is mainly used for web applications.

- Java Micro Edition (ME) – Provides a flexible environment for applications running on embedded and mobile devices (Java Platform, Micro Edition (Java ME)).

Java Virtual Machine

JVM is a virtual machine that enables the computer to run a Java program. When a

Java program is executed, Java compiler first compiles the Java code into bytecode. JVM then translates the bytecode into native machine code which the CPU can read. JVM also optimizes the program memory. Java is platform independent meaning that the code is actually written for JVM instead of the physical machine, which is what ultimately enables Java to run on all platforms. Figure 2 illustrates the program execution phases. (Java JDK, JRE and JVM n.d.)

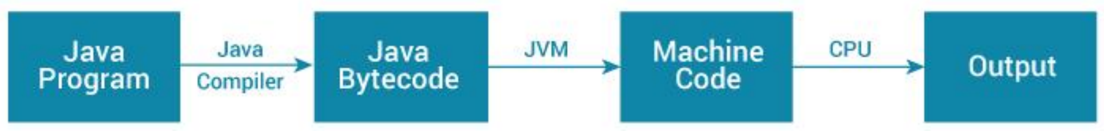


Figure 2. Java program execution

Java Runtime Environment

JRE is a software package that provides Java class libraries and JVM along with other components to run the Java applications. JRE is used when Java applications need to be executed instead of being developed. (ibid.)

Java Development Kit

JDK is a software development kit used to develop Java applications. JDK is bundled with JRE and several development utilities such as compilers, JavaDoc, Java Debugger and more. The relationship between all the components can be seen in Figure 3. (Java JDK, JRE and JVM n.d.)

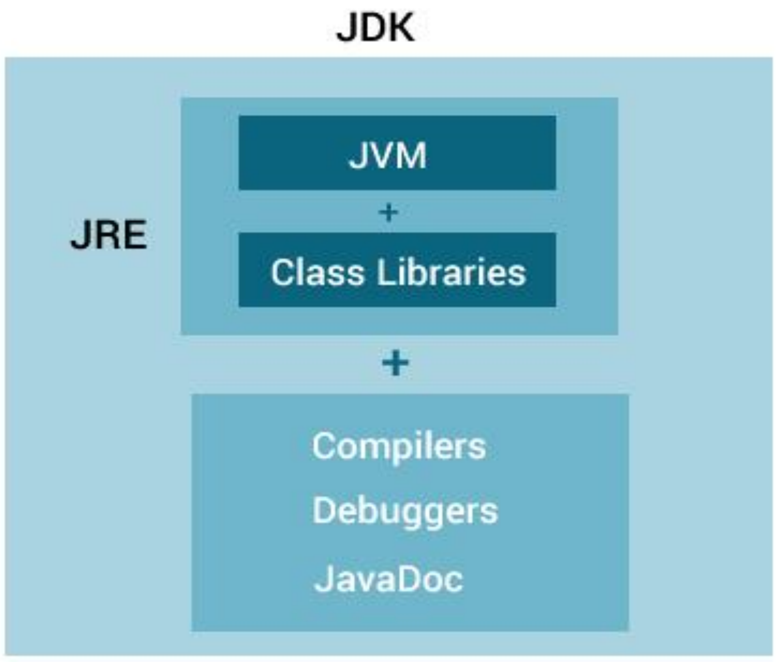


Figure 3. Relationship between JDK, JRE and JVM

As mentioned in chapter 3.5, OpenJDK is a free to use and open source version of the JDK for the Java SE. The project is sponsored and led by Oracle and it is licensed under the General Public License (GNU GPL) version 2 with a linking exception. It means that without linking exception, the components linked to Java class library would be subjected to the terms of the GPL license. (Rouse 2019.)

3.5.2 Oracle Database

Oracle Database is a Relational Database Management System (RDBMS) which manages relational data. Oracle Database allows quick and safe storing and retrieving of data. Oracle Database has several features: (What Is Oracle Database 2020.)

- It can run on various hardware and operating systems such as Windows Server, Unix and different distributions of Linux making it cross-platform.
- Applications running on different platforms can communicate with Oracle Database. For example, applications running on Windows can connect to

Oracle Database running on Linux.

- Oracle Database is ACID (Atomicity, Consistency, Isolation, Durability) compliant meaning that it helps to maintain data integrity and reliability.
- It is one of the first databases to support GNU/Linux before it became a commerce product.

Oracle Database is a powerful database with several structural features that make it popular. There are three main editions of the Oracle Database: (What Is Oracle Database 2020.)

1. Enterprise Edition (EE) is the most commonly used edition. It has no limits on maximum number of CPUs, memory or database size. EE includes features that are not available in other editions.
2. Standard Edition (SE) is the same as EE but with limited features. The only differences are that It is limited to four or lower CPUs and doesn't come with as many features as EE.
3. Expression Edition (XE) is free and is available on Windows and GNU/Linux platforms. It is limited to 2 CPUs, has the maximum of 2GB of RAM and 12GB of user data. XE's features compared to EE and SE are very limited.

In addition to EE, SE and XE, there are other editions in addition to as well; however, these are the main editions. The naming schemes, editions and features slightly differ in different Oracle DB releases.

3.5.3 Oracle WebLogic Server

WebLogic is a scalable, enterprise-ready Java EE application server. WebLogic supports deployment of many types of distributed applications and is ideal for building applications based on Service Oriented Architecture. SOA methodology is aimed at maximizing the reuse of application services. WebLogic provides a standard set of APIs to create distributed Java applications that can access a wide range of services such as messaging services, databases and connections to external systems.

End-users access applications using a web browser client or Java clients. WebLogic allows enterprises to deploy applications in a secure, highly available and scalable environment. Figure 4 illustrates the main Oracle Fusion Middleware stack. (Fusion Middleware Understanding Oracle WebLogic Server 12.1.3 2020.)

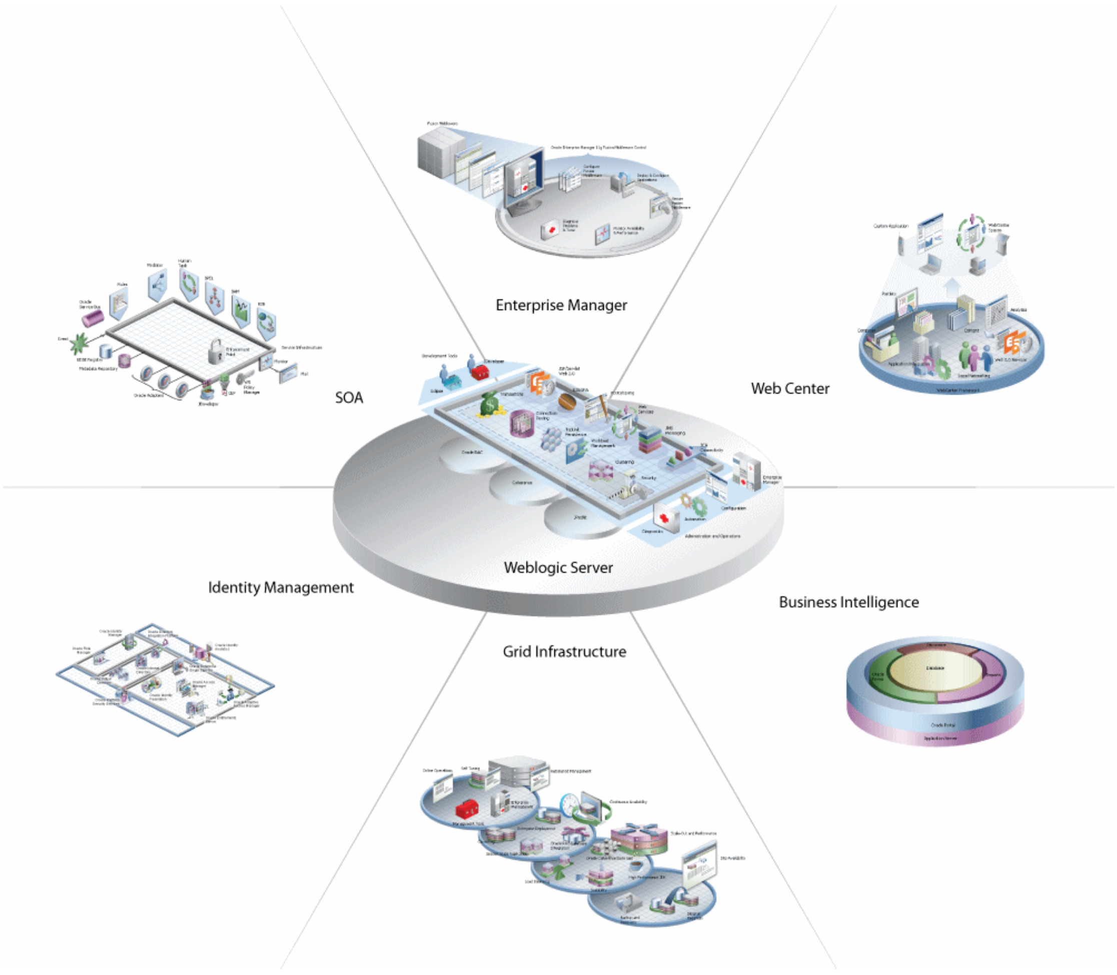


Figure 4. Overview of Oracle Fusion Middleware

4 Implementation

4.1 Test Environment

Patching and testing are conducted in a separate security testing environment running on Landis+Gyr network. It is a virtual Windows Server 2016 running locally; therefore, it is not accessible from the outside the network. The server is powered by

2.39GHz virtual processor and has 24GB of RAM. The IT department of Landis+Gyr can change the virtual server specification if needed. The environment will have the latest and most stable version of Landis+Gyr AIM. Oracle product versions are:

- WebLogic 12c (12.1.3)
- Database 12c (12.2)

The latest versions are used because there are many variations of environments with different combinations of product versions. Including them all in this thesis requires a remarkable amount of resources, and the purpose of this thesis is to provide a basis for the patch management, which can be utilized in the future for all the environments and versions. An environment with the latest versions also works as a simulation of the customer environments.

4.2 Database Security Assessment Tool

Oracle provides a lightweight stand-alone Database Security Assessment Tool (DBSAT) to assess regulatory compliance process by collecting relevant types of configuration information from the database and evaluating the security state to provide recommendations on how to mitigate the identified risks. DBSAT can be used to complement the patching process. For instance, it can be used as a regression test method to check the status of the DB before patching and re-checking after the patching. DBSAT is free for Oracle customers and enables customers to find: (Oracle Database Security Assessment Tool 2020.)

- Security configuration issues and the remediation methods
- Users and their entitlements
- Location, type and quantity of sensitive data

With DBSAT it is possible to: (Security Assessment Tool User Guide 2019.)

- Quickly and easily assess the current security status and identify sensitive data within the Oracle DB.
- Reduce risk exposure using Oracle Database Security best practices, CIS benchmark recommendations and STIG rules.
- Leverage security findings to accelerate compliance with GDPR.
- Improve the overall security of Oracle Databases and promote security best practices.

DBSAT consists of three components: (Security Assessment Tool User Guide 2019.)

Collector – Executes SQL queries and runs OS commands to collect data from the system to be assessed. The data is written into a JSON file which is used by the Reporter in the analysis phase.

Reporter – Analyzes the data collected by the Collector. It generates a Database Security Assessment Report in HTML, Excel, JSON and Text formats.

Discoverer – Functions like the Collector but collects data based on specified settings. The collected data is used to generate a Database Sensitive Data Assessment Report in HTML and CSV formats. The relationships between the DB, DBSAT and output are demonstrated in Figure 5. (Security Assessment Tool User Guide 2019.)

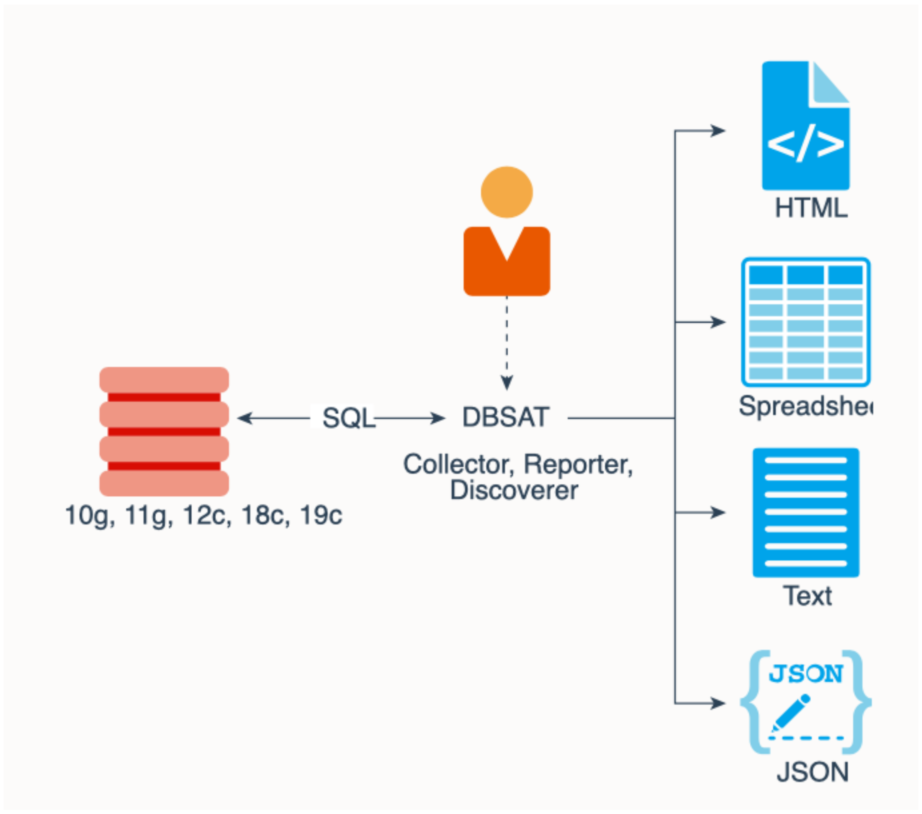


Figure 5. DBSAT Components, Sources and Reports

4.2.1 Prerequisites

DBSAT version 2.2.0 supports Oracle DB 10.2.0.5 and later releases and has several other prerequisites: (Security Assessment Tool User Guide 2019.)

Supported Operating Systems:

- Solaris x64 and Solaris SPARC64
- Linux x86-64
- Windows x64
- HP-UX IA (64-bit)
- IBM AIX (64-bit) & Linux on zSeries (64-bit)

DBSAT uses Zip and Unzip to compress or decompress the generated files. By default, these are included in the Oracle DB installations, however, the Unzip utility is not included in the Oracle DB 12.2 and higher.

Collector must be running on the same server that contains the database in order to collect complete data. It also needs to be run with an OS user with read permissions on files and directories under *ORACLE_HOME* in order to collect and process file systems data using OS commands. Correct DB user privileges are required as well.

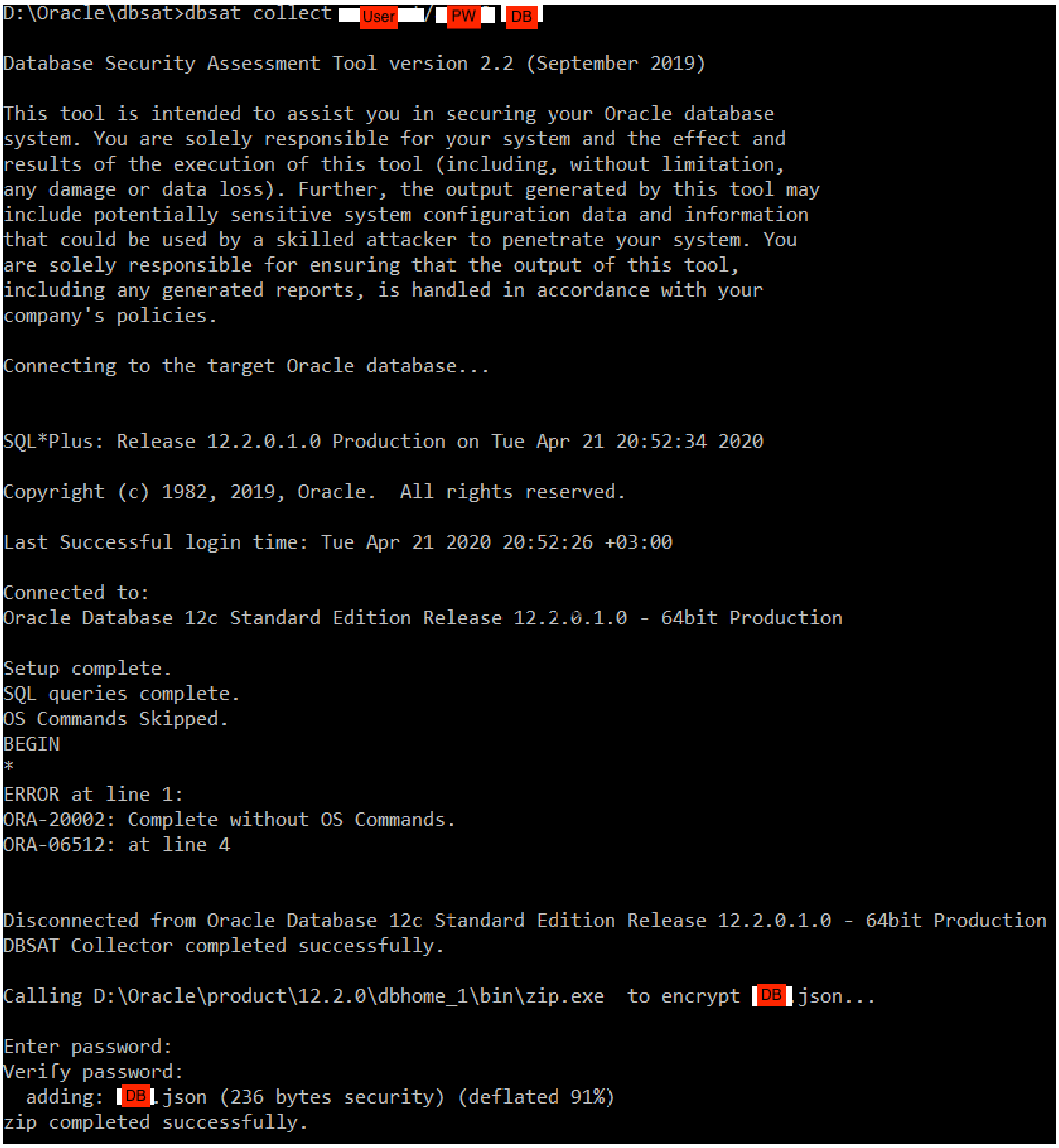
Reporter requires Python 2.6 or later to run.

Discoverer requires JRE 1.8 or later to run. It collects metadata from DB dictionary views and matches them against the patterns specified to discover sensitive data; therefore, sufficient DB user privileges are required as well.

4.2.2 Performing the Assessment

The DBSAT had not been utilized by SIT before and was deemed to be useful when it was discovered; therefore, it was chosen to be a part of this thesis. Utilizing the DBSAT is a great way to increase the security measures with a small amount of effort and without additional costs. DBSAT version 2.2.0 supports the operating systems used by SIT and the tool can be used on Oracle DB versions 10.2.0.5 and later. The assessment was performed on the test environment introduced in chapter 4.1.

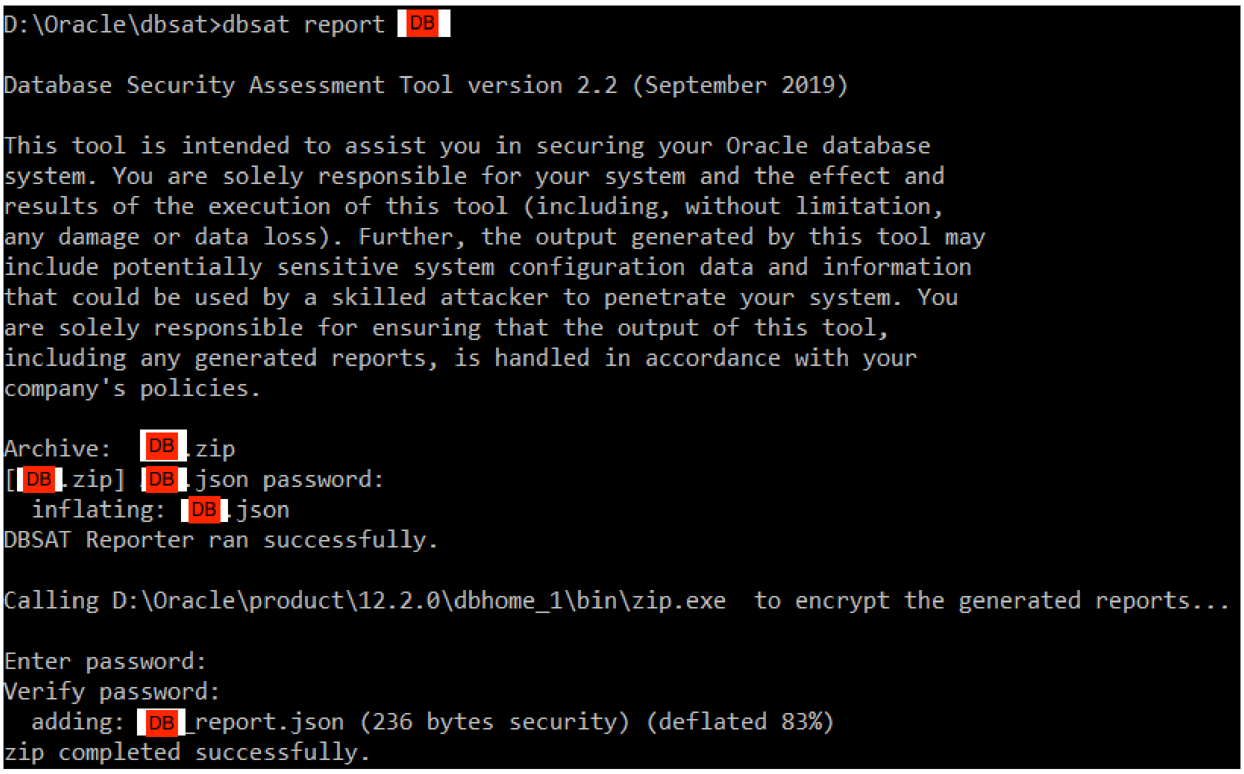
The initial setup was straight forward. The files are downloaded from Oracle site and moved to the server running a database. If the prerequisites are met, there are no extra steps to be taken. The tool should be extracted in a directory named “dbsat” and in this case the directory was created in the Oracle folder which contains other Oracle appliances as well. The next step was to make sure that the *ORACLE_HOME* environment variable path pointed to the Oracle DB directory. The final step was to check whether the Zip and Unzip paths were correct in the DBSAT bat file and the program was ready to execute. The execution begins by running the Collector which is demonstrated in Figure 6.



```
D:\Oracle\dbsat>dbsat collect User/PW DB

Database Security Assessment Tool version 2.2 (September 2019)

This tool is intended to assist you in securing your Oracle database
system. You are solely responsible for your system and the effect and
results of the execution of this tool (including, without limitation,
any damage or data loss). Further, the output generated by this tool may
include potentially sensitive system configuration data and information
that could be used by a skilled attacker to penetrate your system. You
are solely responsible for ensuring that the output of this tool,
including any generated reports, is handled in accordance with your
company's policies.

Connecting to the target Oracle database...


SQL*Plus: Release 12.2.0.1.0 Production on Tue Apr 21 20:52:34 2020

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Last Successful login time: Tue Apr 21 2020 20:52:26 +03:00

Connected to:
Oracle Database 12c Standard Edition Release 12.2.0.1.0 - 64bit Production

Setup complete.
SQL queries complete.
OS Commands Skipped.
BEGIN
*
ERROR at line 1:
ORA-20002: Complete without OS Commands.
ORA-06512: at line 4


Disconnected from Oracle Database 12c Standard Edition Release 12.2.0.1.0 - 64bit Production
DBSAT Collector completed successfully.

Calling D:\Oracle\product\12.2.0\dbhome_1\bin\zip.exe  to encrypt DB.json...

Enter password:
Verify password:
  adding: DB.json (236 bytes security) (deflated 91%)
zip completed successfully.
```

Figure 6. DBSAT Collector phase

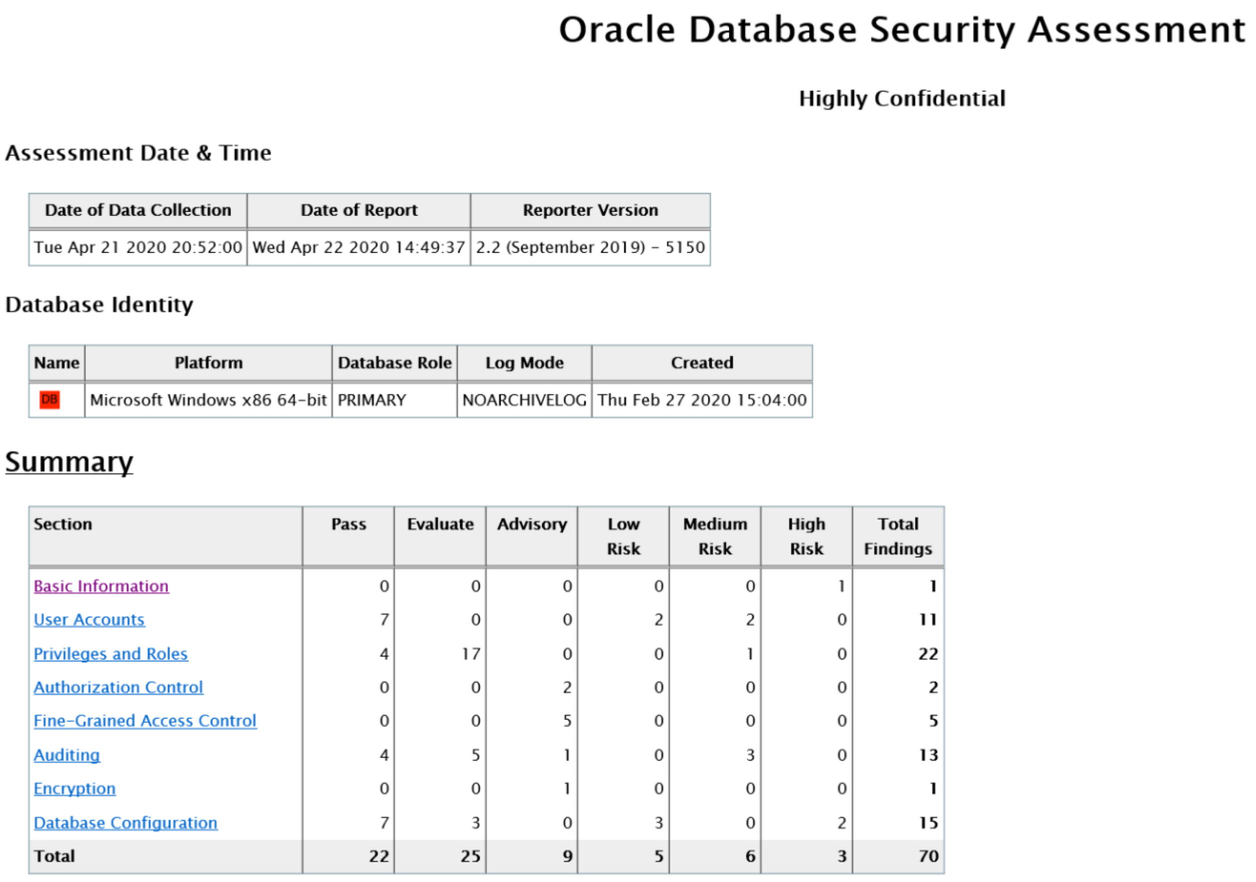
As demonstrated in Figure 6, the Collector first asks for the correct username (User), password (PW) and DB. Then it connects to the DB, gathers the data and exports it as a password protected zip file which contains a JSON file. The zip file was generated with the same name as the DB. The error in the figure was acceptable because no information from the OS was needed. The next step was to run the Reporter and the steps are demonstrated in Figure 7.



```
D:\Oracle\dbsat>dbsat report DB

Database Security Assessment Tool version 2.2 (September 2019)

This tool is intended to assist you in securing your Oracle database
system. You are solely responsible for your system and the effect and
results of the execution of this tool (including, without limitation,
any damage or data loss). Further, the output generated by this tool may
include potentially sensitive system configuration data and information
that could be used by a skilled attacker to penetrate your system. You
are solely responsible for ensuring that the output of this tool,
including any generated reports, is handled in accordance with your
company's policies.

Archive:  DB.zip
[DB.zip] DB.json password:
  inflating: DB.json
DBSAT Reporter ran successfully.

Calling D:\Oracle\product\12.2.0\dbhome_1\bin\zip.exe  to encrypt the generated reports...

Enter password:
Verify password:
  adding: DB_report.json (236 bytes security) (deflated 83%)
zip completed successfully.
```

Figure 7. DBSAT Reporter phase

The Reporter requires the zip filename as a parameter which was generated by the Collector and then it asks for the password which was entered in the Collector phase. The reporter generated three types of reports: A HTML file, An Excel sheet and a text document. The reports provide a lot of information about the DB and an example of a HTML report is demonstrated in Figure 8.


Oracle Database Security Assessment

Highly Confidential

Assessment Date & Time

Date of Data Collection	Date of Report	Reporter Version
Tue Apr 21 2020 20:52:00	Wed Apr 22 2020 14:49:37	2.2 (September 2019) - 5150

Database Identity

Name	Platform	Database Role	Log Mode	Created
DB	Microsoft Windows x86 64-bit	PRIMARY	NOARCHIVELOG	Thu Feb 27 2020 15:04:00

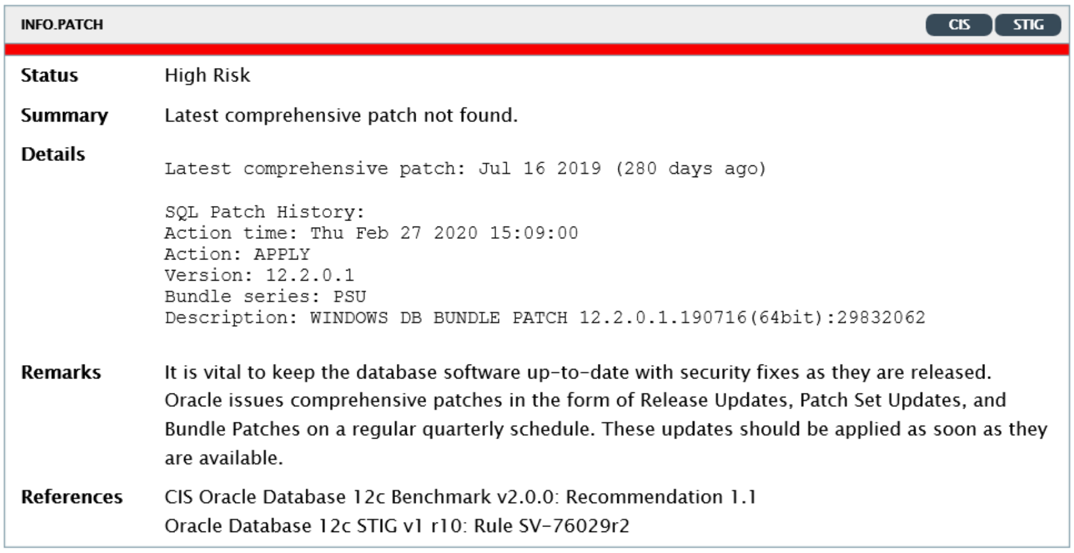
Summary

Section	Pass	Evaluate	Advisory	Low Risk	Medium Risk	High Risk	Total Findings
Basic Information	0	0	0	0	0	1	**1**
User Accounts	7	0	0	2	2	0	**11**
Privileges and Roles	4	17	0	0	1	0	**22**
Authorization Control	0	0	2	0	0	0	**2**
Fine-Grained Access Control	0	0	5	0	0	0	**5**
Auditing	4	5	1	0	3	0	**13**
Encryption	0	0	1	0	0	0	**1**
Database Configuration	7	3	0	3	0	2	**15**
Total	**22**	**25**	**9**	**5**	**6**	**3**	**70**

Figure 8. DBSAT HTML Report

As seen in Figure 8, the report contains a plethora of information about the DB security level. The report provides different sections with the number of total findings and evaluations of the risk levels in the findings. In Figure 9, an example of a "High Risk" finding in "Basic Information" is demonstrated.

Patch Check



Status High Risk

Summary Latest comprehensive patch not found.

Details

```
Latest comprehensive patch: Jul 16 2019 (280 days ago)

SQL Patch History:
Action time: Thu Feb 27 2020 15:09:00
Action: APPLY
Version: 12.2.0.1
Bundle series: PSU
Description: WINDOWS DB BUNDLE PATCH 12.2.0.1.190716(64bit):29832062
```

Remarks It is vital to keep the database software up-to-date with security fixes as they are released. Oracle issues comprehensive patches in the form of Release Updates, Patch Set Updates, and Bundle Patches on a regular quarterly schedule. These updates should be applied as soon as they are available.

References CIS Oracle Database 12c Benchmark v2.0.0: Recommendation 1.1
Oracle Database 12c STIG v1 r10: Rule SV-76029r2

Figure 9. DBSAT HTML Report risk finding

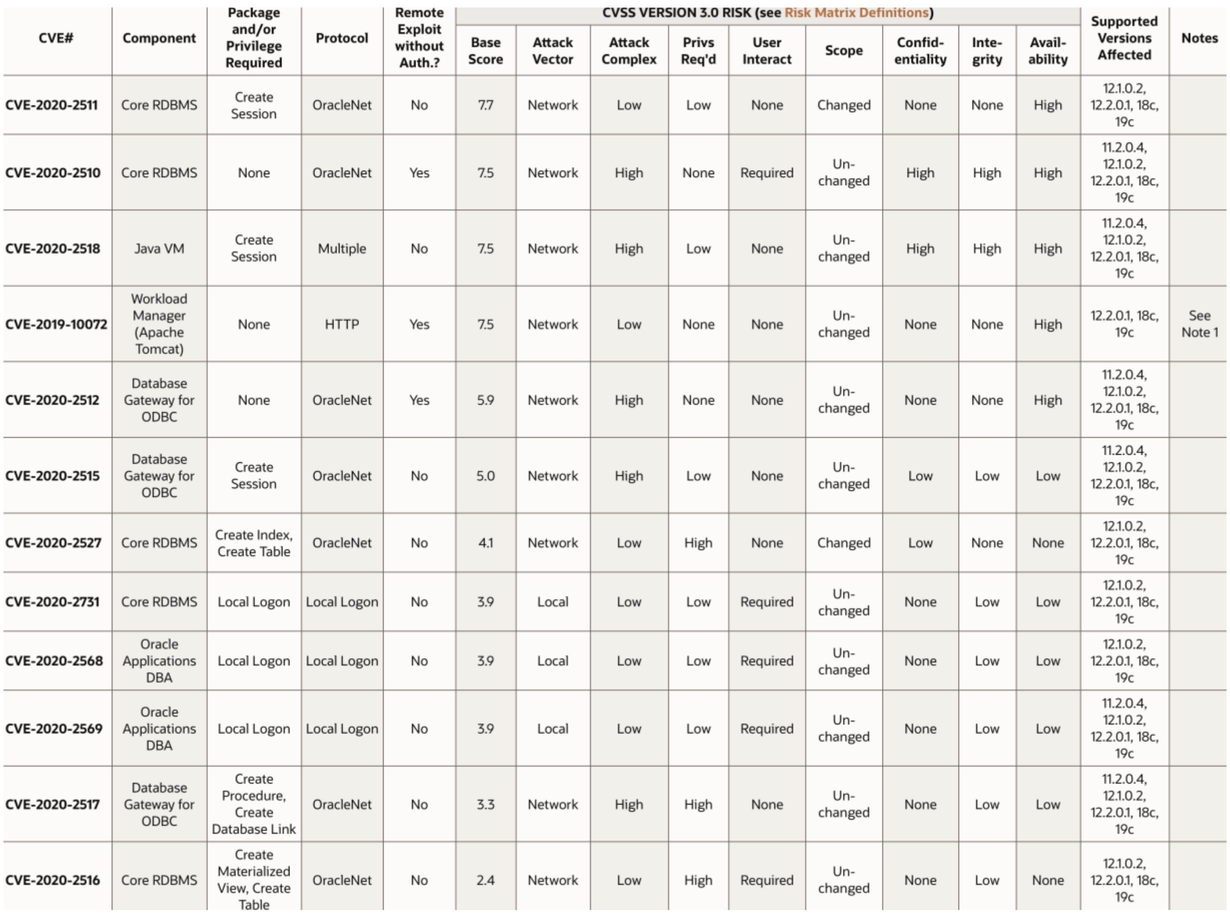
As seen in Figure 9, the DBSAT tool discovered a High-Risk vulnerability which in this case was unpatched software. It provides the remediation method and the patch ID which should be applied. The DBSAT provides lots of information that can be used to complement the patch management process without being too heavy on the resources. The Discoverer was not used in this case because an overall view of the DB using default configuration was enough. It however enables more customized testing in the future if it is deemed necessary.

4.3 Evaluating the Patch Management Process

4.3.1 Oracle Patches

Oracle releases Critical Patch Updates which are a collection of security fixes for Oracle products. They are available for the customers who have a valid support contract with Oracle. CPUs are released on Tuesdays closest to the 17 of January, April, July and October. An announcement of the release is published on Thursdays preceding the CPU release date. (Critical Patch Updates, Security Alerts and Bulletins 2020.) CPUs are cumulative, meaning each update contains security fixes from previous updates and each one has an update advisory describing what kind of security fixes have been added in the update. All the security vulnerability fixes can be tracked back by reviewing the prior CPU advisories. For example, the January 2020 CPU advisory mentions that there are 334 new security patches across a range of different Oracle products. The affected products and versions are mentioned along with a patch availability document for each one. The advisory also provides a Risk Matrix Content table which lists all the newly addressed vulnerabilities as illustrated in Table 2. (Oracle Critical Patch Update Advisory – January 2020, 2020.)

Table 2. Oracle DB CVSS 3.0 Table



CVE#	Component	Package and/or Privilege Required	Protocol	Remote Exploit without Auth.?	CVSS VERSION 3.0 RISK (see Risk Matrix Definitions)									Supported Versions Affected	Notes
					Base Score	Attack Vector	Attack Complex	Privs Req'd	User Interact	Scope	Confid-entiality	Inte-grity	Avail-ability		
CVE-2020-2511	Core RDBMS	Create Session	OracleNet	No	7.7	Network	Low	Low	None	Changed	None	None	High	12.1.0.2, 12.2.0.1, 18c, 19c	
CVE-2020-2510	Core RDBMS	None	OracleNet	Yes	7.5	Network	High	None	Required	Un-changed	High	High	High	11.2.0.4, 12.1.0.2, 12.2.0.1, 18c, 19c	
CVE-2020-2518	Java VM	Create Session	Multiple	No	7.5	Network	High	Low	None	Un-changed	High	High	High	11.2.0.4, 12.1.0.2, 12.2.0.1, 18c, 19c	
CVE-2019-10072	Workload Manager (Apache Tomcat)	None	HTTP	Yes	7.5	Network	Low	None	None	Un-changed	None	None	High	12.2.0.1, 18c, 19c	See Note 1
CVE-2020-2512	Database Gateway for ODBC	None	OracleNet	Yes	5.9	Network	High	None	None	Un-changed	None	None	High	11.2.0.4, 12.1.0.2, 12.2.0.1, 18c, 19c	
CVE-2020-2515	Database Gateway for ODBC	Create Session	OracleNet	No	5.0	Network	High	Low	None	Un-changed	Low	Low	Low	11.2.0.4, 12.1.0.2, 12.2.0.1, 18c, 19c	
CVE-2020-2527	Core RDBMS	Create Index, Create Table	OracleNet	No	4.1	Network	Low	High	None	Changed	Low	None	None	12.1.0.2, 12.2.0.1, 18c, 19c	
CVE-2020-2731	Core RDBMS	Local Logon	Local Logon	No	3.9	Local	Low	Low	Required	Un-changed	None	Low	Low	12.1.0.2, 12.2.0.1, 18c, 19c	
CVE-2020-2568	Oracle Applications DBA	Local Logon	Local Logon	No	3.9	Local	Low	Low	Required	Un-changed	None	Low	Low	12.1.0.2, 12.2.0.1, 18c, 19c	
CVE-2020-2569	Oracle Applications DBA	Local Logon	Local Logon	No	3.9	Local	Low	Low	Required	Un-changed	None	Low	Low	11.2.0.4, 12.1.0.2, 12.2.0.1, 18c, 19c	
CVE-2020-2517	Database Gateway for ODBC	Create Procedure, Create Database Link	OracleNet	No	3.3	Network	High	High	None	Un-changed	None	Low	Low	11.2.0.4, 12.1.0.2, 12.2.0.1, 18c, 19c	
CVE-2020-2516	Core RDBMS	Create Materialized View, Create Table	OracleNet	No	2.4	Network	Low	High	Required	Un-changed	None	Low	None	12.1.0.2, 12.2.0.1, 18c, 19c	

There are risk matrices for each product affected by the CPU. Each vulnerability is identified with a CVE and the matrix uses Common Vulnerability Scoring System to provide information about the severity of the vulnerabilities. The Base Score is between 0.0 and 10.0, where 10.0 represents the most severe vulnerability. It also provides values for metrics that indicate the preconditions required to exploit the vulnerability. This helps to identify the systems that are at most risk so they can be patched first. Oracle adopted the latest CVSS version 3.0 in 2016. (Use of Common Vulnerability Scoring System (CVSS) by Oracle 2020.)

Oracle states that it periodically receives reports of attempts to maliciously exploit vulnerabilities for products that they have already released security patches for. In some instances, some attacks have been successful due to the failure of applying Oracle patches. Therefore, it is highly recommended that the product versions that are used are actively supported and the CPUs are applied without delays. Until applying the patch, it may possible to reduce the risk of an attack by blocking network protocols required by an attack. The same applies for attacks requiring certain privileges or access to certain packages, removing privileges or the ability to access packages from users not in need of privileges may reduce the risk of a successful attack. These approaches; however, may break the application functionality so testing in non-production environments is recommended. Oracle states that this workaround approach should not be used as a long-term solution because it does not correct the underlying problem. If for some reason a CPU has been skipped, Oracle recommends reviewing the prior CPU advisories in order to make sure the products are patched in case of vulnerabilities. (Oracle Critical Patch Update Advisory – January 2020, 2020.)

Oracle issues Security Alerts for vulnerability fixes that are deemed too critical to be included in the next CPU distribution. (Critical Patch Updates, Security Alerts and Bulletins 2020.) These Security Alerts are based on CVE entries and the fix is intended to directly address the vulnerability. Like in a CPU Advisory, the Security Alert Advisory lists the affected products and versions along with the patch availability document. Risk Matrix Content table is included as well.

Oracle has an e-mail subscription to receive notifications of CPU’s and Security Alerts. (Oracle Security Alert Advisory – CVE-2019-2729, 2020.) It is a necessary step in order to be up to date of the new updates.

4.3.2 Requirements

The process should be developed in a way to be quick enough to allow the customers of Landis+Gyr to safely patch the applications, preferably before they even find out about patches or vulnerabilities in the third-party applications. Before any changes, modifications or patches are applied to the environment, checkpoints need to be performed. This is a safety measure to ensure a rollback option if the environment or products are rendered unstable due to the changes. The test environment is a single-server environment, so a checkpoint of the sole server is enough. In a multi-server environment, a checkpoint of all the servers is necessary and all the servers need to be rolled back in order to avoid anomalies. Patch testing should be as swift as possible with a clear process to follow. It is carried out using automation and manual verification is used for the steps that require it. The patching process itself should be implemented without the need of any third-party applications.

4.3.3 Planning the Process

The patch management lifecycle and process will be implemented based on the steps created by Yantz from IT Support Guys. It is a company based in the US with several big-name customers such as Microsoft, Dell, Amazon and Google. IT Support Guys provide solutions for Cloud Services, Desktop Support, VoIP Services, Backup and Recovery, Networking and Security. According to Yantz (2019), the process consists of 10 steps. It must be noted that these are general steps in order to create the process and a single default template will not work for every organization as is; however, the steps will provide a basis for the patching framework in which the process will be consisted with.

Discovery – Keeping an inventory of the assets involved in the process is a good practice. All devices, hardware, systems, operating systems and their versions, third-party software and applications should be listed. When organizations grow, it becomes increasingly difficult to keep up with the systems and programs; thus, an up-to-date inventory is valuable. Every SIT server environment has a Wiki page of its own. The page provides a broad range of information of the environment and the same template is used for all environments. It can be used as the location to store and maintain the inventory list. Inventory is not an essential step of this process in this study because the focus is only on two applications. However, this process is developed with future usability in mind; hence, a list of the assets can prove to be beneficial.

Prioritization – Assessing the risk level and priority of systems and applications makes it easier to direct the resources appropriately. It is necessary to know what assets are at most risk. The vulnerabilities in Oracle applications are assessed by the vendor itself. As mentioned in chapter 4.3.1, all the reported vulnerabilities have a CVSS rating to assess how severe they are. This reduces the workload of patching applications by a huge margin; however, it forces to trust the vendor on the severity ratings. Evaluating the vulnerability severities is a topic of its own and because the focus of this work is on the patching process; the evaluations carried out by Oracle are sufficient and decisions can be made based on the CVSS score. Australian Cyber Security Centre ACSC states that security vulnerabilities with varying severity should be patched in the following timeframe: (Assessing Security Vulnerabilities and Applying Patches 2020.)

- Extreme risk – Within 48 hours of the patch release
- High risk – Within two weeks
- Moderate or low risk – Within one month

ACSC also suggests that patch deployment is prioritized when resources are constrained. Temporary workarounds are possible as well, and Oracle has its own guideline for this. However, it is not recommended if a patch is available. Prioritization requires manual verification from a user, and it should begin by reviewing the Risk Matrices in Oracle CPU advisories or Security Alerts. Preparation is to be carried out by reviewing the CPU Pre-Release Announcement to check whether any WebLogic or Oracle DB versions used are affected. Alternatively, the Recommended Patch Advisor by Oracle can be used to list all the recommended patches; however, this can only be done after the patch is released and does not provide information about the severities.

Policies – There should be policies regarding which systems and applications should be patched and what conditions and requirements are to be followed. The purpose of a patching policy is to set rules and provide a framework for vulnerability identification, threat assessment, priority ranking and remediation. Responsible personnel or a team should be composed in order to create and maintain a patch management process. They are either responsible for the monitoring of patch releases and security alerts and/or applying the patches. They are responsible for identifying the affected applications and versions in patch notes and assessing the severity ratings of the vulnerabilities. Prioritizing the patching timeline is to be done by them as well by following the conditions in “Prioritization”. (Example of CVSS based Patching Policy n.d.) They should test the environment stability in case of errors or conflicts after the patches have been applied.

Monitoring – There are multiple ways to do this. It can be done manually, using programs to automatically check for updates or simply having an e-mail subscription to receive notifications of updates. As mentioned in chapter 4.3.1, Oracle releases CPUs quarterly; therefore, the approximate date of releases is known a year in advance, which allows planning in advance. However, Oracle may release Security Alerts or other unexpected updates in the case of severe threats to its customers. They inform the customers by e-mail notifications so the personnel responsible of patch management should subscribe to the e-mailing list. Alternatively, the Oracle “Critical Patch Updates, Security Alerts and Bulletins” page can be monitored. An RSS feed is also an option. Oracle publishes the following information in its CPU Pre-Release Announcements:

- Name and version numbers of the products affected by vulnerabilities
- Number of security fixes for each product suite
- CVSS score for each product suite
- Additional information

Oracle also states that the Pre-Release Announcement content may change, and the content is official when the CPU Advisory is released. (The Critical Patch Update Program 2020.)

Testing – Having a test environment solely for patch testing is recommended. It allows monitoring in case the changes break something in the environment or applications without causing any critical disturbances. The testing environment should be stable before applying any patches. SIT team uses environment monitoring software which will alert for any problems in the environment. Test automation is used to check whether the environment & AIM and HES applications are functional. A third-party tool is used to deploy the test automation and the results are reported in a web page.

Configuration Management – In case of any problems, having documentation of the intended changes and results allows quick identification of unintended changes. The results should be documented after the testing phase. Landis+Gyr has an internal Wiki page for Oracle DB and WebLogic patches which is used to report in which AIM/HES release the patch has been tested, the patch number and what environment was used.

Roll out – After the patch has been validated and the test environment is stable without any errors or conflicts, other environments running the same AIM/HES versions can be patched. Precautionary measures need to be taken as well when patching other environments.

Auditing – Auditing is carried out to monitor the environments for any errors or incompatibility issues that might occur after patching. After successful patching the environment should be continuously monitored for any unexpected problems. Zabbix is used as the monitoring tool, and SIT has a rotating list of personnel who monitor the environments. Therefore, this step does not require additional work because it is already being done.

Reporting – Informing the stakeholders and persons relevant to the applications and systems is important. This will ensure everyone understands how important patch management is. The purpose of this process is to quickly assess whether a patch can be safely applied to an environment running the latest versions of AIM/HES products. The results should be shared especially with the stakeholders working in the customer interface.

Reviewing, Optimizing and Repeating – Rinse and repeat the lifecycle of the patch management process. It is a never-ending cycle as long as the applications get new patches. Alterations to the process should be done, e.g. new applications added to the process if necessary.

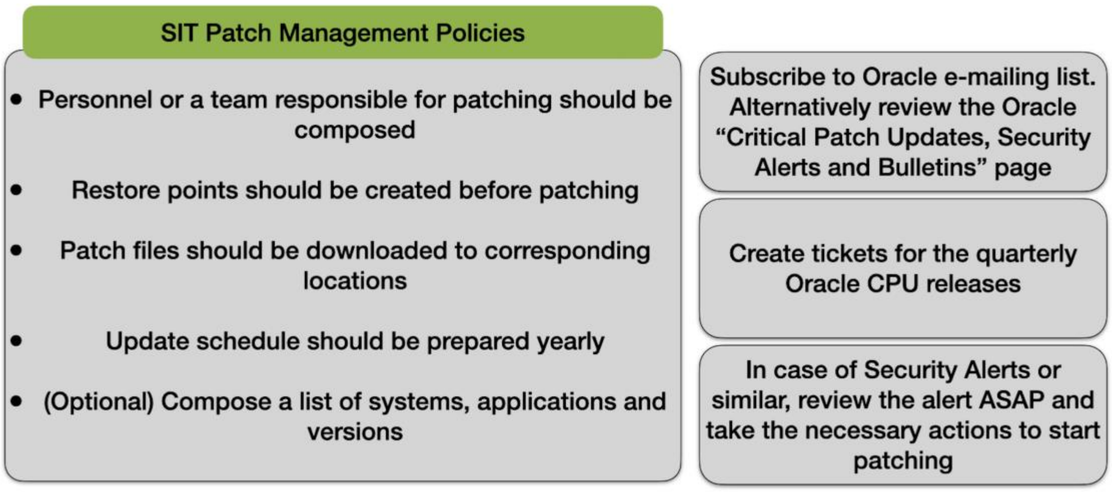
4.4 Practical Implementation of the Process

4.4.1 Policies and Requirements

These are the general policies which should be followed in order to upkeep a patch management process. The policies are consisted based on the theory from the chapter 4.3.3. It is up to the team manager to assign the responsible team or personnel to handle the process tasks and how they are enforced and implemented to the team's routine. As this process is consisted primarily for the Oracle WebLogic and DB products, further expansion and development of the process requires more rules and policies which can be consisted using the theory from chapter 4.3.3. However, the rules consisted for this process are sufficient for the current needs. Alterations need to be made if Oracle changes its release schedule or makes a similar change. A simplified structure of the policies is visualized in Figure 10.

- Team or personnel should be assigned for patch management
    - Personnel responsible for patching should subscribe to the Oracle E-Mailing list. Alternatively, the "Critical Patch Updates, Security Alerts and Bulletins" page can be routinely viewed for any updates.

- Update schedule should be prepared yearly
    - Preparation for the quarterly CPU releases should be planned in advance. Create a study ticket of the upcoming CPU Pre-Release announcements and reserve the resources.
    - In case of Security Alerts or similar review the alert as soon as possible and take the necessary actions to start the patching process.
- Restore points should be created
- Patch files should be downloaded to the corresponding location
- Optionally an inventory list of the systems, applications and versions can be composed


SIT Patch Management Policies

- Personnel or a team responsible for patching should be composed
- Restore points should be created before patching
- Patch files should be downloaded to corresponding locations
- Update schedule should be prepared yearly
- (Optional) Compose a list of systems, applications and versions

Subscribe to Oracle e-mailing list. Alternatively review the Oracle "Critical Patch Updates, Security Alerts and Bulletins" page

Create tickets for the quarterly Oracle CPU releases

In case of Security Alerts or similar, review the alert ASAP and take the necessary actions to start patching

Figure 10. Visualization of the Patch Management Policies

4.4.2 Pre-Patching and Monitoring

A major requirement for the process was swiftness and staying ahead of the customers is preferable. In order to do this, any patch announcements released by Oracle should be reviewed by a responsible team member as soon as possible. Preparation for the patching should begin by following the steps composed in this chapter. By following the steps, it is possible to stay ahead of the customers or at least be informed of any vulnerabilities in a timely manner. Even if the patch is not tested before the customers reach out asking, it is good to acknowledge the situation and be aware of the severity of the vulnerabilities. Being able to respond to the customer query with the prioritization and schedule of the patching is better than informing them that the patch is not yet being tested. A simplified and clear flowchart of the process steps is visualized in Figure 11.

1. Refer to the Oracle CPU Pre-Release announcement when it is announced to check for any affected applications and versions. If the CPU is already released this step is obsolete.

2. Assess the severity ratings of the vulnerabilities by inspecting the Risk Matrices in the CPU Adversary

   a) If a Security Alert or similar is released, assess its Risk Matrices

3. Create a ticket for application patching with the relevant urgency. For example:

   1) 7.5 – 10.0 = Extreme risk
   2) 5.0 – 7.5 = High risk
   3) Less than above = Moderate or low risk

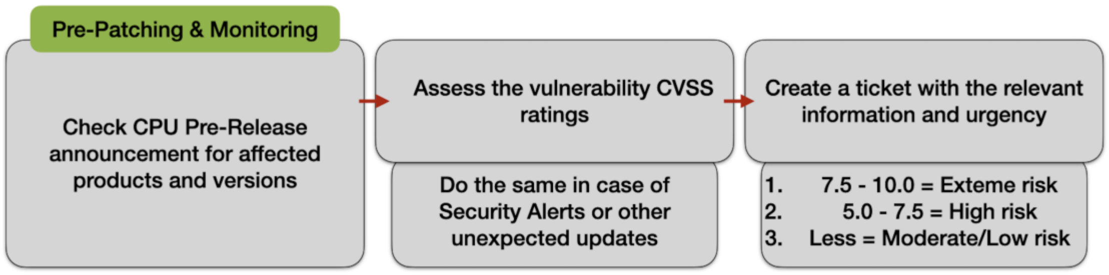

Pre-Patching & Monitoring
Check CPU Pre-Release announcement for affected products and versions
Assess the vulnerability CVSS ratings
Do the same in case of Security Alerts or other unexpected updates
Create a ticket with the relevant information and urgency
1. 7.5 - 10.0 = Exteme risk
2. 5.0 - 7.5 = High risk
3. Less = Moderate/Low risk

Figure 11. Flowchart of Pre-Patching & Monitoring

4.4.3 Patching and Testing

The actual patching and testing are conducted in this phase. The steps are consisted using SIT's prior procedures while following the theory from chapter 4.3.3 and combining the theory with the requirements placed for this process. Ensuring a rollback option in case the environment is rendered unusable due to patching and the environment stability is verified using automation tools and manual methods. The least number of new procedures is introduced in this phase because patching itself is not a new thing in SIT. This new phase description however does provide a clear and simple guideline on how the patches should be applied, as required. The steps are visualized as a flowchart in Figure 12.

1. Download the patch files and move them to the correct product and version folder located in the online file storage. If the files are already downloaded this step is obsolete.

2. Copy the patch files from the online file storage to the environment

3. Check environment monitoring tool for any environment problems

    a. If any problems occur attempt to resolve them before continuing

    b. In case of any unfixable problems: take a note of the them and if they are not deemed critical or won't prevent patching continue with the process

    c. If patching is not possible due to a major problem, consider testing using another environment or consult what can be done about the problem. Prioritize the problem mitigation based on the relevant CVSS vulnerability ratings in the patch notes.

4. Run test automation and review the results

5. Create a checkpoint of the environment

6. Follow the Oracle guideline on how to apply the patch

7. After applying the patch with the OPatch utility it should give feedback whether the patching was successful or not

    a. Repeat steps 3. and 4. if patching was successful

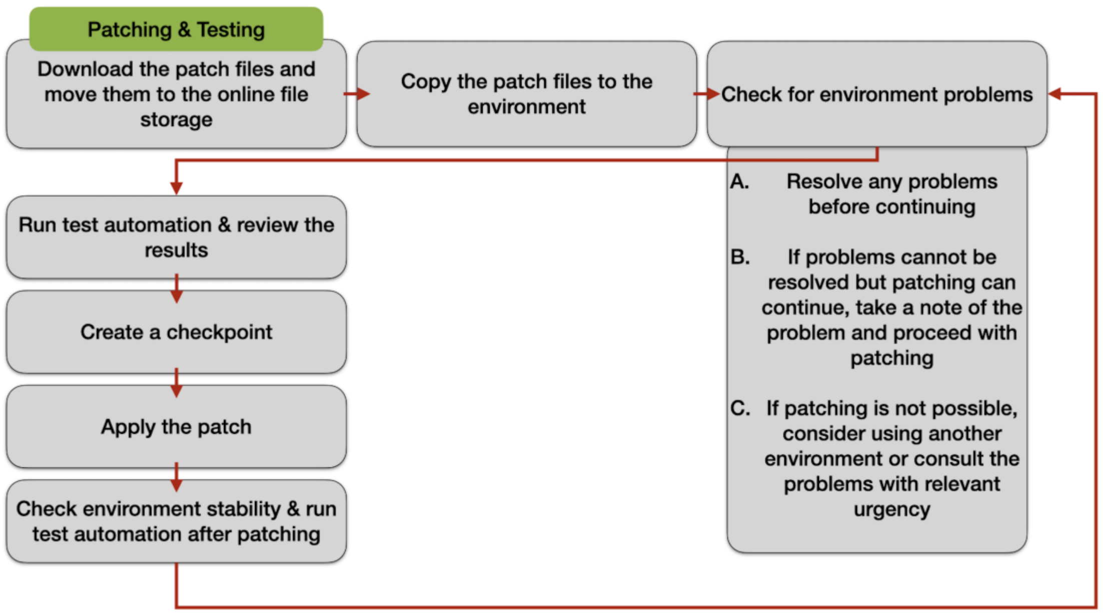

Patching & Testing
Download the patch files and move them to the online file storage
Copy the patch files to the environment
Check for environment problems
A. Resolve any problems before continuing
B. If problems cannot be resolved but patching can continue, take a note of the problem and proceed with patching
C. If patching is not possible, consider using another environment or consult the problems with relevant urgency
Run test automation & review the results
Create a checkpoint
Apply the patch
Check environment stability & run test automation after patching

Figure 12. Flowchart of Patching & Testing

4.4.4 Post-Patching and Reporting

This is the final phase of the patching process. Like in the prior phases, this step also utilized old methods along with new ones. If the patching process was successful and went as intended, the customers and relevant stakeholders can be informed with the results. It is up to the team how it is done and what kind of information is provided. A process should constantly evolve and reviewing it is necessary, in case any changes or alterations are needed. This whole process is a framework which can be shown to customers to demonstrate what kind of measures are taken in order to test the patches. The process was constructed strictly following the given requirements and no third-party tools are necessary. A visualization of the final steps can be seen in Figure 13.

1. Update the internal Oracle DB & WebLogic patch Wiki page

    1) Which environment was used to test the patch
    2) Product versions
    3) Patch ID

2. Inform the stakeholders and relevant people with the results. Especially the personnel working in the customer interface should be informed.

3. The process should be reviewed for any necessary alterations

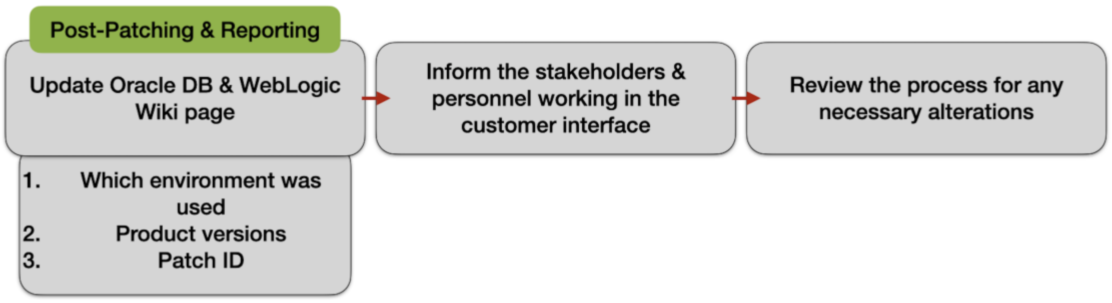

Post-Patching & Reporting
Update Oracle DB & WebLogic Wiki page
1. Which environment was used
2. Product versions
3. Patch ID
Inform the stakeholders & personnel working in the customer interface
Review the process for any necessary alterations

Figure 13. Flowchart of Post-Patching & Reporting

5 Discussion & Conclusions

The objective of this thesis was to create a patch management process using best practices, methods and guidelines composed by the industry professionals and organizations and which can be used for different applications, versions and software now and in the future. In this thesis the practical implementation of the patch management process was composed for Oracle products because SIT has many different variations of environments and Oracle product versions, which makes it difficult to keep up with all of them. The idea was to enhance the security standards in the energy sector through patch management.

The theory for the process steps was established by mixing both the best practices, methods and guidelines with SIT's own procedures. It also required prior knowledge and headwork from the author of this thesis. With the author having hands on experience working in the SIT team, it was easier to plan on how to implement new patch management procedures in the team. As a result, the new process complements the prior way of applying the patches, which can be seen as a good thing because the process consists of familiar procedures. This work utilized the best practices from the prior patching procedures and mixed them with new ones to create a process which is proven to be working along with new enhancements. The basis of the process is implemented in a way which makes it is possible to be altered.

Patch management is a broad field with many variations and variables, and thus creating the perfect process with everything covered is nearly impossible or really difficult. Different applications have different vendors with their own patch management policies and having to follow them all is difficult. This work however fulfilled the requirements put in place but it is only the basis for an ongoing project which patch management is. This work successfully provided the necessary baseline to initialize proper patch management starting with the Oracle products.

There are many options available for handling the burden of patching and some of them cost money. A requirement for this thesis was that the process should be done without causing any additional costs. For example, the ten process steps introduced in chapter 4.3.3 were the basis of the process and composed by Yantz from IT Support Guys. Even though Yantz provided the steps for free and they can be utilized to create a patch management process, the idea was to sell a service. Evaluating the sources which can be used to create the process steps was difficult and required a lot of criticality to avoid misleading information. Forming the process manually made it possible to make sure that everything works within the organization while complementing its ways of handling tasks.

The environment monitoring tools, and test automation have their own role in making sure that the environment is stable after patching. They provide an overall view of the environment and whether the AIM/HES applications or parts of them are working or not. As a new addition the DBSAT tool introduced in chapter 4.2 proved to be useful in addition to the patch management process. For example, in chapter 4.2.2 the DBSAT reported a “High-Risk” vulnerability which in that case was an outdated patch. Monitoring tools and test automation provide the basic information of the environment while the DBSAT tool provides specific information of the databases. The results can be analyzed for a better understanding of the security level of the databases and whether any actions need to be taken in some sectors. It is possibly beneficial for the customers of Landis+Gyr as well because it checks for problems in a way that has not been done before.

As mentioned, patch management is an ongoing process which needs to be constantly maintained in order to stay relevant and up to date. There; however, are many options that can be used to improve the methods. For example, Ansible is a free and open-source tool that can be used to automate Oracle patches, and there are guides on how to do it, using custom scripts to ease the use of Opatch and SQLPlus, the tools used to apply the patches. Oracle even has its own "Recommended Patch Advisor" which was mentioned in "Prioritization" part of chapter 4.3.3 but what really sets this manually created process apart from the premade solutions is that the severities are evaluated manually. This helps to prioritize how, when and what should be patched and in what timeframe. On the organizational level processes should be followed, and this thesis provides the basis for it. There are several parts of the process that can be done differently and even automatically; yet, including everything in this thesis is virtually impossible, and the goal was not to offer a solution which covers every aspect of patching but provides a solid foundation for further development.

References

About CVE. 2019. Page on CVE site. Accessed on 9 March 2020. Retrieved from https://cve.mitre.org/about/

About Landis+Gyr. 2020. Page on Landis+Gyr site. Accessed on 15 January 2020. Retrieved from https://www.landisgyr.com/about/

About the OWASP Foundation. 2020. Page on OWASP site. Accessed on 8 March 2020. Retrieved from https://owasp.org/about/

Assessing Security Vulnerabilities and Applying Patches. 2020. Document on Australian Government site. Accessed on 10 April 2020. Retrieved from https://www.cyber.gov.au/sites/default/files/2020-04/PROTECT%20-%20Assessing%20Security%20Vulnerabilities%20and%20Applying%20Patches%20%28April%202020%29.pdf

Boblin, P. 2018. Why People Don't Update Their Computers. Accessed on 5 March 2020. Retrieved from https://www.techzone360.com/topics/techzone/articles/2018/07/13/438785-why-people-dont-update-their-computers.htm

Bohn, D. 2019. Apple is silently removing Zoom's web server software from Macs. Accessed on 5 March 2020. Retrieved from https://www.theverge.com/2019/7/10/20689644/apple-zoom-web-server-automatic-removal-silent-update-webcam-vulnerability

Cihodariu, M. 2019. Software Patching Statistics for 2019: Common Practices and Vulnerabilities. Accessed on 6 March 2020. Retrieved from https://heimdalsecurity.com/blog/software-patching-statistics-practices-vulnerabilities/

Common Vulnerability Scoring System SIG. 2020. Page on First site. Accessed on 12 April 2020. Retrieved from https://www.first.org/cvss/

Common Vulnerability Scoring System version 3.1: Specification Document. 2020. Page on First site. Accessed on 12 April 2020. Retrieved from https://www.first.org/cvss/examples

Critical Patch Updates, Security Alerts and Bulletins. 2020. Page on Oracle site. Accessed on 6 April 2020. Retrieved from https://www.oracle.com/security-alerts/

Cybersecurity of critical energy infrastructure. 2019. Document on European Parliament site. Accessed on 23 January 2019. Retrieved from http://www.europarl.europa.eu/RegData/etudes/BRIE/2019/642274/EPRS_BRI(2019)642274_EN.pdf

CVE-2020-7222. 2020. CVE entry on CVE site. Accessed on 31 March 2020. Retrieved from https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-7222

Example of CVSS based Patching Policy. N.d. Document on First site. Accessed on 12 April 2020. Retrieved from https://www.first.org/cvss/v2/cvss-based-patch-policy.pdf

Fusion Middleware Understanding Oracle WebLogic Server 12.1.3. 2020. Document on Oracle site. Accessed on 6 March 2020. Retrieved from https://docs.oracle.com/middleware/1213/wls/INTRO/intro.htm#INTRO123

Java JDK, JRE and JVM. N.d. Java tutorial on Programiz site. Accessed on 17 March 2020. Retrieved from https://www.programiz.com/java-programming/jvm-jre-jdk

Java Platform, Micro Edition (Java ME). 2020. Page on Oracle site. Accessed on 15 April 2020. Retrieved from https://www.oracle.com/java/technologies/javameoverview.html

Java vs Java EE. 2020. Tutorial on EDUCBA site. Accessed on 15 April 2020. Retrieved from https://www.educba.com/java-vs-java-ee/

Johari, A. 2020. What Is Java? Accessed on 17 March 2020. Retrieved from https://www.edureka.co/blog/what-is-java/

Kananen, J. 2019. Opinnäytetyön ja pro gradun pikaopas: avain opinnäytetyön ja pro gradun kirjoittamiseen. Jyväskylä: Jyväskylän ammattikorkeakoulu

Lukka, K. 2001. Kari Lukka: Konstruktiivinen tutkimusote. Article on METODIX site. Accessed on 4 May 2020. Retrieved from https://metodix.fi/2014/05/19/lukka-konstruktiivinen-tutkimusote/

Landis+Gyr AIM. 2020. Page on Landis+Gyr site. Accessed on 30 March 2020. Retrieved from https://www.landisgyr.fi/product/gridstream-aim/

Oracle Critical Patch Update Advisory – January 2020. 2020. Page on Oracle site. Accessed on 6 April 2020. Retrieved from https://www.oracle.com/security-alerts/cpujan2020.html

Oracle Database Security Assessment Tool. 2020. Page on Oracle site. Accessed on 20 April 2020. Retrieved from https://www.oracle.com/database/technologies/security/dbsat.html

Oracle Security Alert Advisory – CVE-2019-2729. 2020. Page on Oracle site. Accessed on 6 April 2020. Retrieved from https://www.oracle.com/security-alerts/alert-cve-2019-2729.html

OWASP Top Ten. 2020. Page on OWASP site. Accessed on 8 March 2020. Retrieved from https://owasp.org/www-project-top-ten/

Rouse, M. N.d. Software patch/fix. Accessed on 5 March 2020. Retrieved from https://searchenterprisedesktop.techtarget.com/definition/patch

Rouse, M. 2019. OpenJDK. Accessed on 22 April 2020. Retrieved from https://www.theserverside.com/definition/OpenJDK

Ruppert, B. 2007. Patch Management. Accessed on 30 March 2020. Retrieved from https://www.sans.org/reading-room/whitepapers/iso17799/patch-management-2064

Security Assessment Tool User Guide. 2019. Wiki page on Oracle site. Accessed on 20 April 2020. Retrieved from https://docs.oracle.com/en/database/oracle/security-assessment-tool/2.2/satug/index.html#UGSAT-GUID-1E92F7C9-4FEE-4ECC-8C41-1EC441D79F53

Security Vulnerabilities and Patches Explained. 2015. Document by the Government of Canada, Communications and Security Establishment. Accessed on 5 April 2020. Retrieved from https://cyber.gc.ca/sites/default/files/publications/itsb-96-eng.pdf

Smart meters explained. 2020. Article on Uswitch site. Accessed on 15 January 2020. Retrieved from https://www.uswitch.com/gas-electricity/guides/smart-meters-explained/

Stauffer, C. 2017. 6 Reasons Why Companies Don’t Patch. Accessed on 6 March 2020. Retrieved from https://blog.automox.com/6-reasons-companies-dont-patch

The Critical Patch Update Program. 2020. Page on Oracle site. Accessed on 12 April 2020. Retrieved from https://www.oracle.com/corporate/security-practices/assurance/vulnerability/security-fixing.html

Use of Common Vulnerability Scoring System (CVSS) by Oracle. 2020. Page on Oracle site. Accessed on 6 April 2020. Retrieved from https://www.oracle.com/security-alerts/cvssscoringsystem.html

Vijay, S., Hoikka, H., Blomqvist, K. N.d. Ukraine 2015 Power Grid Cyber Attack. ELEC-E7470 Cybersecurity L – Case Study, 2-8. Document on Aalto site. Accessed on 15 April 2020. Retrieved from mycourses.aalto.fi, Academic site.

Vulnerabilities. 2020. Page on OWASP site. Accessed on 7 March 2020. Retrieved from https://owasp.org/www-community/vulnerabilities/

What is the Smart Grid?. N.d. Page on Smartgrid site. Accessed on 23 January 2020. Retrieved from https://www.smartgrid.gov/the_smart_grid/smart_grid.html

What Is Oracle Database. 2020. Summary on Oracletutorial site. Accessed on 19 March 2020. Retrieved from https://www.oracletutorial.com/getting-started/what-is-oracle-database/

What Is WannaCry ransomware?. 2020. Article on Kaspersky site. Accessed on 9 March 2020. Retrieved from https://www.kaspersky.com/resource-center/threats/ransomware-wannacry

Yantz, M. 2019. Importance of Patch Management to Avoid Business Vulnerabilities. Accessed on 31 March 2020. Retrieved from https://itsupportguys.com/importance-of-patch-management-to-avoid-business-vulnerabilities/