Highlights

Penterep: Comprehensive Penetration Testing with Interactive Checklists

Willi Lazarov, Pavel Seda, Zdenek Martinasek, Roman Kummel

- A novel approach to conducting penetration testing using interactive checklists.
- Automation of the penetration testing process to increase its effectiveness and reduce repetitive tasks.
- Penetration testing environment design and implementation supporting the novel approach using high customizability.
- Showcase using the provided case study, concluded with lessons learned.

Penterep: Comprehensive Penetration Testing with Interactive Checklists

Willi Lazarov[a,*], Pavel Seda[a], Zdenek Martinasek[a] and Roman Kummel[b]

[a]Brno University of Technology, Antonínská 548/1, Brno 601 90, Czechia
[b]HACKER Consulting, Luběnice 65, Luběnice 783 46, Czechia

ARTICLE INFO

Keywords:
Checklists
Cybersecurity
Ethical hacking
Methodology
Penetration testing
Reporting
Vulnerability assessment

ABSTRACT

In the contemporary landscape of cybersecurity, the importance of effective penetration testing is underscored by NIS2, emphasizing the need to assess and demonstrate cyber resilience. This paper introduces an innovative approach to penetration testing that employs interactive checklists, supporting both manual and automated tests, as demonstrated within the Penterep environment. These checklists, functioning as a quantifiable measure of test completeness, guide pentesters through methodological testing, addressing the inherent challenges of the security testing domain. While some may perceive a limitation in the dependency on predefined checklists, the results from a presented case study underscore the criticality of methodological testing. The study reveals that relying solely on fully automated tools would be inadequate to identify all vulnerabilities and flaws without the inclusion of manual tests. Our innovative approach complements established methodologies, such as OWASP, OSSTMM, ISSAAF, and PTES, providing crucial support to penetration testers and ensuring a comprehensive testing process. Implemented within the Penterep environment, our approach is designed with deployment flexibility (both on-premises and cloud-based), setting it apart through an overview comparison with existing tools aligned with state-of-the-art penetration testing approaches. This flexible and scalable approach effectively bridges the gap between manual and automated testing, meeting the increasing demands for effectiveness and adaptability in penetration testing.

1. Introduction

Penetration testing, short pentesting, is growing in importance and is becoming a necessary part of every company or organization that uses information and communication technologies (ICT). The ever-increasing threat of cyberattacks and known cyberattack cases emerging in critical situations such as cyberattacks on hospitals during the COVID-19 pandemic [30, 53, 2] support this claim [44]. Furthermore, cyberattackers use more complex and sophisticated techniques to exploit and use system vulnerabilities. Next, organizations due to the increasing integration of information technologies into their regular processes (e.g., in hospitals [47] or banking sector [45]) become more exposed to risks associated with cyber attacks. Penetration testing allows companies to simulate real-world attacks and identify weaknesses in their systems. Using the results from penetration testing, companies can react quickly and strengthen their cyber resilience. Next, regular penetration testing is appropriate due to the dynamic increase of new technologies such as the Internet of Things (IoT) [23, 10], artificial intelligence (AI) [17], or cloud computing [25] that enable new cyber threats [18].

The increasing demand to conduct regular penetration tests is also highlighted by the European Directive Network and Information Security (NIS2), which expands the number of obliged subjects and the scope of responsibilities, such as the need to test and demonstrate cyber resilience [19]. Furthermore, there are several publications provided by NIST (e.g., 800-53 [31]) providing the catalog of security and privacy controls for information systems and organizations to protect organizational operations and assets, individuals, other organizations, and the nation from a diverse set of threats and risks, including hostile attacks, human errors, natural disasters, structural failures, foreign intelligence entities, and privacy risks. All of these recommendations aim to increase the organization's protection of sensitive data and information, ensure regulatory compliance, and minimize the company's or organization's risk of financial and reputational losses associated with cyber incidents [48].

Penetration testing is performed using various testing tools that identify vulnerabilities, provide security assessment, assist in risk mitigation, provide security assurance, help improve incident response plans, assess the security of web applications (vulnerabilities such as SQL injection or cross-site scripting), improve security awareness by notifying employees about potential security threats, and help assess the security of third-party vendor services or solutions [15]. The representative set of available tools can range from scanner tools such as Nmap [33] or Metasploit [40] to complex solutions such as Burp Suite [37] or Zed Attack Proxy (ZAP) [55], a detailed overview is provided in Section 2.2.

*Corresponding author
lazarov@vut.cz (W. Lazarov); sedap@vut.cz (.P. Seda); martinasek@vut.cz (.Z. Martinasek); kummel.r@penterep.com (.R. Kummel)
https://www.vut.cz/en/people/willi-lazarov-221556 (W. Lazarov); https://www.vut.cz/en/people/pavel-seda-154208 (.P. Seda); https://www.vut.cz/en/people/zdenek-martinasek-88846 (.Z. Martinasek); (.R. Kummel)
ORCID(s): 0000-0001-6820-8391 (W. Lazarov); 0000-0002-6689-1980 (.P. Seda); 0000-0002-6504-5619 (.Z. Martinasek)

1.1. Problem Statement

The penetration testing varies depending on the methodologies used and the environment being tested, making it challenging to generalize the testing procedure [16]. For instance, testing network infrastructure differs significantly from web application penetration testing. When examining a computer network, the tester will mainly work with the IP address, subnet mask, and default gateway of the network, identifying different types of network and endpoint devices (e.g., router, server, firewall, etc.) and assorted network services with different parameters. On the other hand, web application penetration testing might involve identifying various programming languages, frameworks, and other web pages utilized with various user inputs. Due to these variations, it is not feasible to create a one-size-fits-all procedure for every potential test target. Instead, different ICT domains necessitate individualized approaches, adding a layer of complexity to penetration testing.

In an effort to mitigate the above-mentioned penetration testing challenges, a number of automated tools have been developed. However, these automated testing tools do not ensure the completeness of the penetration testing which is critical. This limitation arises because certain vulnerabilities are either hard or even impossible to test automatically, particularly those that appear in various parts of the tested environment, represent business logic errors, or necessitate visual inspection. However, given the natural purpose of penetration testing, it is necessary to detect all vulnerabilities within a specific scope. Therefore, a suitable combination of manual and automated testing based on a measurable methodology (e.g., proposed interactive checklist) is necessary. To the best of our knowledge, current tools and applications fall short of offering this possibility.

1.2. Key Contributions

In this section, we highlight the key contributions of our work, providing a unique perspective on effective, transparent, and complete penetration testing of the tested targets. We denote the following key contributions of this paper:

- A novel approach to penetration testing that effectively supports both automated and manual tests.
- Design, implementation, and verification of the novel approach in the developed Penterep environment.
- Showcase using the provided case study, concluded with lessons learned.

Based on the problem outlined in Section 1.1, we propose a novel approach to penetration testing that combines manual and automated testing while ensuring the completeness, transparency, and accuracy of penetration testing. To apply and verify our novel approach, we developed the Penterep environment. The Penterep provides interactive checklists, a knowledge base containing predefined tests and vulnerabilities, as well as support for automated testing or report generation.

Penterep helps testers perform automated and manual tests with less probability of missing any important test. Furthermore, it allows for storing the records, reports, and results of the tests and sharing them between testers. Among the application of standard methodologies such as the Open Worldwide Application Security Project (OWASP), Open Source Security Testing Methodology Manual (OSSTMM), ISSAF (Information Systems Security Assessment Framework), and Penetration Testing Execution Standard (PTES), testers can complement the methodology with their checklists for a variety of tests and vulnerabilities. In addition to other solutions, Penterep allows the personalization of test selection for specific targets based on its properties. The very appreciated capability is also the possibility to deploy the Penterep environment on-premise or within the cloud, which increases the usability of the solution for various customers and their specific requirements. We demonstrate the environment's capabilities using a case study showing exploitation on selected web pages and comparing it to other state-of-the-art existing tools or other solutions.

1.3. Paper Organization

This paper is organized into six sections. The background and related work are summarized in Section 2. Section 3 provides the platform capabilities and the ability to create its methodologies on top of the standard ones. Section 4 describes the Penterep architecture, data collection, deployment use cases, available workflows to conduct the penetration test, and the usage perspective of the pentester and the customer who requests the pentest. Section 5 presents a case study showing the results of penetration testing using Penterep on the selected web application. In this section, the lessons learned from the Penterep deployment are described for the selected scenarios. Section 6 concludes the paper and introduces open challenges.

2. Background and Related Work

Penetration testing is an extensive method of assessing the security level of information and communication systems. Testing can be related to network infrastructure, services, application layer protocols, web applications, or end users. The main objective of penetration testing is to preventively detect, verify, and describe vulnerabilities of the tested target so that all vulnerabilities found can be remediated. This method of security testing falls under the realm of ethical hacking and is done quite legitimately with the prior consent of the test subject, most often by an external party under contract. The scope and process of penetration testing may vary depending on the methodology, the environment being tested, and the way it is performed [51, 32, 7]. Methodologies determine how to proceed with penetration testing. Commonly used methodologies include OWASP, OSSTMM, ISSAF, and PTES. The different phases of penetration testing are shown in Figure 1 [51, 7].

The first phase of penetration testing is *Pre-engagement*: Before starting the test, it is important to determine the scope, logistics, rules, and all objectives of the penetration

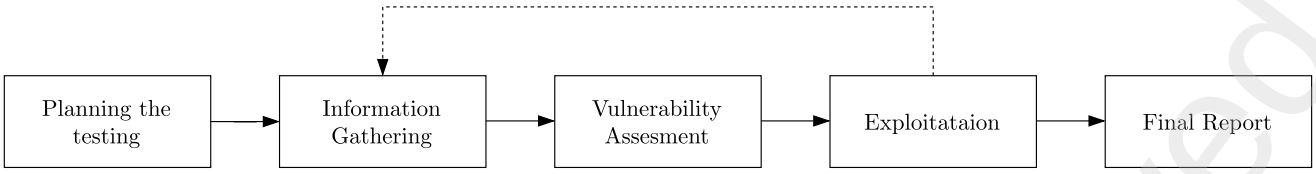

Figure 1: Penetration testing process.

test based on financial and time capacity [39]. *Information gathering*: This phase aims to obtain as much information as possible about the environment under test, such as the network infrastructure, devices, services, or technologies used, and their parameters (e.g., operating system, software, and its version) [27]. *Vulnerability assessment*: Based on the information gathered in the previous testing phase, the vulnerability discovery process includes two main parts: searching for vulnerabilities in databases of known vulnerabilities[1] and partial testing of the environment for the presence of a specific vulnerability (e.g., injection vulnerability, misconfiguration, etc.) [11]. *Exploitation*: Discovering vulnerabilities as a result of penetration testing is not enough. The risks found must be verified by exploiting the vulnerability. The exploitation of a vulnerability may reveal a previously inaccessible part of the tested environment (e.g., the administration of a web application). When such a case happens, the previous two phases have to be repeated [7]. *Final report*: The final report describes the identified risks, determines their severity, and makes recommendations based on which the contracting authority should take appropriate action (e.g., fix source code, system updates, etc.) [4].

2.1. Penetration Testing Approaches

Penetration testing can be assessed: (i) according to the pentester starting position, (ii) by the level of knowledge of the environment, (iii) by the method execution [43, 16].

According to the *tester starting position*, penetration testing can be divided into [54, 20]:

- **External testing**: that is performed without authorization and access to an internal part of the tested environment. According to the phases listed in Figure 1, the tester attempts to compromise the target and obtain or modify protected assets. Therefore, the main objective of external testing is to test the level of security against threats from the external environment of the tested target (usually the Internet) [20].
- **Internal testing**: where the tester has access to the internal part of the tested environment. In this case, the tester represents a legitimate user (e.g., an employee of the tested entity) or an attacker who accesses the intranet. In internal testing, security is evaluated, in particular, security mechanisms, system policies, and access control [54, 20].

[1] Common Vulnerabilities and Exposures (CVE)

By the *level of knowledge of the environment*, penetration tests can be divided into [6, 23, 42, 6]:

- **Black-box tests**: performed without initial knowledge of the tested environment. Testing consists of gradually uncovering individual components and finding their vulnerabilities. An example of such a vulnerability could be an outdated version of an operating system, a misconfigured web server, or an untreated user input of the application under test [56, 6, 23].
- **White-box tests**: which rely on complete knowledge of the test environment. The tester has full access to all system components, source code, database, etc. For white-box tests, a deeper analysis is usually performed. In the case of application testing, there is also a static analysis during which the tester looks for security vulnerabilities in the source code [23, 42].
- **Gray-box tests**: which are composed of a combination of white-box and black-box tests. To reduce time consumption, the tester has information to facilitate finding and verification of vulnerabilities in the environment under test (e.g., source code, structure database, or access data). Unlike white-box tests, there is no complete static analysis of the source code. Information tends to be used only to better understand the functionality of the environment under test as opposed to black-box tests, where the tester must obtain this information himself [23, 50, 6].

By the *method execution*, penetration testing can be divided into [5, 28, 9]:

- **Manual testing**: where the penetration tester has full control over the testing process and the results obtained are completely dependent on his process and skills. The advantage of manual testing is, in particular, the accuracy and consistency in the detection of vulnerabilities. On the other hand, the disadvantage of this method is time consumption and the need to have extensive knowledge [28].
- **Automated testing**: using automated tools to find vulnerabilities in a tested environment. The tools use a set of test strings, databases of known vulnerabilities, etc. The tester is then presented with the results of the possible findings. Compared to manual testing, this method is much faster and requires only knowledge

of how to work with the tools. On the other hand, the disadvantage may be a lower success rate in finding vulnerabilities, false positives, or the inability to test for some vulnerabilities. (e.g., when the vulnerability manifests itself in a different location than the input or logical reasoning is required) [5].

- **Semi-automated testing**: which combines manual and automated tests, using the advantages and mitigating the disadvantages of both types. The exploration and identification phase is carried out primarily by automated tools, while potential vulnerabilities and the possibility of exploitability are tested manually [9].

2.2. Tools Overview

There are many different penetration testing tools on the market, which can be further divided according to the target area (e.g., computer networks, web applications, mobile applications, etc.). Automated penetration testing tools, such as Nmap [33], Metasploit [40], SQLmap [46], Nikto [14], represent some of the widely used tools. On the other hand, solutions for management alone, support for manual penetration testing, or a combination of both are less numerous groups. Since our solution is comprehensive and aims to support all the operations performed during penetration testing, we make the comparison exclusively against platforms and applications that are not purpose-built and represent this less numerous group of tools. The following solutions will be compared:

- **Burp Suite**: Software tool for web application penetration testing. The tool is distributed in 3 editions – Community, Pro, and Enterprise. Although the community edition primarily supports manual tests that can be customized, the Pro and Enterprise editions offer fully automated web vulnerability scans [37].
- **Acunetix**: Web application for continuous vulnerability scanning and testing. The main features of this solution include more than 7000 vulnerability scans, an easy-to-use user interface, and various integrations. The Acunetix application is only offered in annual subscription mode with hidden pricing [1].
- **Nessus**: Automated vulnerability assessment tool for web application, cloud, and external attack surface scanning. Nessus can be extended with community or custom plugins and is distributed in 2 editions – Expert and Professional [49].
- **w3af**: Open source framework for detecting and exploiting vulnerabilities. The framework can be used with a graphical or console user interface and is capable of identifying more than 200 vulnerabilities [52].
- **OpenVas**: Full-featured vulnerability scanner. The scanner includes a feed of tests to detect vulnerabilities based on CVEs, vendor advisories, and other sources. The product is available in 3 editions – Community, Enterprise, and Cloud [22].
- **Penterep**: Web platform for penetration testing (the solution presented in this paper). The Penterep platform supports both manual and automated testing and aims to cover the whole testing area and include all steps of the penetration test.

The comparison of Penterep (presented solution) with Burp Suite, Acunetix, Nessus, w3af, and OpenVas is shown in Table 1. The Penterep platform addresses the issue of penetration testing to maximize testing capabilities, supports manual testing, and increases overall coverage of the tested area. This makes it significantly different from other tools that specialize only in specific ICT subareas. This paper approaches the problem with a novel approach to address the shortcomings of other solutions with the main goal of providing complete support for penetration testing.

Apart from Penterep and Burp Suite, the other compared tools are limited to automated testing. One such tool is w3af, which allows for more configurable automated tests but lacks the ability to manage projects, and its scope of use for penetration testing is therefore limited. The Acunetix web application, Nessus, and OpenVas build testing primarily on its scanners, with little or no configuration and a preview of the entire automated testing process. Due to this limitation, they are used more for vulnerability assessment than as a tool to support manual penetration testing. Burp Suite, on the other hand, offers advanced automated testing capabilities through its modules, including its configuration and integration of community extensions, but its use is limited to penetration testing of web applications.

It should be noted that these are long-developed tools. The oldest of all is Nessus, which was developed in 1998 under the brand name "The Nessus Project" [13]. Burp Suite was developed in 2003 [37] and Acunetix has been on the market since 2005 [1], both commercial solutions for large companies. OpenVAS was created as an open source fork of the Nessus project in 2006 and has been continuously developed since then [21]. The w3af tool has been developed since 2013 as an open source tool and is therefore community-driven [52]. The actual development of the Penterep web platform started in May 2021 and the first Proof of Concept (PoC) was completed in June of the same year. Thanks to its architecture, modularity, and the methodology used, it was possible to create a large-scale environment for penetration testing in a short time.

3. Novel Pentesting Methodology

Different methodologies for penetration testing define the phases that testers follow. Most of them are generic and differ mainly in their steps. Among the more well-known, these include OSSTMM (1. Induction Phase, 2. Interaction Phase, 3. Inquest Phase, 4. Intervention Phase) [24], PTES (1. Pre-engagement Interactions, 2. Intelligence Gathering, 3. Threat Modeling, 4. Vulnerability Analysis, 5. Exploitation, 6. Post Exploitation, 7. Reporting) [38], ISSAF (1. Planning and Preparation, 2. Assessment, 3. Reporting) [41], or NIST (1. Planning, 2. Execution, 3. Post-Execution) [43]

Table 1
Overview comparison of relevant tools with the presented solution.

Feature/Functionality	Burp Suite	Acunetix	Nessus	w3af	OpenVas	Penterep
Support for automatic testing	✓	✓	✓	✓	✓	✓
Support for manual testing	✓	✗	✗	✗	✗	✓
Web application testing	✓	✓	✓	✓	✓	✓
Mobile application testing	✗	✗	✗	✗	✗	✗
Network infrastructure testing	✗	✗	✓	✗	✓	✓
Application server testing	✗	✗	✓	✗	✓	✓
Periodic testing (scheduling)	✓	✓	✓	✗	✓	✗
Detailed test settings	✓	✗	✗	✗	✗	✓
Environment visualization	✗	✗	✗	✓	✗	✓
Interactive checklists	✗	✗	✗	✗	✗	✓
Documentation and help	✓	✓	✓	✓	✓	✓
Records of findings	✓	✓	✓	✗	✓	✓
Severity indicator	✓	✓	✓	✓	✓	✓
Severity calculator	✗	✗	✓	✗	✓	✓
Graphical user interface	✓	✓	✓	✓	✓	✓
Mobile device support	✗	✓	✓	✗	✓	✓
Import of results	✗	✗	✗	✗	✓	✓
Export of results	✓	✓	✓	✓	✓	✓
Project management	✗	✗	✓	✗	✓	✓
Custom report templates	✗	✓	✗	✗	✗	✓
Multi-user support	✗	✓	✓	✗	✓	✓
Free license version	✓	✗	✗	✓	✓	✓
Integration of custom tools	✓	✓	✓	✓	✗	✓
Third party integration	✓	✓	✓	✓	✓	✗

and OWASP (4 standards) [36]. While OSSTMM focuses on operational security and human factors, OWASP standards are primarily specialized in web and mobile applications, NIST covers practical testing methodology, and PTES provides more structured processes.

Our methodology is based on interactive checklists. Each checklist contains a specific set of test cases based on an existing standard, such as the OWASP ASVS (Application Security Verification Standard) [35] or on testers' custom checklists. Each test contains the following properties: name, objective, description, and instructions on how to perform the test. The tests are further divided into groups (e.g., authentication, authorization, session management, etc.) and by type of test objective. Our methodology presents testers with the most relevant tests to increase accuracy and reduce testing time. An example of this approach is a web page with a form where testers are presented with tasks to test form inputs. Conversely, other tests are performed against a `robots.txt` text file according to the checklist, where form elements cannot be logically located but sensitive exposed information can be present. The methodology also works with an environment where tests' target of a specific system (e.g., Linux) will not be submitted in the case of another system (e.g., Windows).

Since penetration tests are designed to find and verify various vulnerabilities, one or more vulnerabilities are assigned to each test as part of our methodology. Each vulnerability contains pre-defined code, name, description, cause, impact, recommendation, and severity. In this way, the tester does not have to manually write everything down when the text is finished and the relevant vulnerabilities are already present. Once a vulnerability is selected, each of the listed properties can be further customized, for example, to further describe the manifestations of the vulnerability in the context of the tested target, to indicate the exact location of the vulnerability, to recalculate the severity, etc. Like custom tests, the vulnerability can be completely customized by the tester independently of the selected checklist.

Another crucial aspect of our approach involves the integration of different checklists. The penetration testing process is contingent on its scope, allowing for the testing of individual applications or extensive infrastructures comprising various network devices and application servers. We assign a set of targets to each checklist, and these targets undergo specific tests based on their properties. For instance, consider the following procedure: (i) utilizing the checklist for testing infrastructure, the tester scans the network to identify communicating devices; (ii) selecting a device, the tester performs a test, revealing it as a server; (iii) following the infrastructure checklist, the tester scans open ports, such as port 80 for web services; (iv) upon discovering an application on the web service, the tester applies the checklist for web application penetration testing, in conjunction with the checklist for testing infrastructure, triggering another set of tests. Alternatively, when the objective is to conduct penetration testing solely on a specific web application, the test target is predefined as that particular web application. In this scenario, the tester uses only the checklist

for web application penetration testing without the checklist for infrastructure testing as in the previous scenario. This interactive testing approach provides flexibility for testers based on the conditions and scope of penetration testing.

4. Penterep Penetration Testing Environment

To support and increase the efficiency of manual and automated penetration testing, we have developed the Penterep penetration testing environment. Penterep is a web penetration testing environment that guides the penetration tester through interactive checklists so that it is always clear what test is about to be performed. For each test, the tester is also provided with detailed instructions on how to run the test, making it possible to involve less experienced testers who are learning simultaneously during testing. Any findings are recorded in the system as the results of individual tests, including severity and attachments.

The penetration testing environment provides an extensive knowledge base in which all vulnerabilities are already thoroughly described or can be added as appropriate. Testers then only need to select a specific vulnerability from the list. At the same time, multiple testers can work in the testing environment without leaving any targets untested or duplicating the work of other testers. This is because each team member has an immediate overview of which tests have been completed and which are still waiting to be executed. In addition, testers also have the option for selected test cases to call automated tests directly from the Penterep, with results being fed back after the work is completed, where they are automatically recorded in the system.

When the penetration test is complete, Penterep produces a fully automated report listing all the details and a list of all the vulnerabilities discovered, including their severity, recommendations for remediation, etc. Thanks to the knowledge base, links to references, descriptions of relevant tests, and other information are also added to the report for each finding. Attachments uploaded by testers during testing are also automatically included with the report.

4.1. Penterep Architecture

The environment for penetration testing supports both on-premise and cloud-based as a Software as a Service (SaaS) solution for individual deployment. The complete production environment, including all its components, can be seen in Figure 2.

In the cloud, the penetration testing environment is operated as a multi-tenant system, where each tenant T_1, T_2,…,T_n accesses a logically separate application connected to tenant's dynamic database DDB_1, DDB_2,…, DDB_n and persistent storage PS_1, PS_2,…, PS_n. While each present tenant has an assigned dynamic database and persistent storage with write privileges, the static database SDB is shared with the others and is kept read-only. Each use case of the penetration testing environment originates in the SDB, where the input data are defined, and their relationships are created. The output of this no-code process is static data representing the data structure of the entire penetration testing environment (e.g., test cases, device types, predefined vulnerabilities, etc.). The static data then determine how the dynamic data will be represented, which enter the database only when the penetration testing environment is used by the tenant in a production environment, and working with it is thus completely separate from the definition of the use cases in the static database. The cloud environment also contains the following supporting services that ensure high availability, scalability, and stability of the system operations: Kubernetes, Load balancing, Backup and Recovery, CI/CD pipelines, Message queuing, and In-memory caching.

The second case of penetration testing environment distribution is on-premise (OP). In this case, the penetration testing environment is deployed on the customer's infrastructure and its operation is managed by the customer. In contrast to the cloud, each OP of the environment contains its static database. The penetration testing environment is deployed as a virtual machine (VM). For the sake of an update, the process is as follows: (i) data are automatically backed up, (ii) the customer downloads the new version, and (iii) the data are restored to it. Except for load balancing and CI/CD pipelines, on-premise also includes supporting services that run in the VM.

In both cases, the user interacts with the penetration testing environment through her/his web browser, which contains a modular Single-page application (SPA). Before the SPA is provided to the user, the user must first authenticate to the User Management Server (UMS). The UMS serves as an authentication and authorization server and contains all user accounts, licenses, and penetration testing modules. Once the user has successfully authenticated against the UMS, she/he is returned to the environment's application server with an assigned token from the UMS. The application server authenticates the token against the UMS, and if valid, the SPA is returned to the user and loaded into her/his web browser. From this point on, all requests are sent by the SPA based on user interaction and other application events. A single UMS is dedicated to the authentication process for all deployed environments (tenants and OPs), and single users can access multiple different environments if there is an existing invitation for them to do so (e.g., contract penetration testing for multiple companies running the environment).

The final component of the whole system is the Server for automated testing (SAT). Users either perform testing manually and upload the results to the Penterep or run automated tests that are executed by the SAT that saves the results to the Penterep when the test is complete. The target of an automated test can be almost anything, as long as the SAT has a suitable built-in tool to test it. There can be multiple test servers, and the SAT can also run on the user's device. The Penterep, to support penetration testing as much as possible, allows testing to be performed not only by penetration testers, but also by other users, such as developers, who fix the found vulnerabilities, and the tester then retests the presence of the vulnerability. The key feature of SAT

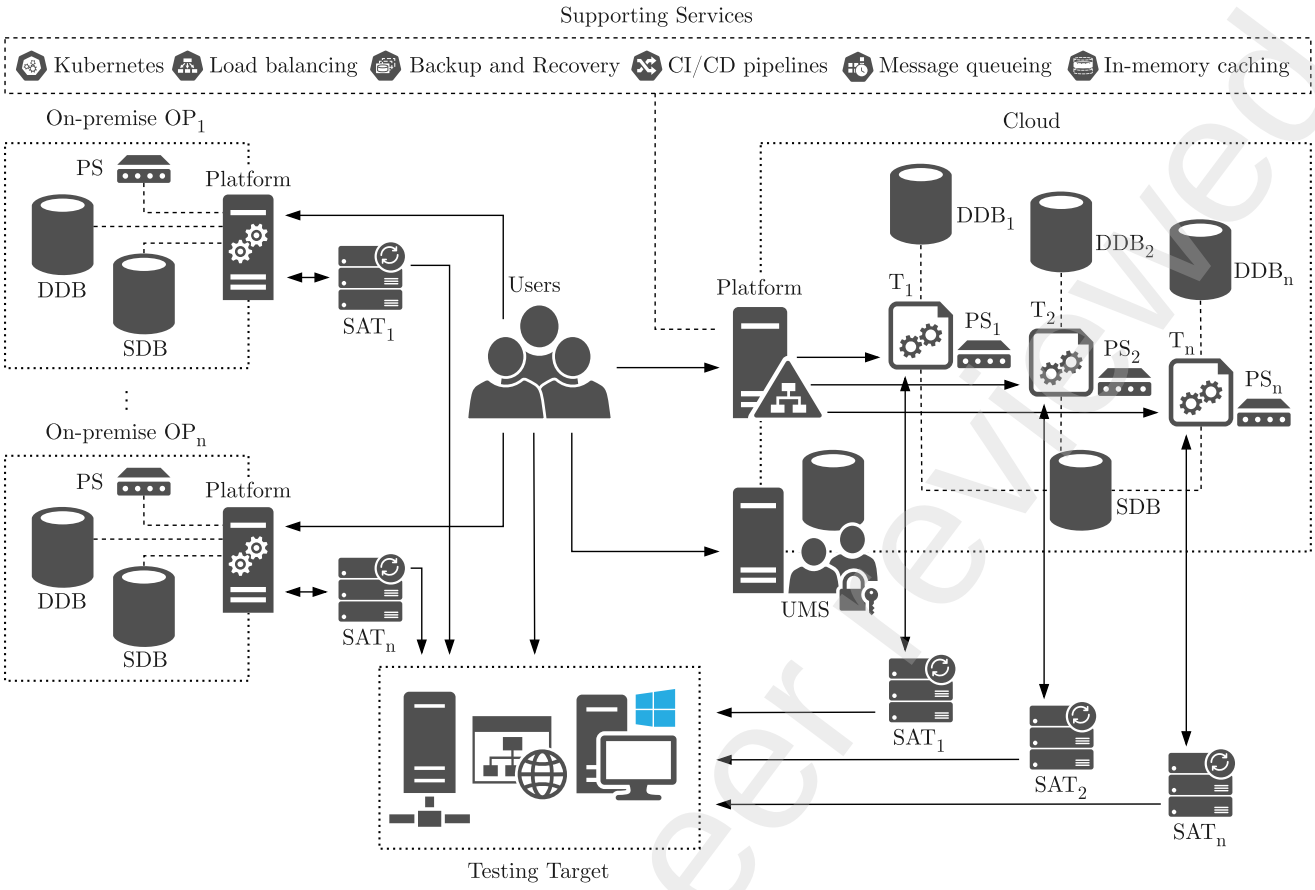

Figure 2: Penterep architecture consists of Users; Cloud platform environment with Tenants T_1, T_2,..., T_n; Dynamic databases DDB_1, DDB_2,...,DDB_n; Persistent storages PS_1, PS_2 ..., PS_n; Static database SDB; User Management System (UMS); Servers for automated testing SAT_1, SAT_2,..., SAT_n; On-premise platforms OP_1,OP_2,...,OP_n; and Testing Target.

is the dividing automation process into predefined single-purpose subtests that increase the accuracy of the overall penetration testing and therefore reduce false positives.

4.2. Penetration Testing Workflows

The course of the penetration test in the environment can be divided into 5 phases, which further consist of sub-operations. 1. Project setup; 2. Selecting methodology; 3. Penetration testing; 4. Analysis of the testing results; 5. Final report. The entire workflow[2] is shown in Figure 3 and the content of each phase is described in detail below.

At the beginning of each penetration test, the tester or the project administrator (in the case of team testing) creates a project in agreement with the customer. During the project setup, it is necessary to define basic information, including contact details. The testing objectives, scope, deadline, and other additional information are also defined.

After the project is created, the next step is to choose the testing methodology that will guide all the following steps. Our approach is that testers choose checklists for projects that are defined in the knowledge base. All possible test targets (e.g., network devices, servers, web applications, etc.) are then assigned to each checklist with the appropriate properties. Then each test target is assigned a set of tests that the tester should perform. To support less experienced testers, each test also includes detailed instructions on how to execute it. When selecting a methodology, one or a combination of multiple checklists can be selected, allowing, for example, penetration testing of an infrastructure consisting of various network devices and application servers with running applications. Once the testers confirm the selected checklists, they proceed to penetration testing.

The process of penetration testing depends on the selected checklists, especially in the way of testing. The Penterep supports both manual and automated testing. Each test can always be performed manually, or, if possible, testers have the option to run an automated test. In manual testing, the tester follows the instructions to accomplish the task given by the test. If a vulnerability or other deficiency is found during the test, the tester confirms this in the Penterep and is presented with a menu to select a predefined vulnerability or create a completely new one. Since the subtests clearly state their purpose, we pre-define the vulnerabilities into the knowledge base and link them to the

[2]The diagram that shows penetration testing workflow is using Business Process Model and Notation (BMPN) 2.0 notation [34].

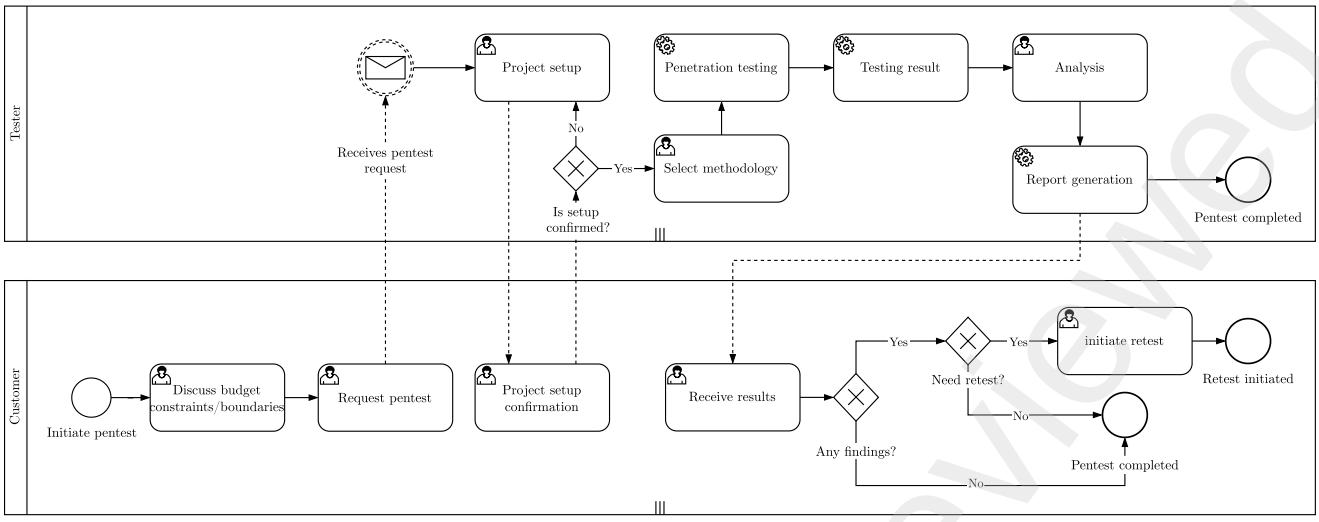

Figure 3: Penetration testing workflow in Penterep environment.

corresponding tests. Therefore, each test can be linked to one or more vulnerabilities. Each vulnerability contains a pre-filled name, description, severity, and recommendation. The tester can modify these predefined properties as needed. If the test allows it, the tester has the option to run an automated test that is inserted into a queue from which the SAT selects the appropriate test, executes it outside the environment (see architecture 2), and sends the results back to the Penterep, which stores it in the same way as if the tester had done it manually. However, the main difference in this process is the vulnerability status, which is always set to "verified" in manual testing, whereas it is set to "not verified" in automated testing. This is because automated testing can return false positives, and therefore the tester is forced to check and validate the results of automated testing. Until she or he does this, the finding does not progress to the final report. With this approach, Penterep aims to avoid false positives and thus increase the accuracy and completeness of penetration testing.

Once penetration testing is complete, an analysis of all findings follows, during which the testers additionally comment on the vulnerabilities and other weaknesses found, if they have not already done it during the testing itself. This also includes reviewing the results of automated testing, severity calculations, where the vulnerability occurred in the context of the target being tested, and other information.

In the last step, testers have the option to automatically generate a final report with the usual details (cover page, table of contents, summary, conclusion, etc.) and a list of all discovered vulnerabilities including their severity, remediation, etc. From the knowledge base (static database), links to references, descriptions of tests performed, and other information are also added to the report for each finding. Attachments uploaded during the testing are also included with the report. Automatic report generation is one of the key features of the platform, which significantly reduces the time-consuming penetration testing process, as testers do not have to manually write the report in a text editor.

The generated report is sent to the customer through a secure channel. Based on the report, the customer should remediate all vulnerabilities and weaknesses found. At this stage, the project can be considered complete, but if a retest has been agreed upon or after testing, another penetration test is performed after the vulnerabilities have been fixed. In this case, the presence of all recorded vulnerabilities in the project is re-verified by the testers. To facilitate the entire retest and communication, developers and other customer employees can be invited into the project to see the vulnerabilities found and mark them as fixed, passing them to retest. If necessary, a discussion is available for all vulnerabilities for communication between all parties.

4.3. Automated Penetration Testing

The automated testing process in the presented solution is very different from conventional vulnerability scanners. The main difference is that the servers for automated testing are not located inside, but outside, both the cloud and on-premise platforms. To operate it, testers need to have a device with SAT installed, that will perform the testing and a testing manager that will pair with the platform and then request tests from the platform, execute them and return the results for further processing, as can be seen in Figure 4.

An automated test cannot be run freely and is only assigned to tests that could be automated (e.g., HTTP header check, URL query, etc.). The test cases and their parameters are predefined and only need to know the target of the test. This is taken from the project from which the test is loaded into the queue in a "scheduled" state. The SAT then requests tests from the queued ones from the platform in each released thread, changing the test state to "running". When the test is completed, the SAT sends the result back to the platform and the status changes to "finished".

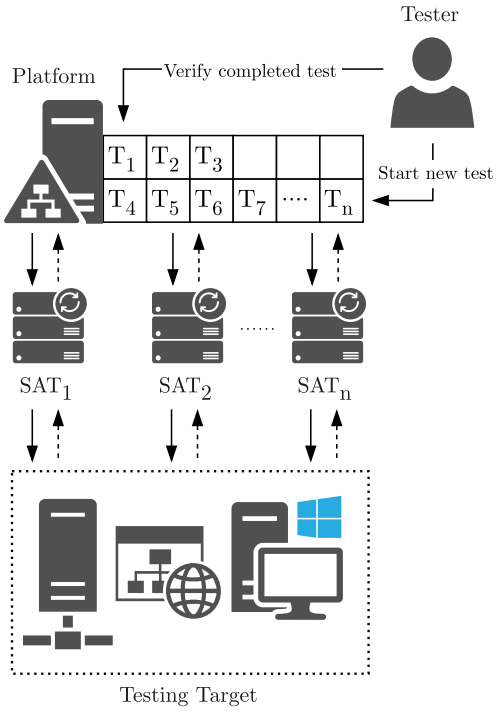

Figure 4: Tester starts tests $T_1, T_2, \ldots, T_n$ that are added into queue from which all connected $SAT_1, SAT_2, \ldots, SAT_n$ request them. Independently, each SAT executes assigned tests against the specified target and then returns the results to the platform where the tester verifies them.

Test results may vary. If a scan has been performed, the resources found are returned to the platform for further testing (e.g., devices, web pages, backup locations, etc.). If a vulnerability is found, it is associated with the specific test and the target that was tested. However, unlike manual penetration testing, the vulnerability status is set to "unverified" and such a finding must be verified. This is due to the risk of false positives, so testers must validate each automated test finding, or it will not be transferred to the report.

Individual test cases are created based on an automated tool with specific parameters that perform the test on the SAT. All supported automated testing tools are open source under the PenterepTools[3] toolkit and can also be used platform-independently. One of the main advantages of this approach is complete transparency, which is critical for penetration testing.

4.4. Benefits of the Novel Approach

Based on the presented novel approach and environment for penetration testing, the main benefits can be divided into completeness, accuracy, and transparency as follows:

- **Completeness**: Based on the agreed scope and limits of testing, checklists are selected during the creation of the penetration testing project that contains all the predefined tests that need to be performed. Until all tests are finished, the project cannot be considered complete. The final report is generated only from all completed tests and confirmed vulnerabilities to ensure completeness.
- **Accuracy**: Each test is created based on a methodology, standard, or from experts in the field. The tests are not generic and are divided into specific test cases, with one or more vulnerabilities associated with each test that can be detected by the test. If a test has been automated, it is marked as "unverified" and must be manually checked to avoid false positives.
- **Transparency**: Penetration tests are described in detail and referenced to their origin. In automated testing, the equipment performing the testing is under the full control of the tester. The automated test tools are open source and the output includes a log of all actions performed for possible revision.

In addition to the main inherent benefits above, we can also consider the benefits of the environment for penetration testing from the perspective of pentesters and customers, as described in the sections below.

4.4.1. Pentesters View

Considering the penetration tester's perspective, one of the main advantages of the presented penetration testing environment is the unification of all activities in one place with regard to teamwork. This is because, in general, if several testers work on one project, there is a risk of duplication, which is mitigated here by interactive checklists. Another advantage is the detailed tests for less experienced testers and a general check that nothing was missed during the test.

In terms of time capability, one of the main benefits of the presented approach is the creation of a knowledge base that contains possible test targets and vulnerabilities in addition to the penetration tests. Similarly to a test, each vulnerability has a predefined description, severity, remediation recommendations, and other information that the tester can further specify in the context of the test objective. All of this, along with automatic report generation, significantly saves testers' time and eliminates the need to perform a significant amount of manual activities as would be the case with a branch-and-bound approach where testers keep notes in various applications (e.g., Excel, Word, Trello, etc.), while our penetration testing environment aggregates data and its management in one place.

4.4.2. Customer View

An important entity in the whole presented environment is also the customer, who has requested the penetration testing and after its completion controls the final report. In our approach, we consider the presence of the customer directly in the project not only as one person but also as a group of customer employees (e.g., developers, system administrators, etc.) who have access to the individual test results, i.e., the vulnerabilities found. After completion or during the actual test, they can fix the found vulnerabilities

[3] PenterepTools are freely available for download as open source under the GPLv3+ license at https://pypi.org/user/penterep/.

and request a retest. If necessary, it is also possible to have a discussion on individual findings directly with the testers, thus facilitating communication between the two parties.

5. Case Study

To validate our penetration testing environment, experimental testing was performed against a vulnerable web application. Furthermore, to compare our approach and demonstrate the main advantages of our solution, the presented case study also includes solutions compared in Section 2.2. To maximize accuracy, we developed a web application (see Section 5.1) that contains selected vulnerabilities with severities of information, low, medium, high, and critical. Since the application was developed entirely under our control, we were able to measure the completeness and accuracy of the penetration testing within this case study.

The vulnerable web application was deployed to a sandbox environment, which contained a tester that interacted with the Penterep environment to perform manual tests and run automated tests. Since in our approach, the server for automated testing is always under the control of the tester, the sandbox environment also included the SAT connected to Penterep. In addition, the w3af and cloud solutions Burp Suite, Acunetix, Nessus, and OpenVas were used in the case study. After the testing was completed, the results from all tools were aggregated and thoroughly analyzed. Since we knew exactly what vulnerabilities the tested application contained, we limited the results from all tools to only the true positives. The entire environment and the steps of the experimental testing are shown in Figure 5 and the case study parameters are listed in Table 2.

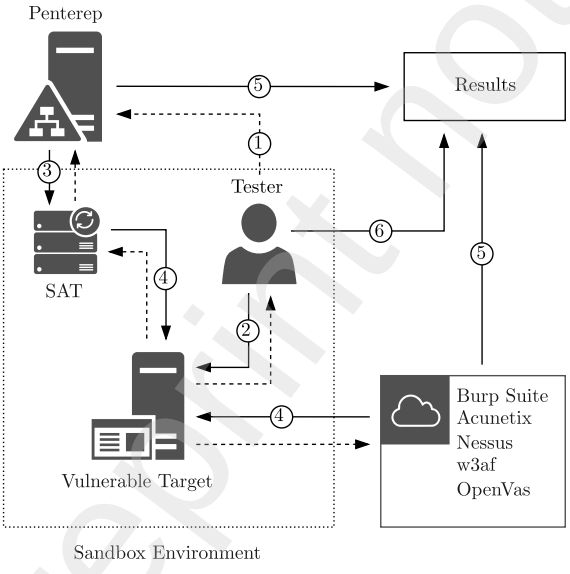

Figure 5: Sandbox environment: 1. Interaction with Penterep environment (checklists, guidelines, running automated tests, etc.); 2. Manual penetration testing; 3. Automated penetration testing; 4. Aggregation of all findings from penetration testing; 5. Results analysis filtering true positives.

Table 2
Overview of penetration testing tools.

Platform	Version	License	Extensions
Penterep	0.9.16	Enterprise	SAT
Burp Suite	2023.2	Enterprise	None
Acunetix	15.4	Premium	None
Nessus	10.4.1	Professional	None
w3af	1.6.49	Community	None
OpenVas	23.3.0	Enterprise	None

5.1. Testing Target

The vulnerable web application was developed in the PHP programming language and has a common web user interface and content including text and images. Logical features include authentication mechanisms in the form of log-in and registration forms, publicly unavailable administration, and session management. There are also sensitive backups located on the web server on which the application runs. The total number of vulnerabilities and other flaws present is 50 and a complete list of vulnerabilities with their descriptions is provided in Appendix A. We also published on GitLab the source code of the developed web application containing all presented vulnerabilities [26].

5.2. Case Study Results

The test results in Table 3 show a significant limitation of restricting penetration testing to automated tools only. This is because some vulnerabilities require visual inspection and user interaction, represent business logic errors, or are located at a different place than the input, and their testing is problematic or impossible to automate. Except for Penterep, all solutions used for testing were used with the highest available license and without any additional extensions. The use of SAT in Penterep was necessary for automated testing, as Penterep does not include SAT in its core environment. It is also important to note that if SAT is not used at all, completeness is not affected, as all tests can always be performed manually, while those that are automated are run optionally by the tester. In addition, each automated test has predefined parameters, making it single-purpose, which increases accuracy. Transparency is then ensured by the distribution of SAT tools as open source and running under full tester's control.

The Penterep environment is built on checklists that must be selected before the start of penetration testing and thus represent a measurable parameter of test completeness. Therefore, this dependency can be perceived as a disadvantage from a certain perspective, since the tester always has to test according to a certain methodology, standard, and other procedures specified by the checklist. However, our approach is based on methodological testing, which is critical due to the nature of the security testing domain. This fact is underscored by the results of the presented case study, as without forcing a set of manual tests, it would be impossible to find all the vulnerabilities and other flaws present using only fully automated tools.

Table 3
Case study demonstrative results from penetration testing using Penterep and other tools Burp Suite, Acunetix, Nessus, w3af, and OpenVas, without additional plugins.

Vulnerability code	Severity	Penterep			Burp Suite	Acunetix	Nessus	w3af	OpenVas
		Automat	Manual	Total					
VULN01	Info	●	○	●	○	○	○	○	○
VULN02	Info	●	○	●	○	○	●	○	○
VULN03	Info	●	○	●	○	○	○	○	○
VULN04	Info	●	○	●	○	○	○	○	○
VULN05	Info	○	●	●	○	○	○	●	○
VULN06	Info	○	●	●	○	○	●	○	○
VULN07	Info	○	●	●	○	○	○	○	○
VULN08	Info	○	●	●	●	○	○	●	●
VULN09	Info	●	○	●	○	○	○	○	●
VULN10	Info	●	○	●	○	○	○	○	○
VULN11	Info	●	○	●	○	○	○	○	○
VULN12	Info	●	○	●	●	○	○	●	○
VULN13	Low	●	○	●	●	○	●	○	○
VULN14	Low	○	●	●	○	○	○	○	○
VULN15	Medium	○	●	●	○	○	○	○	○
VULN16	Low	○	●	●	○	○	○	○	○
VULN17	Low	○	●	●	○	○	○	○	○
VULN18	Low	○	●	●	○	○	○	○	○
VULN19	Medium	○	●	●	○	○	○	○	○
VULN20	Low	○	●	●	○	○	○	○	○
VULN21	Low	○	●	●	○	○	○	○	○
VULN22	Low	○	●	●	○	○	○	○	○
VULN23	High	○	●	●	○	○	○	○	○
VULN24	Medium	○	●	●	○	○	○	○	○
VULN25	Medium	○	●	●	○	○	○	○	○
VULN26	Low	○	●	●	○	○	○	○	○
VULN27	Low	●	○	●	○	○	○	○	○
VULN28	Info	●	○	●	○	○	○	○	○
VULN29	Medium	●	○	●	○	○	○	○	●
VULN30	Low	●	○	●	○	○	○	○	○
VULN31	Medium	○	●	●	○	○	○	○	○
VULN32	Medium	●	○	●	○	○	○	○	○
VULN33	Medium	●	○	●	○	○	○	○	○
VULN34	Critical	○	●	●	○	○	○	○	○
VULN35	Low	○	●	●	○	○	○	○	○
VULN36	Medium	○	●	●	○	○	○	○	○
VULN37	Critical	○	●	●	○	○	○	○	○
VULN38	Medium	○	●	●	○	○	●	○	○
VULN39	Low	○	●	●	○	○	○	○	○
VULN40	Low	○	●	●	○	○	○	○	●
VULN41	Critical	●	○	●	●	○	○	○	●
VULN42	High	○	●	●	○	○	○	○	○
VULN43	High	●	○	●	○	○	○	○	○
VULN44	High	○	●	●	○	○	○	○	○
VULN45	Low	●	○	●	●	●	●	○	○
VULN46	Medium	○	●	●	●	●	○	○	○
VULN47	Medium	○	●	●	○	○	○	○	○
VULN48	Low	●	○	●	○	●	○	○	○
VULN49	Low	●	○	●	○	○	○	○	●
VULN50	Low	●	○	●	●	○	○	○	○
Discovered	-	21/50	29/50	50/50	7/50	3/50	5/50	3/50	6/50

5.3. Limitations

In this case study, penetration testing was performed on a single testing target. Although presenting results from multiple testing targets could expand the vulnerability list and better identify intersections between various tools, the primary objective of this article is not to conduct an exhaustive survey of all penetration testing tools on a myriad of testing targets. Instead, the focus is on introducing a novel approach for comprehensive penetration testing that includes manual tests verified on a demonstrative example.

The findings in this case study compare the vulnerabilities discovered by Penterep with tools Burp Suite, Acunetix, Nessus, w3af, and OpenVas. These tools allow for the integration of third-party plugins. Using these integrated plugins and their extensions, the tools have the potential to expand the list of identified vulnerabilities that could result in a significant overlap in the findings. However, our innovative approach, centered around interactive checklists, ensures that the penetration tester does not overlook any critical test case, including manual tests that are often disregarded by other tools featured in the case study.

5.4. Lessons Learned

For more efficient adoption of this new approach, we highlight the main lessons learned with the following general recommendations:

a *Scope the penetration testing properly:* Before starting penetration testing, it is important to determine the scope of the testing, the number of targets, support materials, deadlines, and any other testing conditions. To complete the test, measurable parameters must be established, which is supported by using checklists that determine what is to be tested against a given target, giving penetration testers and the customer a clear view of what has been tested and how.

b *Focus deeply on information gathering:* Information gathering aims to explore public information related to the target, identifying optimal ways for exploitation. This comprehensive process serves the purpose of understanding the functioning of the application or system, enabling the discovery of security vulnerabilities. This process involves two distinct approaches: Passive Information Gathering, which precedes active information gathering and is less invasive, relying solely on publicly available target information; and Active Information Gathering, which requires more preparation as it leaves traces, potentially triggering alerts to the target. This method involves actively engaging with the target, obtaining details about open ports, services, versions of operating systems and applications, and other pertinent information. The thorough emphasis on information gathering offers essential insights and intelligence, empowering pentesters to carry out comprehensive and effective security assessments. This, in turn, contributes to the improvement of the overall cybersecurity posture of the target system.

c *Analyze the results and verify the automated test results precisely:* Once penetration testing is conducted to identify vulnerabilities present in the given target, the tester has to precisely prioritize the vulnerabilities based on their potential impact and severity. Furthermore, the tester should perform a risk assessment that evaluates all true positive findings considering the potential damage and the value of the assets. Here, the tester has to precisely mark false positive results provided by automated tests to avoid misleading security alerts, which might lead to overlooking valid threats. Eliminating false positives provided by automated tests could be reduced by using Machine Learning (ML) [8, 3] or by tweaking the tool's settings to reduce the number of false positive claims.

d *Discuss the results with the clients and provide suitable recommendations:* The findings should be communicated to the clients through a presentation that emphasizes the identified issues along with the recommended mitigation measures. All of these discoveries should be discussed with the product owners and the development team to ensure that all stakeholders are informed. Furthermore, a comprehensive report should be provided detailing the list of vulnerabilities discovered with specific actions, configuration changes, or patches to improve overall system security. If needed, the penetration tester should also offer recommendations for ongoing improvements to promote security best practices, such as regular pentests or the integration of GitHub security checks [12].

6. Conclusion

This paper introduces an innovative approach to conducting penetration tests through interactive checklists. These checklists provide crucial support to pentesters, ensuring that no essential aspects of testing are overlooked. The proposed approach serves as a valuable complement to established methodologies such as OWASP, OSSTMM, ISSAAF, and PTES. The designed approach is developed in the Penterep environment demonstrating the innovative approach. In this paper, we also highlight the design of the Penterep environment including its deployment flexibility (both on-premise and cloud-based), and overview comparison with the existing tools following state-of-the-art penetration testing approaches. To the best of our knowledge, using our innovative approach, we bridge the gap between manual and automated approaches while ensuring flexibility, scalability, and effectiveness of penetration testing. The importance of emphasizing improvements in penetration testing is underscored by NIS2, which broadens the range of obligated entities and the scope of responsibilities. This includes the requirement to test and demonstrate cyber resilience [19].

Given the limitations of our penetration testing environment, we recognize open challenges for potential expansion to enhance the efficiency of penetration testing and its

management. This involves incorporating third-party tools and emerging AI methods to strengthen the penetration testing process. Specifically, in our case, AI offers alternative avenues for extending our innovative approach by categorizing, classifying, and assigning suitable checklists during the initial penetration testing phases. Another open challenge pertains to supporting the initial non-technical step, the preparation phase. Here, AI could assign testers to specific tests and subsequently assess the confidence and accuracy of individual findings. Tackling these challenges has the potential to propel the current state-of-the-art forward and effectively broaden the scope of our novel approach.

Acknowledgements

The research described in this paper was financially supported by the Ministry of the Interior of the Czech Republic, project No. VK01030019. We would also like to express our gratitude to the National Cyber and Information Security Agency (NCISA) of the Czech Republic for testing and providing feedback on the Penterep environment.

CRediT authorship contribution statement

Willi Lazarov: Conceptualization, Methodology, Software, Writing - Original Draft. **Pavel Seda:** Data Curation, Visualization, Writing - Review & Editing. **Zdenek Martinasek:** Supervision, Project administration, Funding acquisition. **Roman Kummel:** Resources, Investigation, Validation.

References

[1] Acunetix, 2024. Web application security scanner. URL: https://www.acunetix.com/.

[2] Ahmed, N.B., Daclin, N., Olivaux, M., Dusserre, G., 2023. Cybersecurity challenges for field hospitals: impacts of emergency cyberthreats during emergency situations. International Journal of Emergency Management 18, 274–292. doi:10.1504/IJEM.2023.132387.

[3] Alahmadi, B.A., Axon, L., Martinovic, I., 2022. 99% false positives: A qualitative study of SOC analysts' perspectives on security alarms, in: 31st USENIX Security Symposium (USENIX Security 22), USENIX Association, Boston, MA. pp. 2783–2800. URL: https://www.usenix.org/conference/usenixsecurity22/presentation/alahmadi.

[4] Alghamdi, A.A., 2021. Effective penetration testing report writing, in: 2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), pp. 1–5. doi:10.1109/ICECCME52200.2021.9591097.

[5] Auricchio, N., Cappuccio, A., Caturano, F., Perrone, G., Romano, S.P., 2022. An automated approach to web offensive security. Computer Communications 195, 248–261. URL: https://www.sciencedirect.com/science/article/pii/S0140366422003267, doi:https://doi.org/10.1016/j.comcom.2022.08.018.

[6] Aydos, M., Çiğdem Aldan, Coşkun, E., Soydan, A., 2022. Security testing of web applications: A systematic mapping of the literature. Journal of King Saud University - Computer and Information Sciences 34, 6775–6792. URL: https://www.sciencedirect.com/science/article/pii/S131915782100269X, doi:https://doi.org/10.1016/j.jksuci.2021.09.018.

[7] Bacudio, A.G., Yuan, X., Chu, B.T.B., Jones, M., 2011. An overview of penetration testing. International Journal of Network Security & Its Applications 3, 19.

[8] Ban, T., Samuel, N., Takahashi, T., Inoue, D., 2021. Combat security alert fatigue with ai-assisted techniques, in: Proceedings of the 14th Cyber Security Experimentation and Test Workshop, Association for Computing Machinery, New York, NY, USA. p. 9–16. URL: https://doi.org/10.1145/3474718.3474723, doi:10.1145/3474718.3474723.

[9] Bartusiak, A., Kühne, M., Nitschke, O., Lässig, J., Nicolai, S., Bretschneider, P., 2023. First step into automation of security assessment of critical infrastructures. Sustainable Energy, Grids and Networks 36, 101139. doi:10.1016/j.segan.2023.101139.

[10] Bella, G., Biondi, P., Bognanni, S., Esposito, S., 2023. Petiot: Penetration testing the internet of things. Internet of Things 22, 100707. URL: https://www.sciencedirect.com/science/article/pii/S2542660523000306, doi:https://doi.org/10.1016/j.iot.2023.100707.

[11] Blakley, B., Cranor, L., 2023. Katie moussouris: Vulnerability disclosure and security workforce development. IEEE Security & Privacy 21, 11–18. doi:10.1109/MSEC.2022.3222043.

[12] Bühlmann, N., Ghafari, M., 2022. How do developers deal with security issue reports on github?, in: Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing, Association for Computing Machinery, New York, NY, USA. p. 1580–1589. URL: https://doi.org/10.1145/3477314.3507123, doi:10.1145/3477314.3507123.

[13] Carey, M., Rogers, R., Criscuolo, P., Petruzzi, M., 2008. Nessus Network Auditing, Second Edition. Syngress. URL: https://books.google.cz/books?id=s4rUAQAACAAJ.

[14] CIRT.net, 2024. Nikto 2.5. URL: https://cirt.net/Nikto2.

[15] Dalalana Bertoglio, D., Zorzo, A.F., 2017. Overview and open issues on penetration test. Journal of the Brazilian Computer Society 23, 1–16.

[16] Denis, M., Zena, C., Hayajneh, T., 2016. Penetration testing: Concepts, attack methods, and defense strategies, in: 2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT), IEEE. pp. 1–6.

[17] Devagiri, J., Sidike, P., Niyaz, Q., Yang, X., Smith, S., 2022. Augmented reality and artificial intelligence in industry: Trends, tools, and future challenges. Expert Systems with Applications 207, 118002. doi:10.1016/j.eswa.2022.118002.

[18] El Kafhali, S., El Mir, I., Hanini, M., 2022. Security threats, defense mechanisms, challenges, and future directions in cloud computing. Archives of Computational Methods in Engineering 29, 223–246. doi:10.1007/s11831-021-09573-y.

[19] European Union, 2022. Directive (eu) 2022/2555 of the european parliament and of the council. URL: https://eur-lex.europa.eu/eli/dir/2022/2555.

[20] Gangupantulu, R., Cody, T., Park, P., Rahman, A., Eisenbeiser, L., Radke, D., Clark, R., Redino, C., 2022. Using cyber terrain in reinforcement learning for penetration testing. 2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS) , 1–8doi:10.1109/COINS54846.2022.9855011.

[21] Greenbone, 2024a. Background. URL: https://greenbone.github.io/docs/latest/background.html.

[22] Greenbone, 2024b. Open vulnerability assessment scanner. URL: https://www.openvas.org/.

[23] Heiding, F., Süren, E., Olegård, J., Lagerström, R., 2023. Penetration testing of connected households. Computers & Security 126, 103067. URL: https://www.sciencedirect.com/science/article/pii/S016740482200459X, doi:https://doi.org/10.1016/j.cose.2022.103067.

[24] ISECOM, 2024. Research. URL: https://www.isecom.org/research.html.

[25] Jayadeva, S., Ahmed, A.A.A., Malik, R., Shaikh, A.A., Siddique, M., Naved, M., 2023. Roles of cloud computing and internet of things in marketing management: A critical review and future trends, in: Proceedings of Second International Conference in Mechanical and Energy Technology, pp. 165–173. doi:10.1007/978-981-19-0108-9_18.

[26] Kummel, Roman, 2024. Vulnerable app. URL: https://gitlab.utko.feec.vutbr.cz/penterep/vulnerable-app.

[27] McKinnel, D.R., Dargahi, T., Dehghantanha, A., Choo, K.K.R., 2019. A systematic literature review and meta-analysis on artificial intelligence in penetration testing and vulnerability assessment. Computers & Electrical Engineering 75, 175–188.
[28] Mishra, S., 2021. Efficacy of unconventional penetration testing practices. Intelligent Automation and Soft Computing 31, 223–239. doi:10.32604/iasc.2022.019485.
[29] MITRE Corporation, 2024. Cwe – common weakness enumeration. URL: https://cwe.mitre.org/.
[30] Muthuppalaniappan, M., Stevenson, K., 2020. Healthcare cyberattacks and the covid-19 pandemic: An urgent threat to global health. International Journal for Quality in Health Care 33. doi:10.1093/intqhc/mzaa117.
[31] National Institute of Standards and Technology (NIST), 2020. Security and Privacy Controls for Information Systems and Organizations. Technical Report NIST Special Publication 800-53, Revision 5. U.S. Department of Commerce. Washington, D.C. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf.
[32] Nidhra, S., Dondeti, J., 2012. Black box and white box testing techniques-a literature review. International Journal of Embedded Systems and Applications (IJESA) 2, 29–50.
[33] NMAP.org, 2024. Nmap. URL: https://nmap.org/.
[34] Object Management Group, 2024. Business process model and notation (bpmn), version 2.0. URL: https://www.omg.org/spec/BPMN/2.0/PDF.
[35] OWASP Foundation, 2024a. Owasp application security verification standard. URL: https://owasp.org/www-project-application-security-verification-standard/.
[36] OWASP Foundation, 2024b. Owasp integration standards. URL: https://owasp.org/www-project-integration-standards/.
[37] PortSwigger, 2024. Burp suite. URL: https://portswigger.net/burp.
[38] PTES, 2024. High level organization of the standard. URL: http://www.pentest-standard.org/.
[39] Rak, M., Salzillo, G., Granata, D., 2022. Esseca: An automated expert system for threat modelling and penetration testing for iot ecosystems. Computers and Electrical Engineering 99, 107721.
[40] RAPID metasploit, 2024. Metasploit. URL: https://www.metasploit.com/.
[41] Rathore, B., Brunner, M., Dilaj, M., Herrera, O., Brunati, P., Subramaniam, R.K., Raman, S., U., C., 2024. Information system security assessment framework (issaf). URL: https://untrustednetwork.net/files/issaf0.2.1.pdf.
[42] Rawal, B., Manogaran, G., Peter, A., 2023. The basics of hacking and penetration testing, in: The Basics of Hacking and Penetration Testing, pp. 21–46. doi:10.1007/978-981-19-2658-7_2.
[43] Scarfone, K., Souppaya, M., Cody, A., Orebaugh, A., 2008. Technical guide to information security testing and assessment. NIST Special Publication 800, 2–25.
[44] Sharif, M.H., Mohammed, M., 2022. A literature review of financial losses statistics for cyber security and future trend. World Journal of Advanced Research and Reviews 15, 138–156. doi:10.30574/wjarr.2022.15.1.0573.
[45] Shulha, O., Ianenkova, I., Kuzub, M., Muda, I., Nazarenko, V., 2022. Banking information resource cybersecurity system modeling. Journal of Open Innovation Technology Market and Complexity 8, 10.3390/joitmc8020080. doi:10.3390/joitmc8020080.
[46] SQLmap, 2024. Automatic sql injection and database takeover tool. URL: https://sqlmap.org/.
[47] Tadiboina, S.N., 2022. The integration of handheld and smartphone-connected technologies into the doctor-patient relationship-ai. Journal of Positive School Psychology 6, 2933–2940.
[48] Teichmann, F., Boticiu, S., 2023. An overview of the benefits, challenges, and legal aspects of penetration testing and red teaming. International Cybersecurity Law Review 4, 1–11. doi:10.1365/s43439-023-00100-2.
[49] Tenable, 2024. Nessus vulnerability scanner: Network security solution. URL: https://www.tenable.com/products/nessus.
[50] Trickel, E., Pagani, F., Zhu, C., Dresel, L., Vigna, G., Kruegel, C., Wang, R., Bao, T., Shoshitaishvili, Y., Doupé, A., 2023. Toss a fault to your witcher: Applying grey-box coverage-guided mutational fuzzing to detect sql and command injection vulnerabilities, in: 2023 IEEE Symposium on Security and Privacy (SP), pp. 2658–2675. doi:10.1109/SP46215.2023.10179317.
[51] Vats, P., Mandot, M., Gosain, A., 2020. A comprehensive literature review of penetration testing & its applications, in: 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO), IEEE. pp. 674–680. doi:10.1109/ICRITO48877.2020.9197961.
[52] W3af, 2024. Open source web application security scanner. URL: https://w3af.org/.
[53] Wasserman, L., Wasserman, Y., 2022. Hospital cybersecurity risks and gaps: Review (for the non-cyber professional). Frontiers in Digital Health 4, 862221. doi:10.3389/fdgth.2022.862221.
[54] Xie, B., Li, Q., Qian, H., 2022. Weak password scanning system for penetration testing, in: Meng, W., Conti, M. (Eds.), Cyberspace Safety and Security, Springer International Publishing, Cham. pp. 120–130.
[55] ZAP, 2024. Zed attack proxy (zap). URL: https://www.zaproxy.org/.
[56] Zhu, Y., Cui, L., Ding, Z., Li, L., Liu, Y., Hao, Z., 2022. Black box attack and network intrusion detection using machine learning for malicious traffic. Computers & Security 123, 102922. URL: https://www.sciencedirect.com/science/article/pii/S0167404822003145, doi:https://doi.org/10.1016/j.cose.2022.102922.

Willi Lazarov is a researcher at the Faculty of Electrical Engineering and Communication, Brno University of Technology (BUT), Czech Republic, where he specializes primarily in research on profiling, interactivity, and active feedback methods to improve the effectiveness of cybersecurity education. As a researcher of security research projects, he also focuses on penetration testing and source code security analysis. He received his BSc. at BUT in 2022 and is currently completing his MSc. He also actively cooperates with the Czech Army, National Cyber and Information Security Agency, University of Defence, and research groups at Tampere University in Finland and Pennsylvania State University in the US. In recognition of his contributions, he has received awards such as the EY Cyber Security Trophy, EY ESO Cyber Security Future Promise, BUT Rector's Award, and Minister of the Interior Award for outstanding achievements in security research.

Pavel Seda currently holds the position of Development Architect at SAP Labs Czech Republic, focusing on the SuccessFactors Payroll Product. Concurrently, he holds the position of a researcher at Brno University of Technology (BUT), specializing in cybersecurity and heuristic algorithms. He earned his MSc degree in 2017 and later received a PhD in 2022 from the Faculty of Electrical Engineering and Communication at BUT. Additionally, he earned an MSc and Advanced MSc in Computer Systems, Communication, and Security from Masaryk University (MU), Faculty of Informatics in 2018 and 2023. In 2019, he completed a two-month internship at the National Taiwan University of Science and Technology. Pavel's professional experience includes working as a Java developer at IBM Czech Republic from 2014 to 2018, serving as a Software Engineer at MU from 2017 to 2022, and various other professional experiences.

Zdenek Martinasek is an Associate Professor at the Faculty of Electrical Engineering and Communication at Brno University of Technology (BUT), Czech Republic. He accomplished his MSc. degree in 2008 and received his Ph.D. degree in 2013. His research activities focus on applied cryptography, side-channel cryptanalysis, and cyber security. His biggest expertise is in power analysis attacks based on machine learning. Currently, he is involved as a task leader in security research projects of the Ministry of Interior of the Czech Republic. He has published more than 45 papers in international journals and conferences and has provided several invited research and teaching lectures focused on cybersecurity. During his Ph.D. studies, he completed an internship at TU Wien. He also helps to cover pedagogically Bachelor's and Master's program courses in Information Security.

Roman Kummel currently serves as the CEO of HACKER Consulting Ltd. Throughout his career, he held positions as an IT administrator in various companies from 1994 to 2011. Later, he transitioned to Seznam.cz, where he worked as a penetration tester specializing in web applications. Building on this experience, Roman began sharing his knowledge with the cybersecurity community through lectures at GOPAS. In the course of his teaching endeavors, he has educated over a thousand cybersecurity professionals, solidifying his recognition within the industry. Additionally, he holds industry-recognized certifications in Ethical Hacking, including CEHv11. Roman has also made significant contributions to the field, discovered vulnerabilities such as CVE-2019-2089 in the Android operating system, and developed innovative techniques like File From Frame Hijacking.

A. Case Study Vulnerabilities

In the case study, a web application containing a total of 50 vulnerabilities was developed. The description and references to the Common Weakness Enumeration (CWE) of all present vulnerabilities are listed in Table 4. All CWEs were referenced according to the MITRE database [29].

Table 4
Used vulnerabilities in case study experimental testing.

Code	CWE	Description
VULN01	200	OS type identification based on case sensitivity
VULN02	200	OS type identification based on TCP implementation deviations
VULN03	200	Web server identification based on the icons alias
VULN04	200	Web server identification based on server-status module
VULN05	200	Web server identification based on the content of default error pages
VULN06	200	Web server identification based on the order of HTTP response headers
VULN07	200	Web server identification based on responses to long URLs
VULN08	200	Web server identification based on the content of HTTP response headers
VULN09	200	Programming language identification based on session cookie name
VULN10	200	Programming language identification based on default session ID format
VULN11	200	Programming language identification based on file extensions
VULN12	200	Application author identification based on information in the application source code
VULN13	200	The contents of the robots.txt file reveal confidential information
VULN14	215	The X-Forwarded-For header allows debugging information to be displayed
VULN15	549	Ability to detect the password as it is entered
VULN16	521	Users are allowed to use a short password
VULN17	521	The password field limits the length of the input with the maxlength attribute
VULN18	200	The application provides information about the existence of a user account
VULN19	549	The recap page displays login information
VULN20	521	Users are allowed to use compromised passwords
VULN21	307	Password verification for guessing
VULN22	521	Inability to temporarily unmask the password being entered
VULN23	549	Transmission of login credentials in an unsafe manner
VULN24	328	Passwords are stored in an insufficiently secure format
VULN25	759	Passwords are not salted during hashing
VULN26	521	Users are not allowed to use passwords of 64 characters or more
VULN27	284	Cookie transfer across domains is not allowed
VULN28	284	Setting a cookie from another subdomain is not allowed
VULN29	693	Loading external resources from any domain is not allowed
VULN30	1018	It is possible to set any value to the session cookie
VULN31	384	Option to fix the session cookie value
VULN32	1018	Option to set the session cookie value with a parameter in the URL
VULN33	1018	When logging out, the session is not cancelled on the server side
VULN34	22, 98	Missing or insufficient protection against Local File Inclusion attacks
VULN35	284	Missing or insufficient filtering of access to certain file types
VULN36	20	Missing or insufficient server-side value validation
VULN37	89	Missing or insufficient protection against SQL injection attacks
VULN38	79	Missing or insufficient protection against XSS attacks
VULN39	284	Ability to access files intended for inclusion
VULN40	307	Lack of restrictions on automated tools (bots)
VULN41	530	Ability to access backup files
VULN42	269	The X-Forwarded-For header provides the user with higher privileges
VULN43	200	Leakage of confidential data during redirection
VULN44	284	Allowing unauthorized direct access to objects
VULN45	548	Allowed directory listing
VULN46	352	Missing or insufficient protection against CSRF attacks
VULN47	549	Unauthorized reverse access to data sent via POST
VULN48	16	Faulty DNS record
VULN49	16	Non-existent pages not returning a 404 status code
VULN50	524	Caching of pages