Automated Patch Management for B2B IaaS Environments

An initial overview

Alberto Ameglio

13 novembre 2025

Sommario

This paper proposes an initial approach to Patch Management, providing a concise theoretical overview of the current landscape of both Patch Management and Vulnerability Patch Management. The goal is to establish a basis for a future theoretical framework and to serve as supporting material for the writing of a master's thesis. In addition, a preliminary solution or methodology is outlined within the context of the Polo Strategico Nazionale (PSN)[18], focusing on B2B infrastructure. The work also includes some suggestions or references to certain tools, workflows, potential architectural developments, and technologies that could be integrated into the current proposed solution.

Indice

1 Introduction **1**

2 Standards and Best Practices for Patch Management **3**
- 2.1 IEC/ISO 62443 . . . 3
- 2.2 ISO / IEC 27002 . . . 4
- 2.3 NERC CIP-007 . . . 5
- 2.4 NIST SP 800-40 — Enterprise Patch Management . . . 5

3 Definition of Objectives **7**
- 3.1 Proof of Concept (PoC) . . . 8
- 3.2 Overview . . . 8

4 Patch Management Process **10**
- 4.1 NVD Database Discovery . . . 11
- 4.2 Continuous Compliance Management . . . 13
- 4.3 P1 — Active Environment Discovery . . . 14
- 4.4 P2 — Security Discovery & Prioritization 15
- 4.5 P3 — Patch Testing . . . 16
- 4.6 P4. Patch Deployment . . . 18
- 4.7 Summary and Residual Risk Management 19

5 Future Integration & P5: Post-Deployment Assessment **20**
- 5.1 Post-Deployment Assessment . . . 20

1 Introduction

Patch management is one of the fundamental and critical processes in the IT infrastructures. With the expansion of organizations into cloud, on-premises and hybrid environments, the number of systems requiring frequent updates has increased exponentially. Vulnerabilities, such as zero-day exploits or misconfigured libraries, can lead to serious security breaches, data exfiltration, or damage to a company's reputation. As attackers are constantly looking for weaknesses, timely patching to support software protection has become an essential activity.

A patch is a piece of code developed by software vendors to modify, enhance, or correct existing software components. Patch management involves identifying, prioritizing, acquiring, installing, and verifying patches. Nowadays, it is a key component in mitigating security and functional risks in enterprise systems. The purposes of a patch can vary, from addressing security flaws to improving functionality or introducing new features. In general, they can be classified into three categories: feature patches, functional patches, and security patches, the latter specifically intended to fix identified vulnerabilities [1]. The focus of this work is exclusively on security patches, hereinafter

referred to simply as patches.
An effective patch management process must be systematic, repeatable, and measurable, thus serving as a fundamental element of an organization's defense strategy. The research analyzed identified patch management as one of the four essential pillars of cyber defense, along with strengthening systems and applications, preventing network reconnaissance, and protecting against malware. Together, these components define an organization's ability to mitigate the risks posed by attacks originating from external interactions with the information system.
These observations are reinforced by empirical evidence presented in the candidated release of OWASP Top 10:2025[4][3], considered a widely recognized benchmark for advancing secure web application development. The report classifies this issue in the A03:2025—Software Supply Chain Failures[5]. The same category ranked first in the community Top 10 survey. Originally introduced in the 2013 Top 10 as "A9 – Use of Components with Known Vulnerabilities", its classification has since evolved to include a broader range of supply chain failures beyond known vulnerabilities. Despite this extended classification, supply chain-related issues remain challenging to identify nowadays. The category currently encompasses 11 Common Vulnerabilities and Exposures (CVEs) mapped to their corresponding Common Weakness Enumerations (CWEs).
The relevant CWEs are CWE-477: Use of obsolete features, CWE-1104: Use of unmaintained third-party components, CWE-1329: Reliance on non-upgradeable components, and CWE-1395: Dependence on vulnerable third-party components.

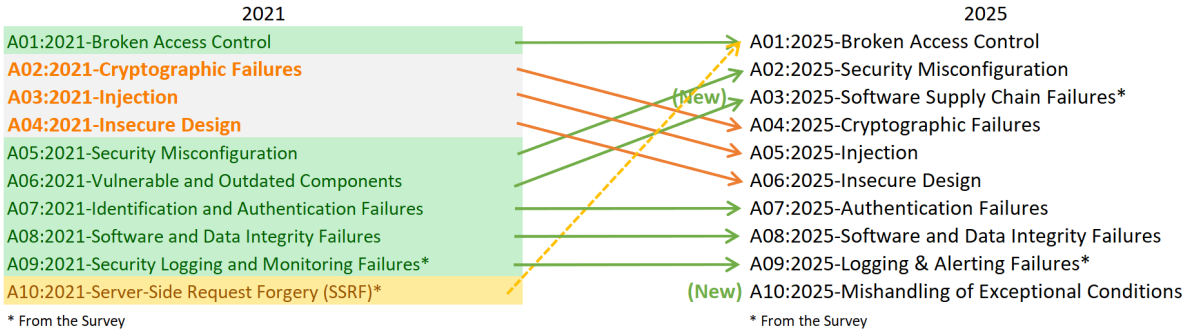

Figura 1: OWASP Top Ten for 2025
[4]

Furthermore, the Gartner Group[2] study on "Reducing Internet-Based Intrusions"[11] reveals that approximately 90% of successful cyberattacks exploit known vulnerabilities in outdated or improperly configured software. Alarmingly, in most cases, patches for these vulnerabilities were already available weeks or even months before the incidents occurred. These findings underscore the need for a proactive and continuous patch management process. Historically, patching has often been perceived as a compliance requirement or a technical best practice rather than a strategic imperative. However, the advent of Zero Trust Architectures (ZTA) has significantly changed this perspective. In architectures where the traditional network perimeter no longer exists, each resource must be independently protected and constantly audited. Within this paradigm, patch management becomes a key element in ensuring system resilience and reducing exposure to known threats. Additionally, patch management directly contributes to compliance with major security and privacy frameworks, including the EU Cybersecurity Act [7], the EU Cloud Cybersecurity Certification Scheme (EUCS) [8], and the U.S. Homeland Security Act [9].
Despite its recognized importance, patching often encounters organizational resistance due to the perceived risks of downtime and operational disruptions. This misalignment between business continuity objectives and security priorities often delays patch implementation, increasing exposure to potential threats. Therefore, patch management should be considered not just a maintenance activity, but a strategic component of enterprise risk management, integrated into both security governance and operational planning.

2 Standards and Best Practices for Patch Management

This section analyzes the main international standards and best practices that specifically address the patch management process. Specifically, it will focus on the contributions of the International Electrotechnical Commission (IEC) and the International Organization for Standardization (ISO), two of the most influential standards bodies in the field. Patch management is addressed in several of their frameworks, particularly ISO / IEC 27002 and IEC TR 62443. The latter is the reference standard adopted by the U.S. Department of Homeland Security, as described later in this section. In addition, will present the standards developed by the North American Electric Reliability Corporation (NERC) and the National Institute of Standards and Technology (NIST), both of which are widely adopted in critical infrastructure and industrial environments. [12].

2.1 IEC/ISO 62443

The IEC / ISO 62443 framework [13] emphasizes that patch management should not only focus on remediation of vulnerabilities but also ensure business continuity, traceability and controlled change management. The standard defines an organized and repeatable workflow that includes the following phases: information gathering and project planning, monitoring and evaluation, patch testing, deployment, verification, and reporting. This lifecycle approach is designed to minimize the likelihood of unplanned downtime or unintended side effects during implementation [12].

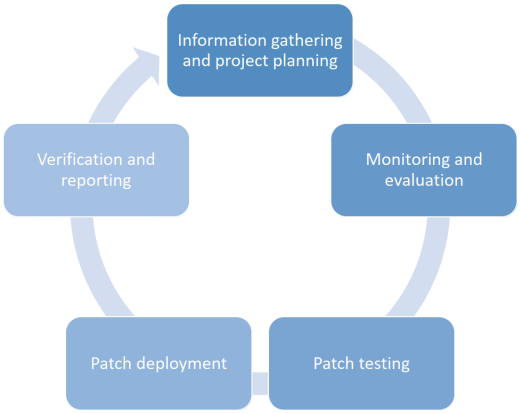

Figura 2: Patch Management lifecycle as defined in IEC 62443.

Information Gathering and Project Planning
The first phase focuses on developing a comprehensive and up-to-date inventory of industrial assets, including details such as communication interfaces, inter dependencies, and current hardware, software, and firmware versions. This inventory allows for the identification of assets potentially affected by future patch implementations. Once the inventory is completed, a project plan is formulated and submitted to corporate management for formal approval. The plan defines the business logic covering benefits, costs, expected downtime, and recovery time objectives, assigns responsibilities to stakeholders, specifies the test environment, and outlines a detailed backup and recovery strategy. In critical infrastructure contexts, this latter element is mandatory to ensure operational resilience.

Monitoring and Evaluation
This phase evaluates the applicability and potential impact of a patch before its deployment. According to IEC 62443 [13], a patch may be automatically approved if:

1. It has vendor authorization
2. It targets the correct asset version

3. It is required to mitigate a known vulnerability.

If these conditions are not met, it is essential to perform a qualitative or quantitative risk analysis to determine whether installing the patch is justified. Although the standard references several risk analysis methodologies, implementation relies on complementary standards and industry-specific practices.

Patch Testing

Patch testing is the verification phase during which both functionality and reliability are validated before deployment in production environments. The standard recommends testing only vendor-certified patches to ensure their authenticity and integrity. The testing process should evaluate system performance and stability after the update, using measurable indicators such as reliability or performance indices. Furthermore, this analysis must include detailed documentation of the installation procedure and the defined rollback strategy. Although IEC 62443 requires such a strategy, it allows for flexibility in choosing the specific rollback mechanisms most appropriate for the operating environment.

Patch Deployment

Deployment is performed only after successful testing and formal authorization. IEC 62443 divides this stage into several controlled activities: stakeholder notification, preparation of the production environment, patch installation (manual or automated), and scheduling deployment windows in alignment with business operations. Subsequent to implementation, the system must be validated to confirm the correct mitigation of identified vulnerabilities and the stability of the environment once updated. The standard recommends adopting verification procedures based on the organization's operational and regulatory context, ensuring traceability and continuous improvement of the patch management process.

2.2 ISO / IEC 27002

The ISO / IEC 27002 standard [14] provides a comprehensive framework for Information Security Management Systems (ISMS), offering guidelines for implementing and maintaining effective security controls, including those related to patch management, across enterprise and critical infrastructure environments. As part of the ISO / IEC 27000 family, it is internationally recognized and widely adopted across diverse industries to support the establishment of structured and consistent information security practices.

The patch management guidelines described in ISO / IEC 27002 have influenced subsequent regulatory developments, particularly those driven by the European Union Agency for Network and Information Security (ENISA). In its 2014 report [15], ENISA built upon ISO principles to define standardized requirements for patch management within industrial control and critical infrastructure systems, reflecting the increasing necessity of harmonized patching procedures in security-sensitive contexts.

The principles derived from the ISO framework emphasize both operational resilience and procedural integrity, ensuring that patch implementations improve security without compromising system availability or performance. These principles can be summarized as follows:

- Updating must preserve system functionality and ensure that patches do not negatively impact the operational behavior of target systems.
- If service interruptions are unavoidable, downtime must be minimized through careful planning or pre-established maintenance windows.
- The infrastructure should incorporate redundancy mechanisms, prioritizing patch deployment on passive or non-critical redundant components first. These resources must be updated and tested separately before large-scale deployment to reduce the risk of service disruption.
- Patch management must be coordinated by a dedicated working group responsible for:
  1. Evaluating the potential cybersecurity implications of each patch

2. Defining deployment strategies and schedules
3. Executing, verifying, and documenting the various phases of patch implementation across all operational resources

Compared to IEC TR 62443, ISO / IEC 27002 offers less detailed technical requirements but places greater emphasis on governance and organizational accountability. Its objective is to establish a controlled and tested process, aligned with broader cybersecurity and risk management frameworks.

2.3 NERC CIP-007

While the standards discussed in the previous sections provide broad, cross-industry frameworks for patch management, the North American Electric Reliability Corporation (NERC) introduces a sector-specific perspective through its Critical Infrastructure Protection (CIP) standards. NERC is considered the primary regulatory authority responsible for maintaining the reliability and security of the North American power system, issuing internationally recognized guidelines, particularly relevant to the protection of critical infrastructure.
Among these, NERC CIP-007 (NERC, 2015) [16] defines the technical and procedural requirements for patch management within the electric utility domain, with particular emphasis on Remote Terminal Units (RTUs) and other field-deployed control devices. While the standard is primarily tailored to electrical grid control systems, many of its principles are applicable to broader industrial control and automation environments.
Similar to IEC 62443[13], NERC CIP-007 emphasizes the importance of comprehensive documentation for each phase of the lifecycle. This requirement improves traceability and facilitates root cause analysis in the event of deployment errors or post-update anomalies.
A distinctive contribution of NERC CIP-007 is the introduction of the trusted sources principle, mandating that patches be obtained exclusively from certified and verified vendors. This measure ensures the authenticity and integrity of software updates, mitigating supply chain risks and limiting the introduction of malicious or compromised patches into critical environments.

2.4 NIST SP 800-40 — Enterprise Patch Management

The NIST Special Publication 800-40 Revision 4 [10] focuses on the development of effective enterprise patch management processes, emphasizing the balance between operational stability, risk mitigation, and organizational coordination. The heterogeneous growth of IT systems, including cloud infrastructures,Internet of Things systems, and virtualized environments, has significantly increased the complexity of security patch management. As a result, patching has evolved from a technical activity to a strategic business function requiring structured planning, prioritization, and interdepartmental collaboration. Enterprise patch management often reflects the tension between maintaining business continuity and ensuring timely vulnerability remediation.
Enterprise patch management often reflects the tension between maintaining business continuity and ensuring timely vulnerability remediation. Organizations must therefore balance rapid patch deployment, essential for minimizing exposure, with comprehensive testing to guarantee operational reliability. In contemporary distributed environments, reactive patching approaches are no longer sufficient. Instead, organizations should adopt proactive strategies emphasizing preventive maintenance and systemic resilience.
This paradigm shift reframes patching-related interruptions as controlled interventions that safeguard operational integrity and prevent uncontrolled incidents such as data breaches or ransomware infections. According to NIST, effective patch management should adhere to some key principles:

- **Preparation over perfection:** Problems are inevitable; preparedness and resilience significantly reduce their impact.
- **Simplified decision-making:** Predefined and standardized procedures enable consistent and rapid responses to emerging vulnerabilities.

- **Automation:** They are essential to ensure scalability, speed and accuracy, especially in complex environments.
- **Continuous improvement:** Incremental refinements strengthen long-term organizational security and maturity.

To apply these principles, NIST recommends that organizations:

1. Minimize patch-related disruptions by reducing vulnerabilities through system hardening, acquiring secure software, and adopting managed or containerized environments.
2. Maintain constantly updated software and asset inventories, preferably automated, to ensure risk assessment and patch prioritization.
3. Define risk response scenarios, distinguishing between routine, emergency, and non-patchable assets, ensuring preparedness in all operational contexts.

Through these measures, management becomes a proactive discipline that improves organizational resilience and strengthens the integrity of enterprise systems in an increasingly hostile landscape.

Risk Responses

According to the principles outlined above, organizations must adopt an adaptive approach to vulnerability management. While patching is the most effective and direct mitigation technique, it is only one possible risk response strategy. According to the document, four main categories of risk responses can be adopted when addressing software vulnerabilities:

1. **Accept** — The organization acknowledges the residual risk and relies on compensating controls or alternative safeguards when the potential impact or likelihood of exploitation is minimal.
2. **Mitigate** — The organization must actively reduce risk by addressing vulnerabilities through updates or implementing additional security measures such as segmentation, isolation, or access control.
3. **Transfer** — The risk is partially delegated to third parties, for example, through system security insurance or by outsourcing patch management to managed service providers.
4. **Avoid** — The source of risk is eliminated entirely, for instance by decommissioning vulnerable systems or discontinuing insecure software components.

Among these strategies, timely patching remains the most effective mechanism, as it directly reduces the attack surface and restores confidence in affected assets.

Preparation and Deployment

Effective patch deployment requires structured planning and precise execution. These preparation phases include several key activities that, taken together, determine the success and reliability of the patch management process:

- **Prioritization:** Patches should be categorized based on the severity of the vulnerabilities found and the criticality of the assets involved, ensuring that high-risk systems receive priority.
- **Scheduling:** Implementation should be integrated into enterprise workflows and change management to align with operational cycles and minimize disruptions.
- **Acquisition:** Patches should be obtained exclusively from verified sources, such as vendor repositories or secure internal channels.
- **Validation:** Authenticity and integrity should be confirmed, ideally through automated cryptographic verification, to prevent tampering or counterfeit updates.

- **Testing:** Controlled pre-deployment testing is essential to detect compatibility issues, regressions, or performance degradation before deployment in production.

Once the preparation phase is complete, the deployment process entails the operational rollout of the patch through several structured steps:

- Distributing the patch to the targeted systems, either automatically, manually, or through vendor controlled channels (e.g., cloud-delivered updates).
- Installing the patch, which may require elevated privileges, including a system reboot or configuration changes.
- Applying necessary configuration or state changes to ensure that the patch is fully effective.
- Monitoring for post-deployment anomalies and addressing any adverse effects through rollback or restoration from validated backups.

A robust patch management must integrate ongoing validation, monitoring, and feedback mechanisms. A combination of strategic planning and automated procedures can ensure effective patch lifecycle management, ensuring a resilient and efficient approach that supports both cybersecurity and operational objectives.

3 Definition of Objectives

As discussed in previous chapters, there is currently no universally accepted academic definition of security patches. For the purposes of this work, we adopt the definition proposed in the literature review by Dissanayake et al. [24]:

> **Definition:** "Software security patch management is a multifaceted process of identifying, acquiring, testing, installing, and verifying security patches for software products and systems."

Within the context of the PSN[18], the main challenge lies in ensuring that the application of security patches does not interfere with clients' production environments. Accordingly, the initial phase of this project focuses on gaining an understanding of the current state of managed resources. At present, no unified overview of PSN-managed assets exists, and assessing their status requires direct access to each client tenant to:

- Inventory all resources, e.g . virtual machines (VMs);
- Verify operating system versions, installed applications, and corresponding patch levels;
- Identify patches that are applied, pending, or suspended.

Patch deployment is currently carried out primarily through manual or on-demand operations, often following client requests, or via Azure Update Manager[19] for certain Windows resources. While this integration may include most PSN customers, the process remains undefined for other cloud service providers (CSP), as the analysis has so far been limited to Azure. In this initial proposal, Azure will therefore serve as the reference CSP, with the long-term goal of designing a CSP-independent process that supports the automated discovery, inventory, and management of resources across heterogeneous environments.
The lack of a centralized and comprehensive inventory currently prevents an accurate estimate of both the number and criticality of patches requiring prioritization, and a more strategic identification for optimal patching. Accordingly, the primary objective is to design and implement a centralized system capable of providing:

- A unified and continuously updated view of all PSN-managed assets;
- The number and status of pending or unpatched security updates per resource and tenant;

- A foundation for automated and risk-aware patch deployment, minimizing disruptions to client operations.

Building upon this approach, a preliminary working definition of Security Patch Management (SPM) for the PSN in this initial phase can be articulated as follows:

> "Security Patch Management for PSN consists in achieving comprehensive visibility into the health of all managed assets through a centralized system that enables automated identification, acquisition, testing, installation, and verification of third-party security patches, without compromising the operational stability of clients' production environments."

This definition intentionally excludes the assessment of security risk, which will be addressed in subsequent phases. The immediate priority is to establish visibility, traceability, and operational control for the patch management process, thus laying the foundation for more advanced security assessments, the definition of roles, and the long-term evolution of the PSN ecosystem. In the current PSN context, the term security patch refers specifically to updates released by vendors to fix vulnerabilities in software in environments sold to customers. The SPM process begins when the next patch is released, thus initiating an evaluation and deployment activity. The main challenge at this stage is determining which patches can be safely applied without impacting the production environment. In particular, the goal is to identify updates that the PSN can autonomously deploy without requiring client or vendor intervention, thereby reducing exposure while preserving overall system integrity and availability.

3.1 Proof of Concept (PoC)

This section presents the Proof of Concept (PoC) for a potential Security Patch Management (SPM) model designed for integration within the PSN environment. The work outlines a proposed patch management process for virtual machines operating in a multi-tenant B2B infrastructure, drawing on insights from previously analyzed use cases [20][21][22][23][24][25] and from the standards discussed in Section 2.
The objective is to formalize the process through a set of workflows that describe its main stages and operational activities. These workflows serve as a foundation for the design and potential development of tools or services currently unavailable within the PSN ecosystem.
A pragmatic design philosophy is adopted: if an existing tool satisfies at least 80% of the functional requirements, it is considered suitable for integration. This approach adheres to the DRY (Don't Repeat Yourself) principle, which discourages the development of existing systems. The remaining 20% of unmet functionality must, however, be carefully evaluated in accordance with the YAGNI principle (You Aren't Gonna Need It), as unnecessary extensions can introduce compliance issues and increase the overall attack surface.
In this exploratory phase, several tools are introduced as illustrative examples addressing specific parts of the proposed workflow. These should be regarded as reference points or conceptual starting frameworks for future research and development. At this stage, the KISS principle (Keep It Simple, Stupid) is intentionally set aside; simplification, rationalization, and component selection will instead be prioritized in subsequent refinement phases.

3.2 Overview

Building on the considerations outlined above, this subsection presents the proposed Security Patch Management architecture. The system design is structured around two core components:

1. **Master Server** — Deployed within the insourcing control tenant, it acts as a centralized repository and provisioning system. Manages lifecycle environments, content monitoring, updating the National Vulnerability Database[34] and managing Proxy Servers.

2. **Proxy Server (Smart Proxy)** — Installed within each tenant or customer infrastructure, operating directly inside the networks where patch management tasks are executed. This component plays a pivotal role in operational orchestration and local patch deployment.

This satellite-based architecture takes inspiration from established enterprise solutions such as Red Hat Satellite[26], Foreman with Katello[27], Spacewalk[28], and Orcharino[29].
The Master Server serves as the central control node, offering an aggregated and real-time view of all managed resources. Conceptually, it can be compared to a monitoring and analytics module, akin to a Security Information and Event Management (SIEM) system, suggesting possible future integrations within the PSN framework. It also enforces strict access control and tenant segregation, in alignment with the principles of separation of privileges and separation of duties, as outlined in the *"PSN_LLD SPC Governance Model_v1.0"*. This document establishes the operational guidelines for managing resources within the Secure Public Cloud, emphasizing non-persistent privileges and the absence of permanent access points.
Additionally, the Master Server is responsible for distributing updates, operational rules, and automation policies to the Proxy Servers, while supporting customized configurations and differentiated security levels for each tenant. The **Smart Proxy** is responsible for the operational management of the patch lifecycle, including configuration, deployment, monitoring, and data collection, which then updates the Master Server's status.

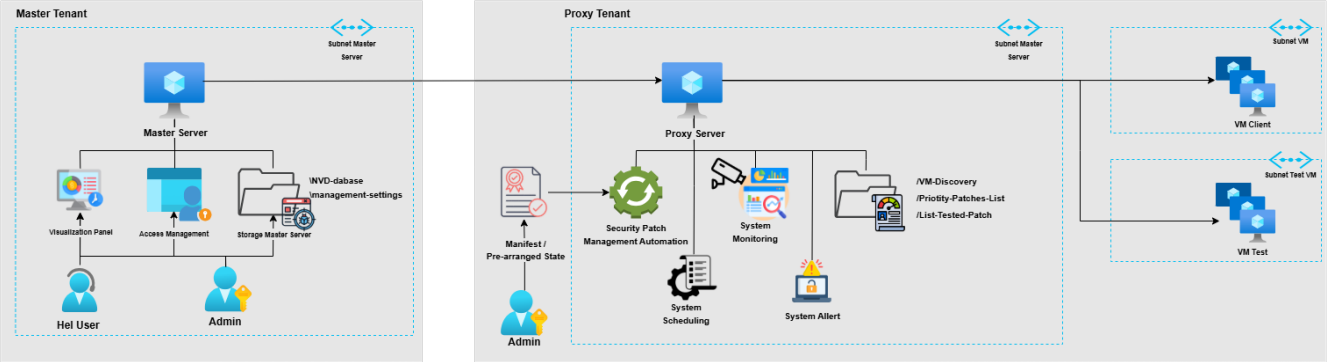

Figura 3: High-Level Design of the Security Patch Management System

As illustrated in Figure 3, communication between the Master Server and Smart Proxies occurs through secure, private, peer-to-peer channels. This ensures that all operational activities remain confined within the tenant's infrastructure, where the Proxy is deployed with the necessary privileges to manage target virtual machines (VMs) and a dedicated test environment. The test environment must remain logically and operationally isolated from production systems. The Master Server never interacts directly with these environments; instead, all communication is mediated exclusively through the Smart Proxy. Each Master Server can manage multiple Smart Proxies, forming a 1:N topology, where cross-tenant communication is strictly segregated in accordance with PSN security guidelines. The Smart Proxy must provide a high degree of configurability during both development and operational phases, allowing administrators to adapt workflows to specific organizational and security requirements. Tools such as Puppet[32] and the Elastic Stack[31] offer declarative automation and monitoring capabilities well-suited for this purpose. Alternatively, orchestration platforms like ManageIQ[30] can support comprehensive governance, resource lifecycle management, and integration with broader PSN operational processes.

In summary, the essential objective of this project is to identify all required security patches, automatically install those that can be safely applied without causing operational disruptions, and systematically catalogue the remaining ones. From a strategic perspective, the overarching goal is to ensure that this process is executed with minimal resource utilization, optimizing both efficiency and scalability within the PSN environment.

4 Patch Management Process

This section describes the process adopted for managing security patches. The overall design and operational logic are inspired by the ISO/IEC 62443 [13] standard discussed in Section 2.1.
As specified in the initial requirements, the management process begins only after the official release of a patch by the concerned vendor. Consequently, the process does not encompass the reporting, request, development, or validation phases of the patch itself. From this starting point, the workflow is divided into a set of components that collectively define the Patch Management process. Each component operates according to configurable time intervals, which can be adapted to the specific needs of the system. These parameters may be applied uniformly across all customer tenants or customized based on service-level requirements and security policies. Activity scheduling can be automated using standard tools such as `cron` [33] or equivalent systems integrated within the Smart Proxy infrastructure.

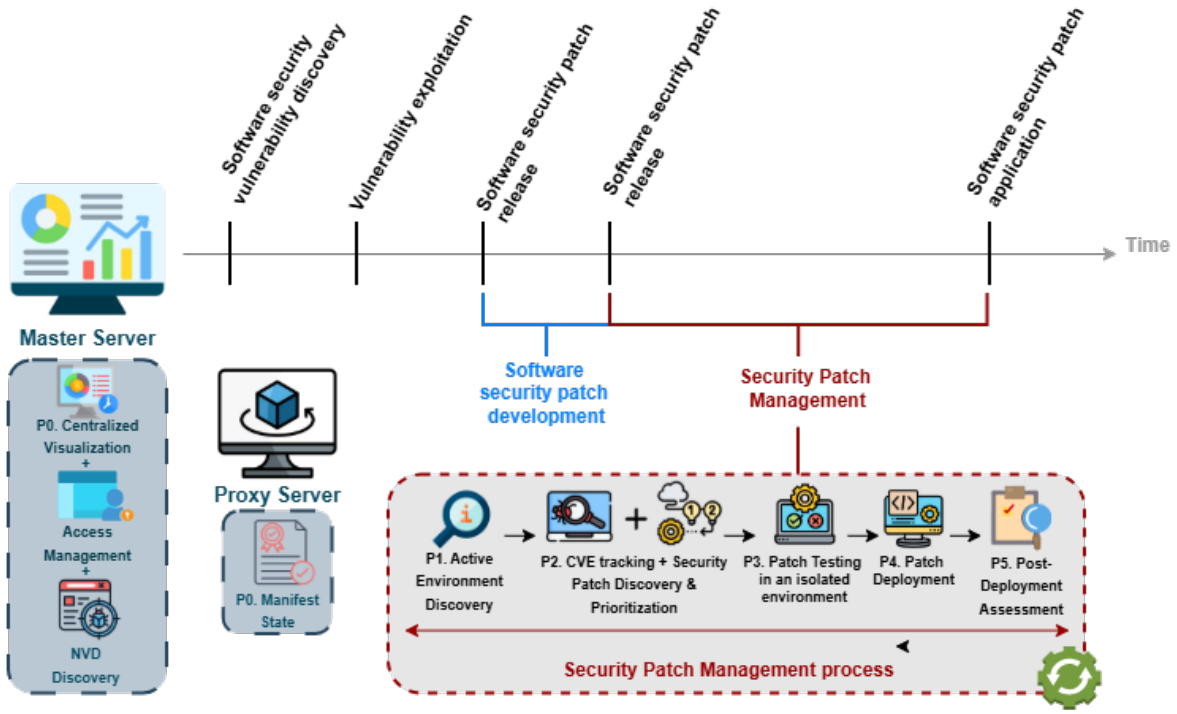

Figura 4: Overview of the Patch Management Process

The Security Patch Management process [Figure 4] consists of four main components that define its operational sequence and associated decision logic:

- **P1 — Active Environment Discovery**
- **P2 — Security Discovery & Prioritization**
- **P3 — Patch Testing**
- **P4 — Patch Deployment**

Additionally, the overall framework includes three complementary elements:

- **Continuous Compliance Management** – the module, upon completion of each Active Environment Discovery cycle, compares the formal state defined by the administrator during the preliminary phase, specific to each Proxy, with the state detected by the process.
- **CVE Tracking / NVD Discovery** — a module integrated within the Master Server, responsible for maintaining the National Vulnerability Database (NVD DB).
- **P5 — Post Deployment Assessment** — an optional phase for post-deployment evaluation [Section 5.1].

Notes on Status and Functional Dependencies

1. **P5 (Post Deployment Assessment)** — At this preliminary stage of the project, P5 has been temporarily excluded. The assessment conducted indicated that it is not currently a priority nor a direct responsibility of the PSN. It may be reinstated in future iterations if deemed essential or required for compliance purposes.
2. **CVE Tracking / NVD Discovery** — CVE tracking is performed through the NVD Discovery module integrated within the Master Server. Its output, the NVD DB, serves as an input to the vulnerability prioritization process (P2).
3. **Process Dependencies**
   - **P2** depends on the output of NVD Discovery. Without updated NVD data, accurate and timely vulnerability prioritization cannot be ensured.
   - **P1** and **NVD Discovery** operate independently and can be scheduled separately. Their execution frequency may be optimized according to bandwidth availability and network conditions.
   - **P1 → P2 → P3 → P4** form a mandatory sequential chain. P3 can only start after P2 has completed, also P4 can only proceed after P3 has successfully completed. Although P2, P3, and P4 may be scheduled within predefined operational or bandwidth windows, their logical dependency remains strict: a new P4 cycle cannot be executed without repeating P2 and P3 for the corresponding set of patches. This dependency is illustrated in the workflow diagram, where the NVD DB represents the output of the NVD Discovery component on the Master Server.

From this phase, **two fundamental concepts emerge and must be emphasized**, as they represent key elements in the design of the entire system. The component identified as **P2 should be regarded as the strategic core of the project**, as it is expected to enable an efficient management process by minimizing both resource utilization and the time required for subsequent phases. This optimization also contributes to reducing the likelihood of operational errors during later stages. Conversely, **P3 constitutes the most critical and delicate stage of the workflow**, as it determines which patches and related resources are authorized to interact with the customers' operating environments.

4.1 NVD Database Discovery

The NVD Database Discovery module ensures that the vulnerability knowledge base is continuously updated and remains intact. This component operates autonomously and on a scheduled basis, synchronizing the local NVD (National Vulnerability Database) with official sources maintained by entities such as NIST, ensuring that all CVE–software correlation analyses performed by the Security Discovery & Prioritization (P2) module are based on up-to-date, complete, and verified data.
The module resides within the Master Server, as the data obtained is commonly used by all Proxy Servers, and its execution is independent of the Patch Management operational flow. The architecture can modify this point if necessary to modulate execution based on load balancing or network traffic management strategies.

At startup, the system checks the status of the local NVD database by checking:

- table consistency,
- data integrity,
- timestamp of the last update.

This preliminary check allows three main scenarios to be identified:

1. Empty database: requires complete initialization;
2. Outdated database: requires incremental update;
3. Updated database: does not require immediate action.

Download and update
If the database is empty or out of date, the system starts a full or incremental download of the latest available version of the NVD from official sources, using secure HTTPS connections. The official NVD database contains structured information on hundreds of thousands of known vulnerabilities, including:

- CVE identifiers;
- detailed descriptions of vulnerabilities;
- CVSS scores (base and temporal);
- attack vectors and exploit complexity;
- affected software products, identified by CPE[35] (Common Platform Enumeration);
- vulnerable versions with specific ranges;
- availability of official patches and documented workarounds;
- references to public exploits.

The amount of data requires high bandwidth and variable download times, which must be taken into account when planning operations.
Upon completion of the download, the system performs rigorous checks to ensure data integrity and authenticity:

- File hash checksums;
- Digital signature validation (GPG/PGP)[36];
- Sanity checks on downloaded data formats.

This verification process is considered mandatory whenever the system interacts with material from external sources, including the download of security patches in the P2 module presented below, ensuring that all information used to make patching decisions is reliable and secure.

4.2 Continuous Compliance Management

For effective patch management, it is essential to adopt a formal and granular approach to resource management, enabling the selection of the most appropriate methodology on a case-by-case basis. It is therefore necessary to define, for each tenant, the levels of priority or exclusion associated with specific resources, in order to comply with bilateral agreements that may be not standardized or subject to modification throughout the process lifecycle. Such flexibility requires the adoption of configurable and differentiated models. An effective approach involves the integration of a Continuous Compliance Management (CCM) system, which takes as input a preliminary definition of the resources under management. This definition formally establishes the correspondence that must be verified over time. As illustrated in the figure [3], the creation of this definition falls under the responsibility of an administrator possessing the necessary authorizations, permissions, and operational resources. Moreover, the process must be placed in the **P0 state**, a preliminary, independent, and isolated stage for each tenant, as it constitutes a prerequisite for the proper functioning of the Smart Proxy.
The detailed structure of this definition is beyond the scope of this preliminary phase and will be addressed in later stages, maintaining flexibility in its contents and required fields for each proxy as a guiding principle.
This process falls within the broader domain of compliance management, an area that has fostered the emergence of paradigms such as Infrastructure as Code (IaC)[54] and Policy as Code (PaC)[55], which enable the automation of rule verification and comparison against initial requirements. The proposed approach integrates with the Active Environment Discovery, which reflects in real time the state of the detected resources.

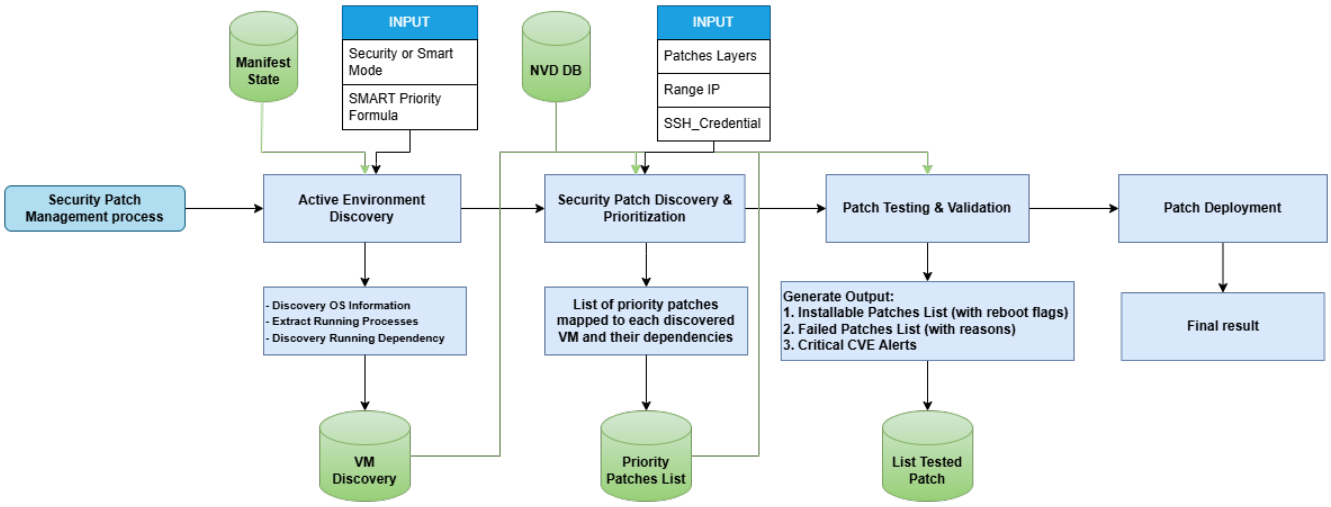

Figura 5: Workflow Management Process

As shown in the figure [5], this environment provides the system's current state, which is then compared with the declaration at the end of the process. Continuous Compliance Management can thus be defined as an ongoing verification process that ensures alignment between managed resources and a formal reference, **manifest state** or **compliance standard**. The manifest represents the Desired State, i.e., the expected system configuration (for instance, the list of virtual machines contractually defined for a specific tenant). The compliance component periodically compares this desired state with the one detected by the Active Environment Discovery, identifying any discrepancies. The process must include mechanisms for **drift detection** and **inventory reconciliation**, which are responsible for identifying differences between the manifest and the actual infrastructure state. A drift occurs when a system element diverges from expected parameters; such deviations are reported through Assertions, ensuring consistency between the contractual and operational states. This enables timely detection and correction of deviations, thereby improving data quality and overall system reliability.

The adoption of the Policy as Code paradigm allows compliance rules to be represented as executable code, which can be integrated into workflows and the CCM system. Compliance policies define a set of validation rules that specify the expected conditions in the detected system state, for example: "all existing VMs must belong to the list defined in the contractual manifest." This approach ensures transparency, reproducibility of control rules, reduced need for manual verification, and greater process standardization.
The integration of Continuous Compliance Management with the discovery thus enables effective governance of the overall process. However, the comparison between the manifest and the detected state must occur automatically at the end of the discovery phase, whereas any modification or rectification of the manifest must be performed exclusively and manually by the administrator. This preserves the principle of manifest immutability, ensuring that it cannot be altered by automated tools. As a result, the manifest may remain unchanged throughout the Smart Proxy's lifecycle or may not be updated as a result of assertions generated by the verification module. Decisions regarding Inventory Reconciliation and evaluations related to such updates will be addressed at a later stage and remain the sole responsibility of the administrator.

4.3 P1 — Active Environment Discovery

The P1 module represents the initial phase of the Security Patch Management process and aims to automatically identify, analyze, and classify all virtual machines present in the IP ranges defined within the infrastructure. The main component of this phase is the Proxy Server, responsible for discovering the environment and collecting the technical information necessary for subsequent security analysis.
The operation begins with the launch of the Discovery Agent, which uses an agentless methodology, i.e., it does not require the installation of dedicated software on the systems to be monitored. To this end, tools such as Ansible[37] can be used, which allow systematic scanning of IP addresses and verification of connectivity:

- via SSH for Linux systems;
- via REST API or WinRM[43] for Windows systems.

Once the connection is established, the system collects preliminary information on:

- operating system and kernel version;
- hardware architecture;
- network configuration;
- status of the main system packages.

Subsequently, the workflow branches into paths optimized for each platform (Windows or Linux), adapting the query and analysis methods to the specificities of the environment. For Linux systems, further specialized paths are also managed depending on the package manager used (e.g., YUM[38], DNF[39], APT[40], Zypper[41], Pacman[42]).
Data Structuring and Patches Layers
The scan results are organized into a multi-level hierarchical structure, called Patches Layers, which allows the detected elements to be classified according to their criticality and influence on the operating environment. The preliminary version of the hierarchy is as follows:

- **L1** – Kernel / Operating System
- **L2** – Firmware / Drivers
- **L3** – Security & Management Components
- **L4** – Core Libraries
- **L5** – Runtime / Middleware

- **L6** – System Applications
- **L7** – Third-party / Browser Components

This classification aims to establish update priorities and provide structured support for the P2 — Security Discovery & Prioritization process.

Integration with repositories and data management

The system must integrate connection mechanisms to the main patch repositories:

- Windows Update Services (WUS)[44] or WSUS for Microsoft environments[45];
- official repositories of Linux distributions or corporate mirrors for open-source environments.

This makes it possible to check for updates in real time and identify any missing security patches. All the information collected is structured in JSON[46] format, which must be updatable, modifiable, and manageable over time to maintain a consistent and historical snapshot of the environment.

The information collected and detected by the system is subsequently transmitted and synchronized to the Continuous Compliance Management module, which compares it with the Manifest State provided as input to the module, as illustrated in the figure [5] and previously described in the preceding section [4.2].

Once the scan is complete, the system generates a summary report containing all the information collected, which is then synchronized with the Master Server.

The process is designed to be iterative and autonomous: it automatically checks for new VMs to analyze before completing the scan cycle and can be scheduled for periodic or on-demand execution.

The infrastructure must ensure the security of communications and data through:

- authentication via SSH keys;
- secure credential management via encrypted vaults;
- encryption of data in transit;
- complete logging of all operations performed, to ensure traceability and auditability.

4.4 P2 — Security Discovery & Prioritization

The P2 process is the strategic core of the Security Patch Management system. In this phase, the outputs from P1 and the NVD Discovery module are integrated with the aim of identifying, correlating, and prioritizing the security patches available for each identified virtual machine. The primary objective is not only to identify the most urgent patches, but also to establish an optimal order in which to process them, thus generating a priority list consistent with the risk, criticality, and operational stability of the environment.

The P2 process architecture is based on the integration of two sources:

1. **VM Discovery Database** — populated by the P1 module and containing details of each detected asset, including operating system, version, installed packages, and patch level.
2. **NVD** — fed by the NVD Discovery component, which collects up-to-date data on known vulnerabilities (CVE), severity scores (CVSS)[47], exploitability level, and risk indicators.

The system matches the version data of the resources on each VM with the known vulnerabilities recorded in the NVD database, identifying missing or obsolete patches for each machine. Prioritization approaches The prioritization mechanism operates in two distinct modes, which can be selected based on company policies and the level of risk tolerated:

1. **Security Mode** - A defensive and conservative mode, in which patches are ordered exclusively on the basis of security criteria, without considering the operational impact. The main parameters evaluated are:

- CVSS score (vulnerability severity);
- availability of public or actively exploited exploits;
- criticality of the asset involved (e.g., production servers or core infrastructure).

This mode is recommended for sensitive contexts or where strict security policies are applied.

2. **Smart Mode** - A more adaptive and contextual mode, which introduces a multi-factor prioritization algorithm. In addition to risk indicators, it takes into account:
   - active processes and critical services detected in phase P1;
   - functional dependencies between components and services;
   - layer classification defined in the Patches Layers structure (L1–L7);
   - history and stability of previous patches;
   - estimated impact on business continuity.

   The goal of Smart Mode is to maximize security without compromising service availability, allowing for a more balanced approach between risk and business continuity.

The output of the process is a data structure (e.g., JSON) called Priority Patches List, which contains:

- the list of detected patches;
- the priority level assigned, defined by its position in the list;
- the risk score;
- references to associated CVEs;
- mapping with the affected virtual machines.

During this phase, the status indicators in the Master Server, which maintains a high-level overview of the infrastructure, are also updated; for example:

- total number of patches identified;
- machines involved;
- distribution of criticality levels.

However, the Master Server does not process or use the priority list (managed locally by the P2 component), but merely records the global indicators for monitoring and reporting purposes.
This synergistic integration between discovery and vulnerability intelligence allows you to move from a reactive and indiscriminate approach to a proactive, risk-based remediation strategy. Remediation resources are thus allocated in a targeted manner to vulnerabilities that pose the greatest risk of problems for the infrastructure, optimizing security investments and reducing the risk of operational downtime.

4.5 P3 — Patch Testing

The P3 process continues the SPM operational chain, transforming the Priority Patches List produced by the P2 module into a verified remediation plan ready for deployment. This component is a critical point in the entire process, as it ensures patch quality assurance through systematic and automated testing, preventing service interruptions, functional regressions, and software incompatibilities.
The system starts the process by loading the Priority Patches List for each target VM.

- If the list is empty, the module enters an active waiting state (non-blocking polling), ensuring responsiveness without unnecessary consumption of resources.
- If there are patches to be validated, the system retrieves the complete details of the VM configuration, including:
  - sizing parameters (CPU, RAM, storage);
  - network configuration;
  - storage layout;
  - mapping of application dependencies.

This information, acquired in phase P1, allows the test environment to be faithfully reconstructed.
The technological heart of P3 lies in the creation of an isolated test environment, obtained through VM Snapshot or bit-perfect cloning (e.g., Azure VM Snapshot)[48]. This process generates an exact copy of the target virtual machine, preserving:

- the state of the operating system,
- application configurations,
- local data.

The test environment must be isolated from the production network, allowing realistic testing to be conducted without any risk of contamination or disruption to critical services.
Before installing patches, the system initializes a monitoring module, based on tools such as Azure Monitor[49], Zabbix[50], Nagios[51], or Prometheus[52], with the aim of defining a behavioral baseline for the VM.
Metrics are collected on:

- CPU, memory, and disk usage;
- network throughput and application latency;
- key service performance.

These values will serve as the reference point for post-patching evaluation.
The testing process proceeds by iterating through the Priority Patches List in descending order of priority, ensuring that the most critical vulnerabilities are tested first.
For each patch:

1. The system automatically orchestrates installation on the clone VM.
2. A configurable stabilization period is allowed to pass, to allow applications to complete any post-installation operations.
3. The system continuously compares the monitoring metrics with the baseline, identifying:
   - resource usage spikes,
   - memory leaks,
   - performance degradation,
   - errors in system or application logs.

At the same time, a suite of functional and regression tests specific to the application hosted on the VM is performed, based on the information gathered during the discovery phase. This verification ensures that:

- operational functionality remains unchanged;
- APIs remain compatible with dependent systems;
- no version conflicts or breaking changes arise due to the patch.

The validation decision-making process combines multiple signals:

- compliance of metrics with the baseline within predefined thresholds;
- complete passing of the test suite without regressions;
- absence of critical errors in system and application logs;
- operational stability during the observation period.

If all criteria are met, the patch is marked as approved and made available for deployment in production (P4). Otherwise, the system generates a fallback recommendation report containing:

- details of the anomalies found,
- estimate of the potential impact on the system,
- suggestions for rollback management or alternative patches.

An additional parameter managed by P3 concerns the need for reboots: if a patch requires the system to be restarted, this information is noted and transmitted to the prioritization module (P2) to update the priority list structure, allowing for coordinated planning and minimizing downtime, also in collaboration with the customer or the data center's operating policies.

Once the test cycle is complete:

- the system automatically cleans up the VM clone to free up resources;
- it generates a detailed validation report, documenting the outcome of each patch (approved, failed, under observation) and the quantitative metrics collected;
- it updates the List Tested Patch database, ensuring complete traceability from identification to validation.

The aggregated information is then synchronized with the Master Server, keeping the overall status of the infrastructure up to date and providing global visibility on the progress of the patching cycle.
The combination of automated testing in cloned environments and behavioral monitoring allows us to maintain high security standards and operational reliability. This approach dramatically reduces the MTTR (mean time to remediation) for critical vulnerabilities, while limiting the risk of disruptions caused by faulty patches.

4.6 P4. Patch Deployment

The Patch Deployment module represents the final and operational phase of the Patch Management framework, translating the patches validated in the P3 module into a controlled, secure, and traceable distribution process on production virtual machines. This phase is designed to ensure an optimal balance between rapid remediation and operational stability through the adoption of enterprise-grade deployment patterns and automated snapshot, rollback, and post-installation validation mechanisms. The goal is to ensure that each installation can be verified independently and, in case of problems, quickly restored without permanent impact on critical services.
The process is based on the List Tested Patch, i.e., the list of patches that have successfully passed all functional and performance validation criteria during the P3 module.
The deployment system adopts a rolling approach:

- Patches are installed on a limited subset of VMs at a time.
- After each wave of installations, stability and consistency checks are performed.
- Only after passing these checks does deployment continue to the next group.

This approach significantly reduces the blast radius, i.e., the scope of the potential negative effects of a faulty patch. A connection draining mechanism is used to ensure a graceful shutdown of services: active connections are completed, ongoing transactions are closed, and clients are disconnected in an orderly manner, avoiding abrupt interruptions or data corruption.
Before each installation, pre-installation checks are also performed to verify:

- disk space availability,
- dependency compatibility,
- absence of software conflicts,
- status of critical services,
- and whether or not a reboot can be performed based on agreed maintenance policies.

If not, the system automatically schedules the patch for a later operating window, minimizing the impact on production activities.
The installation is managed by a system that installs packages with optimized parameters, capturing output, return codes, and detailed logs for each VM. At the end of each installation process, the system must compare the installed version with the expected one to ensure proper production. Any anomalies or rollbacks are automatically reported and logged in the central monitoring system.
Once the deployment cycle is complete, the system generates detailed reports for each VM, containing:

- the final status of the installation,
- any reboots performed,
- problems detected,
- and completion timestamps.

These reports are sent to the Master Server, which updates the overall status of the infrastructure, ensuring end-to-end traceability of the entire patch management process.

4.7 Summary and Residual Risk Management

- **P1 (Active Environment Discovery)**: discovers and maps the state of the environment.
- **P2 (Security Discovery & Prioritization)**: identifies available patches and establishes application priorities.
- **P3 (Patch Validation)**: tests patches in a controlled manner and generates the final list of secure and compatible patches.
- **P4 (Patch Deployment)**: installs only validated patches, ensuring security, rollback, and traceability.

It is important to note that not all identified patches are necessarily installed: some may be incompatible or require external interventions that cannot be managed by the system. In such cases, the framework must:

1. Inform the customer of the status of missing patches and the technical reasons preventing their application; Manage the residual risk according to NIST SP 800-40[10][**??**] and SP 800-30 guidelines[53], adopting one of the following approaches:
    - **Accept**: accept the risk and monitor the VM more closely;
    - **Mitigate**: implement compensating measures or increase surveillance;
    - **Transfer**: delegate resolution (e.g., software compatibility) to third parties or the customer;
    - **Avoid**: decommission or isolate the asset that is no longer needed, reducing the attack surface.

This model enables informed and structured risk management, maximizing the overall security of the operating environment and ensuring that every decision is documented, justified, and traceable.

5 Future Integration & P5: Post-Deployment Assessment

Up to this point, the entire Patch Management framework has been based on two fundamental assumptions:

1. that vulnerability assessment tools have produced accurate and reliable results;
2. that the security patch applied has effectively resolved the CVE with which it was associated.

However, none of the previous modules (P1–P4) include a formal verification of these assumptions. It is precisely to fill this gap that module **P5: Post-Deployment Assessment** is introduced, aimed at confirming, in an objective and measurable way, that the vulnerabilities declared as resolved are actually mitigated within the real operating context.
In many cases, the patching process can be affected by false positives or environmental contexts that differ from those taken into account by the detection tool. A CVE may be found to be "present" according to an assessment tool even when:

- it has already been mitigated by a workaround or pre-existing security configuration;
- it refers to a service that is not actually exposed or active in the environment being analyzed;
- the vulnerability has been incorrectly detected due to signature or correlation errors.

A truly mature patch management workflow should therefore include a contextual verification mechanism, i.e., ensuring that the vulnerability is actually reproducible in your environment before applying the patch. This can be done through white-box analysis or targeted penetration testing, focusing exclusively on the specific CVE and affected components, thus assessing the actual exposure of the system.
From an operational point of view, this verification can be carried out before or after the validation phase (P3), based on a cost-benefit analysis.

5.1 Post-Deployment Assessment

The purpose of module P5 is to formally verify the effectiveness of the security patch. This does not involve confirming that the patch has been installed correctly (which is already handled in P4), but rather validating that the original vulnerability, for which the patch was released, can no longer be exploited.
Practical example
Consider the case of a WordPress plugin that has a vulnerability such as stored XSS (persistent cross-site scripting). The vendor then releases a patch to sanitize the input, preventing the insertion of HTML tags. However, the payload that was submitted to the vendor used for the CVE was generated in an HTML context, while the vulnerability

could still be exploitable in a JavaScript context where sanitization is not present.
In this case, the system is still vulnerable, but in a more "subtle" way: the CVE is public, the exploits are known, and an attacker now has more precise information to conduct a targeted attack by changing only the context of the attack vector. This scenario highlights the need for effective post-patching verification, which formally validates the vulnerability through a post-patch analysis.

Output and Results Management

If the Post-Deployment Assessment determines that the vulnerability is still present (e.g., through a successful proof-of-concept exploit), the system must:

1. Immediately report the result to the Master Server and the customer;
2. Notify the vendor or patch maintainer, providing the updated payload or condition that reproduces the residual CVE;
3. Recalculate the risk associated with the CVE in the current context, updating the priority metrics;
4. Activate the risk management cycle, according to the strategies already defined in the previous section:
    - Accept: acceptance of risk with intensified monitoring;
    - Mitigate: introduction of compensating controls (e.g., WAF, hardening, segmentation);
    - Transfer: delegation of mitigation to the vendor or third parties;
    - Avoid: disposal or isolation of the vulnerable asset.

Riferimenti bibliografici

[1] F. M. Nicastro, *Security Patch Management*, CRC Press, Boca Raton, FL, 2011.

[2] J. C. Perez, *Gartner: Most IT Security Problems Are Self-Inflicted*, Computerworld, 9 Oct. 2001. Available at: https://www.computerworld.com/article/2583657/gartner--most-it-security-problems-self-inflicted.html

[3] OWASP Foundation, *OWASP Top Ten Project*. Available at: https://owasp.org/www-project-top-ten/

[4] OWASP Foundation, *OWASP Top 10:2025*. Available at: https://owasp.org/Top10/2025/0x00_2025-Introduction/

[5] OWASP Foundation, *A03:2025 Software Supply Chain Failures*. Available at: https://owasp.org/Top10/2025/A03_2025-Software_Supply_Chain_Failures/

[6] OWASP Foundation, *A06:2021 – Vulnerable and Outdated Components*. Available at: https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/

[7] European Parliament and Council of the European Union, *Regulation (EU) 2019/881 of 17 April 2019 on ENISA (the European Union Agency for Cybersecurity) and on information and communications technology cybersecurity certification and repealing Regulation (EU) No 526/2013*, Official Journal of the European Union, L 151, 7 June 2019, pp. 15–69. Available at: https://eur-lex.europa.eu/eli/reg/2019/881/oj (accessed 11 January 2023).

[8] European Union Agency for Cybersecurity (ENISA), *European Cybersecurity Certification Scheme for Cloud Services (EUCS)*, ENISA Report, 2020. Available at: https://www.enisa.europa.eu/publications/eucs-cloud-service-scheme (accessed 11 January 2023).

[9] United States Congress, *Homeland Security Act of 2002*, Public Law 107-296, 25 Nov. 2002. Available at: https://www.dhs.gov/homeland-security-act-2002 (accessed 11 January 2023).

[10] National Institute of Standards and Technology (NIST), *NIST SP 800-40 Rev. 4: Guide to Enterprise Patch Management Planning – Preventive Maintenance for Technology*, 2013. Available at: https://csrc.nist.gov/pubs/sp/800/40/r4/final

[11] B. Brykczynski and R. A. Small, *Reducing Internet-Based Intrusions: Effective Security Patch Management*, Software Productivity Consortium, IEEE, 2002. Available at: https://ieeexplore.ieee.org/document/1159029

[12] U. Gentile and L. Serio, *Survey on International Standards and Best Practices for Patch Management of Complex Industrial Control Systems: The Critical Infrastructure of Particle Accelerators Case Study*, 2019. Available at: https://www.researchgate.net/publication/331883190_Survey_on_international_standards_and_best_practices_for_patch_management_of_complex_industrial_control_systems_the_critical_infrastructure_of_particle_accelerators_case_study

[13] International Electrotechnical Commission (IEC), *IEC 62443 – Industrial communication networks – Network and system security*, IEC, Geneva, Switzerland, 2011. Available at: https://webstore.iec.ch/en/publication/22811

[14] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), *ISO/IEC 27002:2022 – Information Security, Cybersecurity and Privacy Protection – Information Security Controls*, ISO/IEC, Geneva, Switzerland, 2022. Available at: https://www.iso.org/standard/75652.html

[15] European Union Agency for Cybersecurity (ENISA), *International and Interoperable Certification Environment for SCADA System Updates*, ENISA Report, 2013. Available at: https://www.enisa.europa.eu/publications/window-of-exposure-a-real-problem-for-scada-systems

[16] North American Electric Reliability Corporation (NERC), *CIP-007-6 – Cyber Security: System Security Management*, NERC Standard, 2015. Available at: http://www.nerc.com/pa/Stand/Pages/CIPStandards.aspx (accessed September 2018).

[17] N. Dissanayake, A. Jayatilaka, M. Zahedi and M. A. Babar, *Software security patch management – A systematic literature review of challenges, approaches, tools and practices*, Information Software Technology, Vol. 144, 2022. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0950584921002147

[18] Polo Strategico Nazionale S.p.A., *Polo Strategico Nazionale – Cloud sicuro per l'Italia digitale*, Available at: https://www.polostrategiconazionale.it/

[19] Microsoft Corporation, *Azure Update Manager (formerly Update Management Center) – Patch Management for Azure and Hybrid Environments*, Available at: https://azure.microsoft.com/en-us/products/azure-update-management-center (accessed October 2025).

[20] V. A. Mehri, P. Arlos, and E. Casalicchio, *Automated Patch Management: An Empirical Evaluation Study*, in *Proceedings of the 2023 IEEE International Conference on Cyber Security and Resilience (CSR)*, pp. 321–328, IEEE, 2023.

[21] T. Pulliainen, *Linux Patch Management: Comparison of Practical Implementations*, Jyväskylä University of Applied Sciences, 2016.

[22] R. Sharma, A. Verma, S. Patil, and M. Reddy, *Red Hat Satellite Cron-Based Patching: A Zero-Touch Approach*, 2020.

[23] *Smart Patching with Cron Jobs: An Ops-Centric Perspective*, Available at: https://www.researchgate.net/publication/393971910_Smart_Patching_with_Cron_Jobs_An_Ops-Centric_Perspective

[24] N. Dissanayake, A. Jayatilaka, M. Zahedi, and M. A. Babar, *Software Security Patch Management – A Systematic Literature Review of Challenges, Approaches, Tools and Practices*, *Information and Software Technology*, Vol. 144, 2022, Article 106771. Publisher: Elsevier.

[25] U. Rani, *The Recent Automating System Patching via Satellite and Puppet Integration*, 2021.

[26] Red Hat, *Red Hat Satellite – Gestione centralizzata dei sistemi*, Available at: https://www.redhat.com/it/technologies/management/satellite

[27] The Foreman Project, *Katello Plugin – Content and Subscription Management for Foreman*, Available at: https://theforeman.org/plugins/katello/

[28] Spacewalk Project, *Spacewalk – The Open Source Systems Management Solution*, Available at: https://spacewalkproject.github.io/

[29] Orcharhino, *orcharhino – Hybrid Data Center Management Made Easy*, Available at: https://orcharhino.com/en/

[30] ManageIQ, *ManageIQ – Open Source Hybrid IT Management*, Available at: https://www.manageiq.org/

[31] Elastic N.V., *Elastic Stack – Ricerca, osservabilità e sicurezza unificate*, Available at: https://www.elastic.co/elastic-stack

[32] Puppet, *Continuous Configuration Automation – Use Cases and Benefits*, Available at: https://www.puppet.com/why-puppet/use-cases/continuous-configuration-automation

[33] Cronhub, *Crontab Guru – The quick and simple editor for cron schedule expressions*, Available at: https://crontab.cronhub.io/

[34] National Institute of Standards and Technology (NIST), *National Vulnerability Database (NVD)*, Available at: https://nvd.nist.gov/

[35] National Institute of Standards and Technology (NIST), *Common Platform Enumeration (CPE)*, Available at: https://nvd.nist.gov/products/cpe

[36] GNU Project, *GPG/PGP – GNU Privacy Guard and Pretty Good Privacy Encryption Standards*, Available at: https://www.gnupg.org/

[37] Red Hat, *Ansible Documentation*, Available at: https://docs.ansible.com/

[38] Red Hat, *YUM – Yellowdog Updater, Modified*, Available at: https://access.redhat.com/solutions/9934

[39] Fedora Project, *DNF – Dandified YUM Package Manager*, Available at: https://docs.fedoraproject.org/en-US/quick-docs/dnf/

[40] Debian Project, *APT – Advanced Packaging Tool*, Available at: https://it.wikipedia.org/wiki/Advanced_Packaging_Tool

[41] openSUSE Project, *Zypper – Command Line Package Manager for openSUSE*, Available at: https://it.opensuse.org/Portal:Zypper

[42] Arch Linux, *Pacman – Package Manager for Arch Linux*, Available at: https://wiki.archlinux.org/title/Pacman

[43] Microsoft, *Windows Remote Management (WinRM)*, Available at: https://learn.microsoft.com/it-it/windows/win32/winrm/portal

[44] Microsoft, *Windows Server Update Services (WSUS)*, Available at: https://learn.microsoft.com/it-it/windows-server/administration/windows-server-update-services/get-started/windows-server-update-services-wsus

[45] Microsoft, *WSUS for Microsoft Environments*, Available at: https://learn.microsoft.com/en-us/windows-server/administration/windows-server-update-services/get-started/windows-server-update-services-wsus

[46] JSON.org, *JavaScript Object Notation (JSON)*, Available at: https://www.json.org/json-en.html

[47] National Institute of Standards and Technology (NIST), *Common Vulnerability Scoring System (CVSS)*, Available at: https://nvd.nist.gov/vuln-metrics/cvss

[48] Microsoft Azure, *Azure Virtual Machine Snapshot Documentation*, Available at: https://learn.microsoft.com/it-it/azure/virtual-machines/snapshot-copy-managed-disk?tabs=portal

[49] Microsoft Azure, *Azure Monitor – Overview and Fundamentals*, Available at: https://learn.microsoft.com/it-it/azure/azure-monitor/fundamentals/overview

[50] Zabbix LLC, *Zabbix – Enterprise-Class Open Source Monitoring Solution*, Available at: https://www.zabbix.com/

[51] Nagios Enterprises, *Nagios – IT Infrastructure Monitoring*, Available at: https://www.nagios.org/

[52] Prometheus Authors, *Prometheus – Monitoring System and Time Series Database*, Available at: https://github.com/prometheus/prometheus

[53] National Institute of Standards and Technology (NIST), *NIST Special Publication 800-30 Revision 1 – Guide for Conducting Risk Assessments*, Available at: https://csrc.nist.gov/pubs/sp/800/30/r1/final

[54] IBM, *Infrastructure as Code (IaC) – Automazione e gestione dell'infrastruttura IT*, Available at: https://www.ibm.com/it-it/think/topics/infrastructure-as-code

[55] Palo Alto Networks, *Policy as Code (PaC) – Definizione e gestione automatizzata delle policy di sicurezza*, Available at: https://www.paloaltonetworks.com/cyberpedia/what-is-policy-as-code

Il tutor Aziendale Mauro De Pascale

Il tirocinante
Alberto Ameglio