Revista de Informática Teórica e Aplicada - RITA - ISSN 2175-2745
Vol. 33, Num. 3 (2026) 45-58

REVIEW

Penetration Testing in Big Data Frameworks: A Systematic Review of Security Assessment for Hadoop and Spark

Testes de Penetração em Frameworks de Big Data: Uma Revisão Sistemática da Avaliação de Segurança para Hadoop e Spark

Luís Barata[1]*, Francisco Martins[1], Eurico Lopes[1]

Abstract: Big Data systems are expanding rapidly, and frameworks like Hadoop and Spark are now central to that growth. Yet this expansion also raises new security challenges, particularly regarding how vulnerabilities are identified, assessed, and mitigated in complex, distributed environments. While the existing literature predominantly focuses on defensive mechanisms, systematic evidence on offensive approaches such as penetration testing remains limited. This paper presents a systematic literature review of security assessment practices for Hadoop and Spark between 2015 and 2025, with an emphasis on penetration-testing techniques in Big Data frameworks. Our search across major digital libraries retrieved 1578 records. After title and abstract screening, 46 articles were selected. Of these, 20 were considered relevant and read in full; among these, only four explicitly applied penetration testing to Hadoop or Spark deployments. The review reveals a shortage of realistic testing environments and standardized metrics for evaluating the effectiveness of mitigation strategies, alongside a strong reliance on generic security tools rather than specialized offensive frameworks. These findings underscore the need for dedicated, framework-aware penetration-testing solutions for Hadoop and Spark, reproducible testing protocols, and tighter integration between offensive assessments and proactive mitigation practices.
Keywords: hadoop — spark — vulnerabilities — big data — intrusion

Resumo: Os sistemas de Big Data estão a expandir-se rapidamente, e *frameworks* como o *Hadoop* e o *Spark* são hoje centrais a esse crescimento. Contudo, esta expansão traz também novos desafios de segurança, em particular na forma como as vulnerabilidades são identificadas, avaliadas e mitigadas em ambientes distribuídos complexos. Embora a literatura existente se foque predominantemente em mecanismos defensivos, a evidência sistemática sobre abordagens ofensivas, como o *penetration testing*, permanece limitada. Este artigo apresenta uma revisão sistemática da literatura sobre práticas de avaliação de segurança para *Hadoop* e *Spark* entre 2015 e 2025, com ênfase em técnicas de *penetration testing* em *frameworks* de *Big Data*. A pesquisa em grandes bibliotecas digitais recuperou 1578 registos. Após triagem por título e resumo, 46 artigos foram selecionados. Destes, 20 foram considerados relevantes e lidos na íntegra; entre estes, apenas quatro aplicaram explicitamente *penetration testing* a implementações *Hadoop* ou *Spark*. A revisão revela uma escassez de ambientes de teste realistas e de métricas normalizadas para avaliar a eficácia das estratégias de mitigação, bem como uma forte dependência de ferramentas de segurança genéricas em detrimento de *frameworks* ofensivos especializados. Estes resultados sublinham a necessidade de soluções dedicadas de *penetration testing* sensíveis ao contexto de *Hadoop* e *Spark*, de protocolos de teste reprodutíveis e de uma integração mais estreita entre avaliações ofensivas e práticas de mitigação proactivas.
Palavras-Chave: hadoop — spark — vulnerabilidades — big data — intrusão

[1] *Escola Superior de Tecnologia, Instituto Politécnico de Castelo Branco (IPCB), Portugal*
***Corresponding author**: luis.barata@ipcb.pt
DOI: http://dx.doi.org/10.22456/2175-2745.151182 • **Received:** 25/10/2025 • **Accepted:** 11/05/2026

1. Introduction

The growing adoption of Big Data systems has expanded the role of distributed frameworks such as Apache Hadoop and Apache Spark across multiple sectors, including healthcare, finance, industry, IoT, and public administration [1, 2, 3]. This rapid growth not only reflects the effectiveness of these tech-

nologies in managing large volumes of data but also exposes critical domains to significant risks, given the strategic importance of data for decision-making and operational continuity [4, 5].

Due to their open and highly distributed nature, these frameworks enlarge and reshape the attack surface, making deployments susceptible to specific vulnerabilities that may compromise data integrity, availability, and confidentiality [6, 7]. While the scientific literature has largely emphasized defensive mechanisms—such as access control, encryption, and authentication—the active exploration of vulnerabilities through offensive approaches, particularly penetration testing, remains underexplored in Big Data contexts [8]. The limited number of studies that simulate real-world attacks, test documented Common Vulnerabilities and Exposures (CVEs), or employ adapted exploitation tools raises concerns about the effectiveness and empirical validation of current security solutions, especially considering the potential impact of successful attacks on critical infrastructures [7, 1].

In this context, this paper presents a systematic literature review of security assessment practices for Hadoop and Spark, with a particular focus on penetration-testing methodologies, validation environments, and practical exploitation of flaws. The main goal is to map the state of the art in offensive security for Big Data frameworks, while also accounting for defensive mechanisms, and to highlight existing gaps in this critical research area.

The remainder of the paper is organized as follows. Section 2 describes the methodology, including the search strategy, selection criteria, and quality assessment. Section 3 presents the main findings and a cross-study comparison (Table 3). Section 4 discusses methodological gaps and implications for research and practice. Section 5 outlines practical recommendations and future work, while Section 6 summarizes related studies. Finally, Section 7 concludes the paper with key insights and opportunities for advancing penetration testing in Big Data frameworks.

2. Methods

2.1 Review objectives and research questions

This systematic literature review aims to characterize how security assessment, with particular emphasis on penetration testing, is conducted in Big Data frameworks such as Hadoop and Spark. The review is guided by the following research questions:

- RQ1: How do existing studies assess security in Hadoop and Spark deployments, in terms of threats, vulnerabilities, and attack vectors?
- RQ2: Which penetration-testing techniques, tools, and experimental setups are used to evaluate security in Hadoop and Spark?
- RQ3: How do offensive (penetration-testing) approaches relate to defensive mechanisms such as authentication, encryption, and intrusion detection?
- RQ4: What are the main methodological limitations and gaps regarding reproducibility, realism of test environments, and availability of specialized offensive frameworks?

2.2 Search Strategy

The literature search was conducted across a wide range of well-established scientific databases and repositories, including IEEE Xplore, ACM Digital Library, arXiv, ScienceDirect, SpringerLink, Google Scholar, USENIX, DEFCON, Black Hat, DOAJ, and ResearchGate. This diversity of sources was intended to ensure a comprehensive view of both academic and applied research related to the security of Big Data frameworks, with particular emphasis on Hadoop and Spark.

The search strategy combined controlled vocabulary terms, when available, with free-text keywords related to vulnerabilities, penetration testing, and the Hadoop and Spark platforms. Boolean operators (AND, OR) and truncation (*) were adapted to the syntax of each database. Core search expressions followed patterns such as ((”Hadoop” OR ”Spark”) AND (”vulnerability” OR ”penetration test*” OR ”pentesting” OR ”CVE” OR ”exploit”)), and were refined iteratively through test queries to balance sensitivity and specificity. Each query, together with the target database, filters applied, and result counts, was recorded in a structured search log to support transparency and reproducibility of the review process.

Temporal filters restricted the scope to studies published between 2015 and 2025, reflecting the evolution of Hadoop and Spark security research over the past decade. Only publications written in English or Portuguese were considered; although studies in other languages (e.g., Mandarin, Spanish, Korean, or German) may contain relevant findings, they were excluded due to translation constraints, which constitutes a limitation of this review.

Across all databases and queries, the search log records 23 distinct search expressions and a total of 1578 raw results (including duplicates across queries). After deduplication and screening by title and abstract, 46 unique articles were selected for potential full-text review, as described in Section 2.6. The raw results form the starting point for the screening and selection procedures.

To complement academic sources, specialized cybersecurity repositories were also consulted—namely Exploit-DB, CVE Details, and the MITRE CVE database—to identify verified vulnerabilities and documented exploits directly related to Hadoop or Spark components. This triangulated search process was designed to capture both peer-reviewed research and practical insights from the security community, ensuring that the analysis encompassed technical as well as operational dimensions of Big Data vulnerability assessment.

2.3 Inclusion Criteria

This systematic review included studies that directly or substantially addressed security assessment in Big Data frame-

works, with explicit reference to Apache Hadoop and/or Apache Spark. Eligible articles had to meet at least one of the following conditions: (i) identify or analyse technical vulnerabilities in Hadoop or Spark deployments; (ii) describe intrusion or penetration-testing activities targeting these platforms; (iii) investigate fault exploitation methods or the practical impact of documented vulnerabilities (e.g., specific CVEs) on Hadoop/Spark components; or (iv) present theoretical or practical security testing approaches explicitly applied to Big Data environments that deploy Hadoop or Spark.

We accepted studies with academic, experimental, or applied perspectives (including case studies and industrial reports), provided they offered concrete contributions to the understanding, execution, or empirical evaluation of offensive or combined (offensive and defensive) security practices in these frameworks.

2.4 Exclusion Criteria

We excluded purely conceptual or high-level discussion papers that did not include any demonstrated practical or experimental application to Hadoop or Spark. Studies focusing on technologies unrelated to Big Data frameworks, or on Big Data platforms other than Hadoop or Spark, were also excluded.

In addition, we removed duplicate publications, preliminary versions of later peer-reviewed articles (e.g., preprints superseded by journal versions), and materials that were not clearly subjected to a scientific peer-review process, such as slide decks, tutorials, or non-technical white papers. Finally, full-text articles for which sufficient methodological detail was not available (e.g., missing description of the system under test or experimental setup) were not considered for detailed quality assessment.

2.5 Methodological Decisions

To ensure both robustness and relevance, several methodological decisions were defined before data collection and consistently applied throughout the review. First, we adopted a protocol-driven approach in which the search sources and query strings (Section 2.2), as well as the inclusion and exclusion criteria (Sections 2.3 and 2.4), were specified a priori and documented in a search log. This protocol restricted the scope to security assessment in Big Data frameworks with explicit reference to Apache Hadoop and/or Apache Spark, while still allowing for heterogeneous study types (experimental, case study, and applied industrial reports).

The inclusion and exclusion criteria were aligned with the main objective of this study: to identify research that explicitly addresses technical vulnerabilities in Hadoop or Spark, intrusion or penetration-testing methods, CVE exploitation, or practical attack simulations in distributed Big Data environments. Conceptual works without concrete validation on Hadoop/Spark, duplicate entries, and non–peer-reviewed publications were systematically filtered out during the screening process.

To evaluate the methodological soundness of the selected works, we applied a structured quality assessment framework to a subset of full-text articles most closely related to our research questions. Each study was scored from 0 to 2 across four key dimensions: (i) clarity and depth in the description of vulnerabilities and CVEs; (ii) reproducibility of the testing procedure (e.g., availability of configuration details, tools, and datasets); (iii) realism of the validation environment (testbed, cluster size, workloads); and (iv) degree of relevance to Hadoop or Spark systems and to offensive or combined security assessment. The aggregate score supported, but did not mechanically determine, the final inclusion decisions, which also considered conceptual fit with the review's focus on penetration testing in Big Data frameworks.

These methodological choices sought to strike a balance between the breadth of the search and the depth of critical analysis, ensuring that only studies offering verifiable and technically grounded insights were included in the final synthesis. By standardizing evaluation criteria across heterogeneous sources, the review provides a consistent basis for comparing methodological rigour and for identifying both contributions and limitations in the existing body of work on offensive security for Hadoop and Spark.

2.6 Selection Process

The selection process followed a systematic approach based on the inclusion and exclusion criteria defined in Sections 2.3 and 2.4. Screening was conducted in three sequential stages:

- **Initial filtering (title/abstract):** A total of 1578 raw records were retrieved across all database queries. These were screened by title and abstract, and duplicate entries were disregarded during the process. Studies whose relevance to security assessment in Hadoop or Spark was not evident at this stage were excluded, resulting in 46 unique articles selected for potential full-text review. These 46 articles are listed in Appendix A, along with screening decisions and metadata.

- **Full-text review:** The 46 articles selected in the previous stage were assessed for eligibility. Of these, 20 studies were identified as addressing security issues in Hadoop and/or Spark and were read in full, forming the basis for the overall mapping of security approaches (defensive and offensive). The remaining 26 articles were excluded after full-text examination.

- **Quality assessment and final selection:** A structured quality assessment was then applied to the 20 full-text articles that had been read in full. Based on the resulting scores (0 to 2 per criterion, as described in Section 2.7) and conceptual fit, 4 primary studies were ultimately selected as core evidence for penetration-testing practices in Hadoop and Spark, while the remaining studies were used to contextualize defensive mechanisms and broader security trends.

Table 1. Summary of the screening and selection process

Stage	Number of publications
Initial search (raw records)	1578
Title/abstract screening (unique articles selected)	46
Full-text articles read and assessed	20
Final primary studies on penetration testing	4

As shown in Table 1, the rigorous application of the inclusion, exclusion, and quality assessment criteria reduced the initial pool of 1578 raw records to 46 unique articles after title/abstract screening, then to 20 full-text articles, of which only 4 provide primary evidence on penetration-testing practices. This reduction not only reflects the breadth of the search but also highlights the predominance of defensive approaches in the literature on vulnerabilities in Big Data platforms, in contrast with the scarcity of explicitly offensive penetration-testing strategies. This synthesis underscores the relevance of the selected studies and sets the stage for the subsequent analysis of quality and methodological limitations presented in Sections 2.6–2.8.

All search queries, article-level screening decisions, and quality assessment scores are publicly available as a dataset in Zenodo (DOI: 10.5281/zenodo.19686185) [9].

2.7 Quality Assessment

The quality assessment of the selected studies was conducted using a structured scoring framework designed to ensure methodological consistency and transparency. Each study was analysed according to four predefined criteria—description of vulnerabilities or CVEs, reproducibility of the testing procedure, realism of the validation environment, and relevance to Hadoop or Spark—each rated on a scale from 0 to 2, as summarized in Table 2. A score of 0 indicated the absence of information or methodological rigour for a given criterion, 1 denoted a partial or superficial description, and 2 reflected full compliance with the expected level of technical detail and clarity.

This framework enabled a balanced comparison of heterogeneous sources, given the diversity of publication types included in the review, ranging from peer-reviewed academic articles to experimental reports and security-conference proceedings. By adopting a numerical scale, the evaluation process synthesised qualitative judgments into comparable quantitative indicators, facilitating the identification of patterns across studies—for instance, whether work that provided detailed descriptions of attack procedures also tended to use realistic validation environments or to discuss mitigation mechanisms. The resulting quality scores were used to weight the discussion of findings in Sections 3 and 4, where higher-rated studies provide the empirical backbone for the synthesis of vulnerabilities and penetration-testing approaches. This structured approach thus offers a transparent foundation for evaluating the reliability and generalization of the reviewed literature.

2.8 Limitations and Considerations on Replicability

Despite the rigour applied throughout the screening and assessment procedures, this systematic review is subject to several limitations that may influence the comprehensiveness and replicability of its findings.

First, the language and temporal constraints constitute clear sources of bias. By restricting the search to publications written in Portuguese and English and to the period 2015–2025, relevant studies published in other languages or outside this time window may have been overlooked. This exclusion, although necessary for feasibility and consistency, limits the cultural and historical breadth of the analysed research.

Second, a degree of subjectivity is inherently associated with the screening and evaluation procedures. Inclusion decisions relied on qualitative judgements operationalised through the 0–2 scoring framework applied to dimensions such as methodological rigour and practical relevance. While this framework improved consistency, human interpretation during title/abstract screening and full-text assessment may still have introduced variability, particularly in borderline cases where methodological quality was not explicitly reported.

Finally, the scarcity of empirical penetration-testing studies in the existing literature significantly constrains the evidence base. Most publications on Hadoop and Spark security emphasise defensive strategies—such as access control, encryption, or anomaly detection—while relatively few provide detailed, reproducible descriptions of offensive or adversarial testing. In many cases, key parameters, attack configurations, and experimental conditions are only partially documented, which hampers replication of the reported tests and prevents systematic comparison across studies.

Collectively, these limitations suggest that the field of offensive security in Big Data remains both methodologically and empirically underdeveloped. Addressing these gaps will require broader linguistic coverage, clearer reporting standards for experimental setups, and a concerted effort to publish practical, experiment-driven research on vulnerabilities and penetration testing in distributed data systems.

Figure 1 summarises the flow of records through the identification, screening, eligibility, and inclusion phases, following a PRISMA-like structure.

The methodological protocol described in this section resulted in a curated set of studies that provide a structured view of security assessment in Hadoop and Spark. From the initial 1578 records, 20 full-text articles were read and assessed. Of these, 4 high-quality primary studies offer detailed evidence on penetration-testing practices. In the next section, we anal-

Table 2. Quality assessment criteria (score 0–2 per criterion)

Criterion	Description
Vulnerabilities/CVEs described	0: not mentioned; 1: mentioned without detail; 2: described with concrete examples or specific CVEs.
Reproducibility of pentesting	0: none; 1: partial (some steps/tools mentioned); 2: full (step-by-step procedure, tools, data/configuration).
Validation environment realism	0: not specified; 1: basic or small-scale simulation; 2: realistic cluster or testbed representative of production settings.
Relevance to Hadoop/Spark	0: no direct relation; 1: superficial or tangential mention; 2: Hadoop/Spark is a central focus of the study.

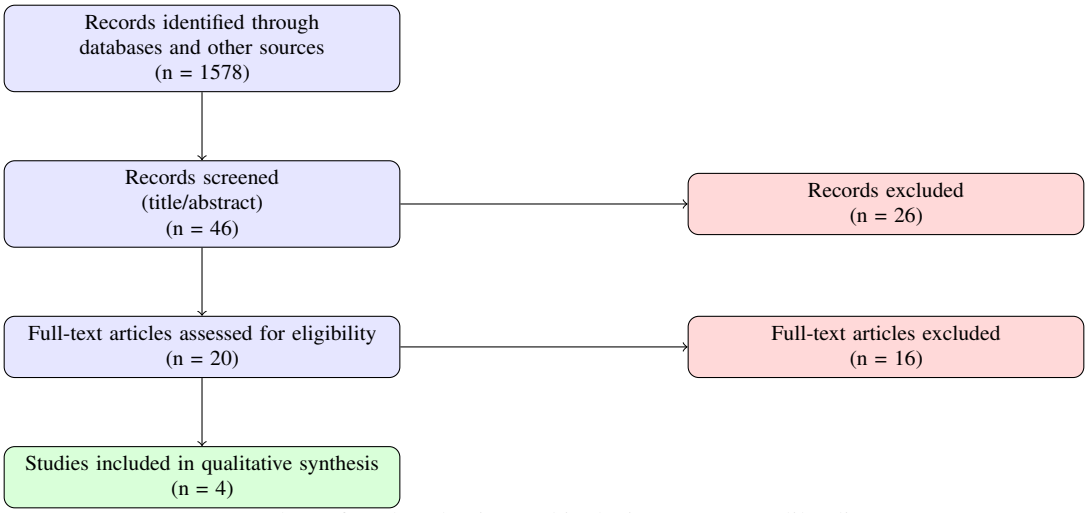

Figure 1. Flow of study selection and inclusion (PRISMA-like diagram).

yse these works in depth, characterising their security focus, tools and techniques, validation environments, and reported vulnerabilities, and contrasting defensive mechanisms with the comparatively scarce offensive approaches identified in the literature.

3. Results

3.1 Vulnerabilities in Hadoop

Security in Hadoop environments, though critical for maintaining the integrity of Big Data ecosystems, faces structural and operational challenges intrinsic to its distributed architecture. The analyzed studies [10], [1], and [11] highlight systemic vulnerabilities such as the centrality of the NameNode (a preferred target for DDoS attacks), weak authentication in components like YARN and Ambari, and the exposure of unencrypted services. These weaknesses—further aggravated by poor operational practices (e.g., use of default credentials)—enable attacks such as SYN floods, XSS, and privilege escalation.

Proposed solutions—ranging from Kerberos and Apache Ranger to active-standby redundancy and network segmentation—emphasize the need to integrate robust technology, proactive governance, and continuous pentesting to mitigate risks in environments increasingly exposed to sophisticated threats.

The study by Bhatal and Singh [1] provides a comprehensive analysis of the inherent vulnerabilities in the Hadoop ecosystem, highlighting technical flaws, misconfigurations, and operational risks in critical components such as HDFS, MapReduce, and YARN. The authors observe that Hadoop's distributed architecture, while efficient for Big Data processing, introduces systemic vulnerabilities—particularly due to the lack of built-in security features in early versions of the platform. Among the most significant issues are weak authentication mechanisms, exposure of unencrypted internal services, and an overreliance on so-called "trusted" environments, which facilitate attacks such as denial of service (DoS), cross-site scripting (XSS), and privilege escalation.

The article highlights specific CVEs, such as CVE-2018-11766, which allows arbitrary command execution as root by unprivileged users within YARN, and CVE-2017-3162, which relates to improper parameter validation in HTTP requests to the NameNode, thereby exposing HDFS metadata to injection attacks. Additionally, the authors experimentally demonstrate the impact of a SYN flood attack targeting the NameNode web interface (port 50070) using the hping3 tool. In a virtualized environment (Cloudera on CentOS), the attack increased the average MapReduce job processing time from 129.25 seconds (under normal conditions) to 265.113 seconds, highlighting the system's vulnerability to coordinated disruptions [1].

As mitigation strategies, the study proposes the adoption of Kerberos for multi-factor authentication, integrated with tools such as Apache Knox (to obfuscate critical endpoints) and Apache Ranger (for fine-grained access control). Additional measures include internal network segmentation, firewall implementation to filter non-essential traffic, and strict enforcement of patch management policies—particularly for web components such as Hue, which is vulnerable to XSS attacks. The results reinforce the need for proactive approaches, such as replicating the NameNode in active-standby mode and encrypting data both at rest and in transit, to reduce the attack surface in Big Data environments [1].

In summary, the article [1] demonstrates that vulnerabilities in Hadoop are not solely technical in nature but also stem from negligent operational practices, such as maintaining default passwords and exposing critical ports unnecessarily. The convergence of technological solutions (e.g., Kerberos and Knox) and robust security administration is identified as essential to mitigating risks at scale.

The study by Samet, Aydin, and Toy [11] presents a detailed analysis of the weaknesses present in Hadoop environments, highlighting both the vulnerabilities inherent to its distributed architecture and the mechanisms and tools that can be employed to identify and exploit such flaws. The authors begin by noting that Hadoop, having been originally designed without a strong focus on security, incorporates critical gaps, particularly in its authentication and authorization mechanisms. The absence of a consolidated security model allows malicious actors to exploit multiple attack vectors, such as unauthorized access to the Hadoop Distributed File System (HDFS) through interfaces like Remote Procedure Call (RPC) and HTTP, enabling the reading, modification, or deletion of sensitive data.

This situation is further exacerbated by the fact that critical components—such as Ambari, which is commonly used for cluster management—often rely on weak credentials, making them susceptible to dictionary attacks using tools like Hydra. In such cases, attackers can generate password lists with utilities such as crunch, enabling the brute-force compromise of administrative accounts. Once Ambari is breached, the attacker can exploit pre-established SSH connections using clients like PuTTY to execute commands directly on the node operating the HDFS, thereby expanding the scope of the compromise.

Additionally, the article underscores the importance of Hadoop's internal communication mechanisms, highlighting that data transmission between DataNodes via the streaming pipeline can be intercepted and manipulated in the absence of robust security protocols. This vulnerability, compounded by the lack of encryption in communication channels, exposes the environment to sniffing and interception attacks, facilitating the reading of data blocks and compromising the system's integrity.

To mitigate such vulnerabilities, the authors emphasize the need to adopt a suite of tools and security mechanisms, including the implementation of Kerberos for strict authentication control—ensuring that only authorized users and nodes have access to the cluster—and the use of Apache Ranger, which provides centralized policy management for protecting data at both the database and file system levels. Additionally,

solutions such as Apache Knox and Project Rhino are presented as alternatives to obscure network topology and offer unified platforms for authentication and authorization, respectively, thereby strengthening defenses against the exploitation of vulnerabilities [11].

In the context of penetration testing, the study demonstrates how brute-force attacks can be performed using Hydra, powered by wordlists generated with crunch, to crack Ambari credentials and gain privileged access. Once the administrative account is compromised, the attacker can use PuTTY to establish SSH connections and manipulate the HDFS—revealing how easily security flaws can be exploited in a minimally secured configuration (Level 0), where the absence of control mechanisms enables arbitrary command execution. Other notable vulnerabilities include the misuse of Oozie, a workflow management tool, which can be exploited to submit fraudulent jobs on behalf of other users, further highlighting the need for stricter access control policies [11].

The study concludes that, although Hadoop is highly valued for its scalability, flexibility, and ability to process large volumes of data, its security model must be fundamentally rethought. The integration of strong authentication mechanisms such as Kerberos, along with tools for monitoring and access policy management like Apache Ranger, as well as strategies for data and communication encryption, are essential to mitigate risks.

Similarly, awareness of penetration testing techniques and the use of tools such as Hydra, crunch, and PuTTY demonstrate how easily an attacker can exploit vulnerabilities if proper safeguards are not in place. The analysis leads to the conclusion that securing a Hadoop environment depends not only on the implementation of advanced technologies but also on an integrated approach that incorporates security by design and throughout the system's operational lifecycle—preventing its capabilities from becoming risk vectors for the organizations that rely on it [11].

The study by Shakeel Ahmad, Amanullah Yasin, and Qaisar Shafi [10] explores the vulnerabilities of the Hadoop environment, highlighting that the platform's architecture, originally designed without a strong security component, exhibits multiple weaknesses, particularly in terms of service availability. The authors demonstrate that the NameNode, which is responsible for metadata management, represents the critical component and single point of failure in the system. Its potential exploitation could lead to significant and irreparable interruptions in the operation of the cluster.

In standalone and pseudo-distributed configurations, where data read, write, and maintenance processes are centralized or lack sufficient redundancy, a Denial-of-Service (DoS) or Distributed Denial-of-Service (DDoS) attack targeting the NameNode can result in the complete shutdown of Hadoop services.

Beyond the theoretical and experimental analysis of this vulnerability, the study presents a detailed and practical approach to test execution, employing various components and tools to build the analysis environment. To validate the attack scenarios and mitigation strategies, VMWare Workstation 9.0 was used to create virtualized environments, along with Ubuntu 14.04 as the operating system, and Java JDK 1.7 to run Apache Hadoop 2.7.1. These elements, combined with YARN 2.0/MapReduce and the implementation of JournalNode and Zookeeper modules, enabled the construction of a testbed capable of realistically simulating DDoS scenarios aimed at compromising service availability [10].

From this same perspective, the study integrates a comprehensive set of penetration testing and monitoring tools, namely LOIC, HOIC, and TRIN00, which were used to execute UDP, HTTP, and TCP flood attacks—simulating offensive strategies that an attacker could deploy to saturate the resources of the NameNode and DataNodes. Additionally, tools such as Bmon, IPTraf, TCPTracker, and Slurm were employed to monitor node traffic and load in real time, providing detailed insights into the effects of the attacks and enabling the evaluation of mitigation strategies.

This integration of offensive and monitoring tools, combined with redundancy and failover mechanisms provided by JournalNode and Zookeeper, underscores the importance of a systemic and multi-layered approach to protecting Hadoop environments, particularly with respect to mitigating attacks targeting service availability [10].

In summary, the analysis demonstrates that although data replication across DataNodes enhances the operational resilience of the environment, the critical dependence on the NameNode represents a key vulnerability which, if exploited, could bring the entire Hadoop service to a halt. Therefore, the integration of high-availability solutions and the systematic use of penetration testing tools are essential for identifying weaknesses and implementing effective countermeasures—ultimately ensuring the robustness and operational continuity of Big Data platforms in the face of increasing threats [10].

3.2 Vulnerabilities in Spark

In real-time data processing systems such as Apache Spark, fault recovery efficiency is often prioritized at the expense of robust security mechanisms. The study by Tian et al. [12] exemplifies this imbalance by exposing a critical vulnerability in the Spark Streaming checkpoint mechanism: the absence of authentication during data recovery allows users with access to shared directories (e.g., HDFS) to tamper with critical files, thereby compromising job integrity and output accuracy. This flaw—classified under Insecure Permissions—underscores the urgent need to combine strict access control policies with proactive penetration testing practices to ensure that fault tolerance solutions do not become attack vectors in collaborative environments.

Tian et al. [12] explore a critical vulnerability in Spark Streaming's checkpoint mechanism, identifying the lack of authentication during recovery procedures in fault-tolerant scenarios. This gap enables malicious users within the same

group—who have access to the checkpoint directory in systems like HDFS—to replace legitimate files with compromised versions, interfering with job execution and potentially leading to data loss or inaccurate results. The authors demonstrate that in collaborative environments, the sharing of directories without consistent verification of the AppID allows for the injection of fraudulent checkpoints, which are improperly used during fault recovery [12] . This vulnerability, falling within the Insecure Permissions category, reveals an operational weakness in Spark deployments that prioritize efficiency over robust security controls.

The relevance of this discovery for penetration testing lies in the need to validate not only Spark's logical configuration, but also the access policies governing underlying storage resources such as HDFS. Permission analysis tools like Apache Ranger or Cloudera Navigator could be applied to detect the absence of authentication in critical directories, mapping exposure to checkpoint replacement attacks. Additionally, offensive techniques based on intrusion simulation—for example, using custom scripts to manipulate files within HDFS—could reproduce the attack scenarios described in the study, validating the exploitability of the vulnerability in controlled environments. This approach aligns with penetration testing methodologies that emphasize the exploitation of configuration flaws in distributed systems [12] .

To mitigate this risk, Tian et al. [12] propose a solution structured into three modules: locker, monitor, and filter. The first module ensures temporary write-locking of permissions during job execution, while the second monitors the creation of unauthorized UUID directories, issuing alerts for suspicious activity. The third module, implemented via filtering scripts, guarantees that only legitimate checkpoints are considered during data recovery. Although effective, this solution requires integration with continuous monitoring tools to enable real-time anomaly detection—an aspect that the authors do not explore in depth [12] .

Although the article does not explicitly reference formal penetration testing methodologies, the simulation of attacks through malicious jobs and the manual manipulation of checkpoints reflect an empirical approach to vulnerability validation. This gap opens space for discussion on the integration of structured platforms—such as the MITRE ATT&CK framework—into threat analysis for Spark environments, enabling the mapping of attack tactics (e.g., Credential Access, Data Manipulation) to specific ecosystem vulnerabilities. The absence of CVEs (Common Vulnerabilities and Exposures) in the study does not diminish its contribution, as the detailed technical description of the vulnerability provides a replicable model for custom intrusion tests tailored to Big Data architectures.

In summary, the research by Tian et al. [12] underscores the importance of aligning native fault-tolerance mechanisms with strict authentication protocols, highlighting the interdependence between application-level security and the configuration of underlying infrastructure. The integration of specialized penetration testing tools—capable of operating in distributed and scalable environments—becomes crucial for identifying and remediating similar vulnerabilities, thus ensuring the integrity of real-time data processing workflows. This study, therefore, serves as a catalyst for the adoption of proactive security practices in Spark deployments, balancing performance and robustness.

3.3 Comparison of Selected Studies

The comparative analysis of the four selected studies, summarized in Table 3, reveals several consistent patterns that illustrate both the current maturity level and the methodological fragmentation of research on offensive security in Big Data environments. While all works examined share the goal of exposing vulnerabilities in Hadoop or Spark ecosystems, their approaches vary significantly in terms of technical depth, reproducibility, and scope of validation.

A first and most striking observation concerns the superficial nature of the offensive techniques employed. Each study focuses on isolated attack scenarios—such as distributed denial-of-service (DDoS) attacks targeting the NameNode, brute-force intrusions through weak Ambari credentials, or manipulation of Spark Streaming checkpoints—but rarely provides sufficient procedural detail to enable replication. Descriptions of attack vectors are often limited to high-level overviews, with only partial or missing documentation of tools, configurations, or execution parameters. As a result, the experiments cannot be easily reproduced or compared, which limits their value for cumulative scientific progress in this domain. This lack of methodological transparency underscores the absence of standardized guidelines for conducting penetration testing in distributed data frameworks.

The second pattern involves the reliance on generic security and network tools rather than specialized platforms tailored for Big Data architectures. Utilities such as LOIC, HOIC, Hydra, Nmap, and Wireshark are frequently cited across studies, yet they are designed for traditional network or system testing and not optimized for the layered and service-oriented nature of Hadoop and Spark clusters. None of the analyzed works adapt these tools to exploit Big Data–specific vulnerabilities, such as misconfigurations in YARN resource managers, insecure RPC channels, or exposure of HDFS web interfaces. This observation reinforces the need for domain-specific pentesting toolkits capable of simulating realistic adversarial behavior within multi-node, data-intensive environments.

A third and equally important trend is the imbalance between Hadoop and Spark coverage. The reviewed literature is dominated by studies addressing Hadoop, which benefits from a longer research history and greater availability of open testbeds. In contrast, Spark—despite its widespread adoption in modern data processing pipelines—receives far less attention, with only one study directly targeting vulnerabilities in Spark Streaming. This asymmetry may stem from Spark's relative architectural complexity, the scarcity of reproducible

exploitation datasets, and the difficulty of designing controlled experiments in real-time processing contexts. Nevertheless, the lack of empirical evidence on Spark's security posture constitutes a critical blind spot, given its growing prevalence in both academic and industrial applications.

Beyond these three main patterns, additional cross-study insights can be drawn. None of the reviewed works integrate a comprehensive testing framework encompassing reconnaissance, exploitation, post-exploitation, and mitigation phases—stages commonly formalized in classical penetration testing methodologies such as OSSTMM or PTES. Moreover, the studies tend to evaluate vulnerabilities in isolation, without considering interdependencies between components (e.g., how an Ambari breach may escalate to HDFS manipulation). This compartmentalized approach prevents the identification of systemic risks that emerge from the orchestration of multiple services, which is precisely the hallmark of Big Data architectures.

The comparison of selected studies reveals a field that is still in its early stages of methodological development. Offensive investigations into Hadoop and Spark remain exploratory, fragmented, and heavily dependent on conventional security tools. The absence of unified testing frameworks, coupled with limited reproducibility and the narrow focus on individual vulnerabilities, highlights the need for more comprehensive and standardized approaches to penetration testing in Big Data systems. These findings provide the foundation for the critical discussion presented in Section 4, where methodological gaps and future research directions are examined in greater detail.

4. Discussion

4.1 Critical Analysis of the Studies

The comparative examination of the selected studies reveals a fragmented and uneven landscape in research on offensive security for Big Data systems. Most works focus on isolated attack scenarios—such as distributed denial-of-service (DDoS) assaults on the Hadoop NameNode, brute-force credential attacks on Ambari, or checkpoint manipulation in Spark Streaming—without proposing comprehensive or standardized penetration testing methodologies tailored to distributed architectures. As a result, the literature remains predominantly descriptive, offering partial insights rather than a coherent methodological framework.

A recurring pattern is the predominance of defensive perspectives. The majority of studies concentrate on mitigation strategies such as Kerberos authentication, Apache Ranger authorization policies, or network-level segmentation, while the offensive dimension—aimed at evaluating the actual exploitability of vulnerabilities—receives limited attention. This imbalance prevents a systematic comparison between defensive mechanisms and the real threats they are meant to counter. Furthermore, the lack of standardized protocols or shared benchmarks for offensive testing makes it difficult to assess whether the proposed security controls are effective under adversarial conditions.

Another critical shortcoming concerns the scarcity of methodological detail. Most studies omit essential information such as the specific attack commands used, the software versions tested, or the configuration of the experimental clusters. In many cases, system topologies and dataset characteristics are described only superficially, rendering the experiments effectively irreproducible. This deficiency weakens the reliability of findings and hinders cumulative progress, as future researchers cannot replicate or extend the analyses on equivalent platforms.

The third notable gap lies in the uneven distribution of research coverage. Hadoop has received significantly more attention, likely due to its earlier adoption and broader use in academia and industry. Spark, on the other hand, remains underexplored, particularly regarding vulnerabilities in its real-time processing modules such as Spark Streaming. This disparity limits the understanding of how newer frameworks behave under offensive conditions and leaves a critical gap in assessing the evolving security posture of the Big Data ecosystem.

These methodological and thematic gaps reflect deeper structural challenges inherent to penetration testing in distributed systems. Setting up realistic multi-node clusters demands considerable computational and networking resources, often beyond the reach of academic research environments. Additionally, organizations are typically reluctant to disclose or test vulnerabilities in production systems due to confidentiality, compliance, and reputational risks. Finally, the lack of specialized tools exacerbates these limitations: widely used platforms such as Metasploit or Nmap were not designed for multi-layered, service-oriented Big Data architectures and therefore cannot model realistic attack surfaces effectively.

Overcoming these challenges requires a shift toward purpose-built offensive security infrastructures for Big Data. Dedicated pentesting platforms capable of orchestrating distributed attack simulations, realistic testbeds populated with representative datasets, and standardized evaluation metrics—such as detection latency, impact severity, and mitigation effectiveness—are essential steps toward methodological maturity in this field. Establishing such foundations would enable the research community to move beyond ad hoc demonstrations and toward reproducible, quantitatively driven assessments of security in large-scale data frameworks.

4.2 Penetration Testing Tools Adapted for Big Data

Traditional penetration testing tools such as Nmap, Hydra, and Wireshark are widely used across the analyzed studies, yet their application remains generic and not adapted to the architectural complexity of Hadoop and Spark. While effective for network scanning, credential brute-forcing, and traffic inspection, these tools fail to capture the distributed and service-oriented nature of Big Data environments. Consequently, most experiments rely on ad hoc scripts or isolated

Table 3. Comparison of Selected Studies

Criteria	Study 1 (DDoS)	Study 2 (Hadoop)	Study 3 (Ambari UI)	Study 4 (Spark Streaming)
CVEs/vulns cited	Yes (NameNode)	Yes (e.g., YARN/NameNode)	Yes (Ambari UI)	Yes (checkpoint)
Attack tools used	LOIC, HOIC	Nmap, Wireshark, hping3	Hydra, Nmap	LOIC, HOIC, TRIN00, Bmon
Offensive techniques detail	Superficial	Partial	Superficial	Superficial
Mitigations tested	Hadoop 2.7.1, YARN	Cloudera/CentOS	Ambari Sandbox 2.6.4 (assumed)	Spark 1.6.0 cluster
Target platform	Hadoop MapReduce	Hadoop MapReduce	Ambari UI/MapReduce	Spark Streaming

attack simulations that offer limited insight into systemic vulnerabilities.

Hadoop and Spark operate through multiple interdependent components—HDFS, YARN, MapReduce, and Spark Executors—each introducing specific exposure points, such as unsecured APIs or misconfigured web interfaces. Conventional tools can identify open ports or weak credentials but lack the contextual awareness to interpret distributed authentication flows, inter-node communication, or workload orchestration. This limitation prevents the execution of coordinated, multi-vector attack scenarios representative of real-world cluster behavior.

Addressing these constraints requires the development of specialized penetration testing frameworks capable of enumerating cluster topologies, analyzing configuration files (e.g., `core-site.xml`, `yarn-site.xml`), and simulating distributed exploit propagation. Integrating such tools with orchestration technologies like Ansible or Kubernetes would enable automated and reproducible attack environments. Moreover, coupling these frameworks with monitoring systems such as Apache Ranger or Prometheus could facilitate real-time observation of attack impact and mitigation effectiveness.

4.3 Methodological and Replicability Challenges

In addition to the selection limitations previously noted (see Section 2.7), we identified specific challenges related to replicating penetration testing experiments in Big Data environments:

- Lack of method standardization: Many articles fail to detail attack parameters, such as the exact version of tools, commands used, and payloads;
- Insufficient information about test environments: Cluster configurations, input data, and network topology are often incompletely described;
- Lack of comparable metrics and platforms: There are no shared KPIs (e.g., detection time, mitigation overhead), nor a unified repository of Big Data-specific CVEs/exploits;
- Prevalence of conceptual studies: Without practical implementation, it becomes difficult to reproduce or validate results in real or emulated environments.

These limitations must be considered when interpreting the findings and should serve as a foundation for improving future methodologies—those aimed not only at identifying vulnerabilities but also ensuring the replicability of experimental results.

5. Practical Implications and Future Research

The analysis of vulnerabilities identified in platforms such as Hadoop and Spark reveals not only theoretical challenges but also significant practical implications for Big Data environments. For instance, the exposure of critical flaws in core components can compromise the integrity and availability of data in sectors such as healthcare and finance.

In this context, it is recommended that future research explore offensive approaches—such as penetration testing—more effectively, treating them as diagnostic tools. The development of specialized tools capable of simulating attacks in real operational environments could contribute to:

- Improving authentication and encryption mechanisms tailored to distributed architectures;
- Integrating automated intrusion testing for continuous risk assessment;
- Creating protocols that enable the simulation of coordinated attacks in Big Data clusters, thereby providing more robust data to inform security policy development.

This reflection not only broadens the understanding of risks but also guides research efforts toward practical solutions that enhance the security posture of Big Data environments.

6. Related Work

While the majority of Big Data security research has concentrated on preventive strategies, there is a growing body of work that addresses offensive security techniques and penetration testing within distributed platforms like Hadoop and Spark.

Kang et al. [13] proposed a cybersecurity testbed tailored for Hadoop-based environments to simulate attacks and evaluate system behavior under stress. Their study emphasizes the importance of real-world test environments for validating security controls.

Amir Latif et al. [14] explored adversarial threat modeling for Hadoop systems, proposing a layered attack taxonomy and simulating vulnerabilities across various Hadoop components. Their work provides valuable insight into how complex multi-stage attacks could propagate in distributed environments.

Siddiqui and Uzmi [15] conducted a Spark-based assessment using custom scripts and modified exploitation tools to assess system vulnerabilities in memory handling and job scheduling. Their methodology bridges the gap between theoretical vulnerabilities and practical, reproducible attacks.

Mahmood et al. [16] designed a Spark-Sandbox intrusion simulation to validate open CVEs on cluster nodes. Their findings highlighted how lack of isolation and container-level weaknesses could be exploited to compromise Spark clusters in public and hybrid cloud deployments .

Xu and Li [3] developed an intelligent software security framework built on Spark Streaming to actively detect and exploit design flaws via penetration testing modules, representing one of the few studies incorporating automated offensive analytics into Big Data platforms.

These studies collectively demonstrate the emergence of offensive security as a vital aspect of Big Data protection. However, they also reveal methodological fragmentation, lack of standardization in testing environments, and under representation of cross-platform (Hadoop and Spark) comparative penetration testing. This study addresses these gaps through a systematic review that consolidates offensive approaches across both ecosystems, assessing their efficacy, reproducibility, and adaptability to real-world attack scenarios.

7. Conclusions

This systematic review identified a critical gap in the literature: the scarcity of studies addressing vulnerabilities in Hadoop and Spark from a technical penetration testing perspective. Despite the growing importance of security in Big Data platforms, the available studies predominantly focus on defensive approaches, such as access control, encryption, and strong authentication. In contrast, offensive approaches—including practical CVE exploitation, automated testing, and real-world attack simulations—remain underexplored and poorly documented. This gap significantly limits our understanding of the attack surface and the potential malicious exploitations in these environments. Moreover, the lack of detailed methodologies and specific penetration testing tools for Hadoop and Spark hinders the replicability of experiments and the validation of proposed security measures. As a result, there is a considerable risk that current defensive solutions are being developed without a comprehensive understanding of the real threats these systems face. The imbalance in platform coverage is also evident. While Hadoop appears more frequently in the reviewed studies, Spark remains underrepresented. This disparity may be attributed to Spark's more recent adoption or the lower maturity of available analysis tools targeting that platform.

7.1 Recommendations for Future Work

Based on the gaps identified throughout this review, the following research directions are proposed:

- Pentesting tools tailored to Big Data: Develop platforms specifically adapted to distributed environments such as Hadoop and Spark, supporting multi-node analysis, malicious job simulation, and automated CVE exploitation.
- Realistic testing environments: Create Hadoop/Spark clusters with representative data and real-world configurations (using VMs or cloud-based infrastructures) to enable practical and controlled penetration testing.
- Standardization of offensive methodologies: Define reproducible attack protocols (including parameters, software versions, and cluster topologies), based on frameworks such as MITRE ATT&CK adapted for Big Data environments.
- Evaluation metrics for effectiveness: Establish offensive security indicators (e.g., time to detection, data impact, mitigation effectiveness) that enable comparability across studies.
- Open repositories and exploit sharing: Create collaborative databases to register Hadoop- and Spark-specific CVEs and their verified exploitations, in order to accelerate the technical community's response.

In summary, the findings of this review underscore the urgent need for more structured approaches to penetration testing in Hadoop and Spark. By addressing the identified gaps, future research can make significant contributions to the security of Big Data systems, ensuring their resilience in the face of emerging threats.

Author Contributions

FM: initial manuscript and investigation; LB: supervision, revision, translation, and extended version; EL: supervision and revision.

References

[1] BHATHAL, G.; SINGH, A. Big data: Hadoop framework vulnerabilities, security issues and attacks. *Array*, Elsevier, v. 1, 2019. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S2590005619300025⟩.

[2] SPIVEY, B.; ECHEVERRIA, J. *Hadoop Security: Protecting your big data platform*. O'Reilly Media, 2015. Disponível em: ⟨https://books.google.com/books?id=VXEJCgAAQBAJ⟩.

[3] XU, C.; LI, J. Design of intelligent software security system based on spark big data computing. *Wireless Personal Communications*, Springer, 2025. Disponível em: ⟨https://link.springer.com/article/10.1007/s11277-024-11015-4⟩.

[4] KAUSHIK, P.; RATHORE, S.; RATHORE, R. Big data-powered analytics for fortifying virtualized infrastructure security in the cloud. Springer, 2023. Disponível em: ⟨https://link.springer.com/chapter/10.1007/978-3-031-80778-7_12⟩.

[5] HART, M.; DAVE, R.; RICHARDSON, E. Next-generation intrusion detection and prevention system performance in distributed big data network security architectures. *International Journal of Emerging Technology and Advanced Engineering*, 2023. Disponível em: ⟨https://search.proquest.com/openview/2b773d51151376afb674268b2d505ff8⟩.

[6] MA, L. Research on vulnerability exploitation and detection technology based on big data analysis. IEEE, 2021. Disponível em: ⟨https://ieeexplore.ieee.org/document/9699917/⟩.

[7] RAHMAN, M.; SHAHRIAR, H. Clustering enabled robust intrusion detection system for big data using hadoop–pyspark. In: *IEEE Conference on Improving Quality of Life using AI*. [s.n.], 2023. Disponível em: ⟨https://ieeexplore.ieee.org/document/10374747/⟩.

[8] BAGUI, S. et al. Introducing uwf-zeekdata22: A comprehensive network traffic dataset based on the mitre att&ck framework. *Data*, MDPI, v. 8, n. 1, 2023. Disponível em: ⟨https://www.mdpi.com/2306-5729/8/1/18⟩.

[9] BARATA, L.; MARTINS, F.; LOPES, E. *Security Assessment in Big Data Frameworks: Systematic Review Dataset*. 2025. ⟨https://doi.org/10.5281/zenodo.19686185⟩. Zenodo dataset.

[10] AHMAD, S.; YASIN, A.; SHAFI, Q. Ddos attacks analysis in bigdata (hadoop) environment. In: *Proceedings of the 2018 15th International Bhurban Conference on Applied Sciences & Technology (IBCAST)*. Islamabad, Pakistan: IEEE, 2018. p. 495–501. Disponível em: ⟨https://ieeexplore.ieee.org/document/8312270⟩.

[11] SAMET, R.; AYDIN, A.; TOY, F. Big data security problem based on hadoop framework. In: *Proceedings of the 2019 4th International Conference on Computer Science and Engineering (UBMK)*. Samsun, Turkey: IEEE, 2019. p. 525–530. Disponível em: ⟨https://ieeexplore.ieee.org/document/8907074⟩.

[12] TIAN, Y. et al. Non-authentication based checkpoint fault-tolerant vulnerability in spark streaming. In: *Proceedings of the 2018 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. IEEE, 2018. p. 783–786. Disponível em: ⟨https://ieeexplore.ieee.org/document/8538745⟩.

[13] KANG, J.; LEE, H.; KIM, D. A cybersecurity testbed for evaluating penetration attacks in hadoop ecosystems. In: *2021 IEEE International Conference on Big Data Security on Cloud*. IEEE, 2021. p. 57–64. Disponível em: ⟨https://ieeexplore.ieee.org/document/9441082⟩.

[14] LATIF, A.; ABBAS, H.; KHAN, A. Adversarial threat modeling and simulation for hadoop security assessment. *Journal of Information Security and Applications*, Elsevier, v. 54, p. 102526, 2020. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S221421262030059X⟩.

[15] SIDDIQUI, M.; UZMI, S. Offensive security assessment of apache spark: A penetration testing approach. In: *Proceedings of the 2020 International Conference on Cyberworlds*. IEEE, 2020. p. 45–52. Disponível em: ⟨https://ieeexplore.ieee.org/document/9287311⟩.

[16] MAHMOOD, K.; ZHOU, Z.; LYU, M. R. Sandbox-based vulnerability simulation for apache spark security evaluation. *Future Generation Computer Systems*, Elsevier, v. 127, p. 123–137, 2022. Disponível em: ⟨https://doi.org/10.1016/j.future.2021.09.020⟩.

1. Appendix

A.1 Articles screened by title and abstract

Table 4 lists the 46 articles that passed the initial title and abstract screening, including title, year, DOI, and screening decision. The corresponding metadata and screening decisions are also available in the `articles` sheet of our public dataset in Zenodo (DOI: 10.5281/zenodo.19686185) [9], which additionally records the database source, query identifier, and screening notes in machine-readable form.

Table 4. Records screened at title/abstract level (n=46) and corresponding screening decisions.

ID	Title (shortened)	Year	DOI	Decision
1	Research on the application of OpenStack to build a new heterogeneous real-time virtual cloud to reproduce application vulnerability and training demonstration architecture	2017	10.1109/ITNEC.2017.8285007	excluded
2	Non-Authentication Based Checkpoint Fault-tolerant Vulnerability in Spark Streaming	2018	10.1109/ISCC.2018.8538745	included
3	Security framework using Hadoop for big data	2017	10.1109/CCAA.2017.8229813	included
4	Clustering Enabled Robust Intrusion Detection System for Big Data Using Hadoop–PySpark	2023	10.1109/HONET59747.2023.10374747	included
5	BDF-SDN: A Big Data Framework for DDoS Attack Detection in Large-Scale SDN-Based Cloud	2021	10.1109/DSC49826.2021.9346269	excluded
6	HydraDB: a resilient RDMA-driven key-value middleware for in-memory cluster computing	2015	10.1145/2807591.2807614	excluded
7	Method for Mining Security Vulnerabilities of Data Storage of Electric Power Internet of Things Based On Spark Framework and RASP Technology	2022	10.1109/ICKECS56523.2022.10059626	excluded
8	Identification of Threats and Vulnerabilities in Public Cloud-Based Apache Hadoop Distributed File System	2019	10.1109/ICENCO48310.2019.9027300	included
9	A survey on miscellaneous attacks in Hadoop framework	2018	10.1109/ICISC.2018.8398961	included
10	Research on Hadoop-based Intrusion Detection by IP Address Mining	2020	10.1109/ICBAIE49996.2020.00078	excluded
11	Current security threats and prevention measures relating to cloud services, Hadoop concurrent processing, and big data	2015	10.1109/BigData.2015.7203960	included
12	Research on Vulnerability Exploitation and Detection Technology Based on Big Data Analysis	2021	10.1109/IAAI54625.2021.9699917	excluded
13	A review on security measures of Hadoop	2017	10.1109/ICICCIS.2017.8660887	included
14	Efficient k-means Using Triangle Inequality on Spark for Cyber Security Analytics	2019	10.1145/3309182.3309187	excluded
15	An Efficient Approach to Improve Security for MapReduce Computation in Cloud System	2018	10.1145/3230905.3230954	included
16	The Queen's Guard: A Secure Enforcement of Fine-grained Access Control In Distributed Data Analytics Platforms	2021	10.48550/arXiv.2106.13123	included
17	A System Architecture for the Detection of Insider Attacks in Big Data Systems	2016	10.48550/arXiv.1612.01587	included
18	Call Trace and Memory Access Pattern based Runtime Insider Threat Detection for Big Data Platforms	2016	10.48550/arXiv.1611.07392	included
19	BlockHDFS: Blockchain-integrated Hadoop distributed file system for secure provenance traceability	2021	10.1016/j.bcra.2021.100032	excluded
20	A Scalable Platform for Enabling the Forensic Investigation of Exploited IoT Devices and Their Generated Unsolicited Activities	2020	10.1016/j.fsidi.2020.300922	excluded
21	Distributed real-time SlowDoS attacks detection over encrypted traffic using Artificial Intelligence	2021	10.1016/j.jnca.2020.102871	excluded
22	Token-based authentication for Hadoop platform	2023	10.1016/j.asej.2022.101921	excluded
23	Big Data: Hadoop framework vulnerabilities, security issues and attacks	2019	10.1016/j.array.2019.100002	included
24	DDoS Attack Detection System using Apache Spark	2021	10.1109/ICCCI50826.2021.9457012	excluded
25	Method for Mining Security Vulnerabilities of Data Storage of Electric Power Internet of Things Based On Spark Framework and RASP Technology	2022	10.1109/ICKECS56523.2022.10059626	excluded
26	Identification of Threats and Vulnerabilities in Public Cloud-Based Apache Hadoop Distributed File System	2019	10.1109/ICENCO48310.2019.9027300	excluded
27	Real time cyber attack analysis on Hadoop ecosystem using machine learning algorithms	2015	10.1109/APWCCSE.2015.7476223	included
28	Research on Vulnerability Exploitation and Detection Technology Based On Big Data Analysis	2021	10.1109/IAAI54625.2021.9699917	excluded
29	Big Data Technology Application in Software Security Vulnerability Analysis	2023	10.1109/ICMIII58949.2023.00033	excluded
30	An Intrusion Detection System Based on Hadoop	2015	10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.162	excluded
31	Zero-Day Attack Identification in Streaming Data Using Semantics and Spark	2017	10.1109/BigDataCongress.2017.25	excluded
32	SEINA: A stealthy and effective internal attack in Hadoop systems	2017	10.1109/ICCNC.2017.7876183	included
33	A Hadoop Based Analysis and Detection Model for IP Spoofing Typed DDoS Attack	2016	10.1109/TrustCom.2016.0302	excluded
34	Attack Models for Big Data Platform Hadoop	2019	10.1109/BigDataSecurity-HPSC-IDS.2019.00037	included
35	DDoS attacks analysis in bigdata (hadoop) environment	2018	10.1109/IBCAST.2018.8312270	included
36	Haddle: A Framework for Investigating Data Leakage Attacks in Hadoop	2015	10.1109/GLOCOM.2015.7417387	included
37	Memory access pattern based insider threat detection in big data systems	2016	10.1109/BigData.2016.7841027	included
38	Security issues of big data applications served by mobile operators	2017	10.1109/SIU.2017.7960253	excluded
39	Big Data Security Problem Based On Hadoop Framework	2019	10.1109/UBMK.2019.8907074	included
40	SecHDFS: A Secure Data Allocation Scheme for Heterogenous Hadoop Systems	2016	10.1109/NAS.2016.7549419	excluded
41	Advanced resource management with access control for multitenant Hadoop	2015	10.1109/JCN.2015.000106	excluded
42	Data Block Management Scheme Based on Secret Sharing for HDFS	2015	10.1109/BWCCA.2015.70	excluded
43	Large-Scale Encryption in the Hadoop Environment: Challenges and Solutions	2017	10.1109/ACCESS.2017.2700228	excluded
44	Research on Vulnerability Exploitation and Detection Technology Based On Big Data Analysis	2021	10.1109/IAAI54625.2021.9699917	included
45	Musti: Dynamic Prevention of Invalid Object Initialization Attacks	2019	10.1109/TIFS.2019.2894356	excluded
46	Big data security: Requirements, challenges and preservation of private data inside mobile operators	2018	10.1109/BlackSeaCom.2017.8277711	excluded

A.2 Quality assessment of included and borderline studies

Table 5 reports the quality assessment scores for the 20 full-text articles evaluated in detail. The criteria correspond to the dimensions described in Section 2.7: vulnerability analysis, reproducibility, validity, and relevance. The aggregate score is used

to support, but not solely determine, the final inclusion decision.

Table 5. Quality assessment scores for included and borderline studies

Art. ID	Vulnerability	Reproducibility	Validity	Relevance	Total	Included final
2	2	1	2	2	7	yes
3	0	0	1	2	3	no
4	0	0	1	1	2	no
8	1	0	1	1	3	no
9	1	0	1	2	4	no
11	1	0	0	1	2	no
13	1	0	0	2	3	no
15	0	0	0	2	2	no
16	1	0	1	2	4	no
17	1	0	1	2	4	no
18	1	0	1	2	4	no
23	2	1	1	2	6	yes
27	1	0	1	2	4	no
32	1	0	2	2	5	no
34	1	2	2	2	7	no
35	2	2	2	2	8	yes
36	1	0	1	2	4	no
37	1	0	2	2	5	no
39	2	2	2	2	8	yes
44	1	0	0	2	3	no