Asian Journal of Information Science and Technology
ISSN: 2231-6108 (P); 3049-2661 (E)
Vol.15 No.1, 2025, pp.44-53

www.crijournals.org
DOI: https://doi.org/10.70112/ajist-2025.15.1.4332

Mitigation of Web Vulnerabilities Arising from Directory Brute-Forcing and Exposed Development Artifact: A Qualitative Study

Aminu Muhammad Auwal
Faculty of Natural Sciences, University of Jos, Plateau State, Nigeria
E-mail: i.elameenu@gmail.com
(Received 13 January 2024; Revised 27 February 2025; Accepted 15 March 2025; Available online 21 March 2025)

***Abstract* - Web applications increasingly face threats not only from sophisticated exploits but also from basic oversights such as misconfigured directories and exposed development artifacts. This study explores the awareness and mitigation strategies of developers, Dev Ops engineers, and system administrators regarding vulnerabilities arising from directory brute-forcing and the exposure of sensitive files such as .git/, .env, and .bash_history. Using a qualitative approach, data were collected through semi-structured interviews with 11 IT professionals across different sectors in Nigeria, where the rise of small- and medium-scale web deployments has amplified the security stakes. Findings reveal a concerning inconsistency in mitigation strategies, even among technically proficient participants. While some employ directory restrictions and CI/CD security checks, others rely on ad hoc, manual practices. Most participants were aware of the risks posed by exposed artifacts, yet only a few incorporated automated tools or vulnerability scanners into their deployment pipelines. Notably, a gap persists between theoretical knowledge and operational execution, leaving systems vulnerable to reconnaissance and chained attacks. This study highlights the need for stronger Dev Sec Ops integration, improved developer hygiene practices, and automated security enforcement within web deployment workflows. The results underscore a critical call to action for organizations and individual professionals to revisit their deployment pipelines and invest in proactive security measures that go beyond basic configuration.**
Keywords: **Web Application Security, Dev Sec Ops, Directory Brute-Forcing, Deployment Pipelines, Vulnerability Mitigation**

I. INTRODUCTION

Web application vulnerabilities continue to pose significant threats to organizational cybersecurity, with attackers increasingly targeting overlooked or misconfigured elements of server infrastructure. Among these, the exposure of sensitive directories and residual artifacts-such as .git folders, .bash_history, and CI/CD configuration files-presents an under-addressed yet critical vector for exploitation (e.g., [1], [2]).

These files often exist outside standard navigation paths and may return HTTP 403 or 401 errors without fully restricting access. When discovered via directory brute-forcing or sub domain enumeration, they can leak information about internal systems, credentials, deployment logic, or even source code history [3]. Automated tools such as Go buster and FFUF have made the discovery of such exposures trivial for even moderately skilled attackers [4].As organizations accelerate Dev Ops practices and frequent deployments, the risk of exposing temporary or legacy files increases, particularly when security reviews lag behind development cycles. Studies show that many Dev Ops pipelines lack adequate safeguards against publishing sensitive build or deployment artifacts [5]. Despite increased awareness in the cybersecurity community, there remains limited research on the operational awareness and mitigation strategies adopted by administrators, developers, and Dev Ops engineers in relation to directory and artifact exposure. This study investigates this gap through qualitative analysis, aiming to uncover the behavioral, procedural, and tooling inconsistencies that leave web infrastructure vulnerable to such probing.

II. LITERATURE REVIEW

A. Understanding Web Application Vulnerabilities and Attack Vectors

Web applications have become indispensable in modern society, facilitating everything from e-commerce to critical infrastructure management. However, this ubiquity also makes them prime targets for malicious actors. A foundational understanding of web application vulnerabilities is crucial for developing robust security postures. The Open Web Application Security Project (OWASP) Top 10 consistently highlights prevalent risks, including injection flaws, broken authentication, and insecure deserialization, each posing significant threats to data confidentiality, integrity, and availability [6], [7].

A common initial phase for attackers is reconnaissance, where they gather information about a target system to identify potential weaknesses. This often involves mapping the application's structure, discovering hidden paths, and enumerating accessible resources. Effective reconnaissance can lead to more focused attacks, making early detection and mitigation of information exposure a critical component of secure web application design [6], [8].

B. Directory Brute-Forcing and Enumeration

Directory brute-forcing and enumeration are potent reconnaissance techniques used by attackers to discover

hidden directories, files, and resources on a web server that are not typically linked or publicly advertised. Tools such as Gobuster, Dirb, and Feroxbuster automate this process, systematically guessing common directory and file names to identify accessible endpoints [9], [10]. The goal is to uncover sensitive information, administrative interfaces, backup files, or misconfigured resources that can lead to further exploitation [11].

The impact of successful enumeration can be severe. Discovering an unlinked administrative panel, for instance, could lead to unauthorized access if default credentials are in use or if authentication bypass vulnerabilities exist. Similarly, finding old backup files or configuration files can expose sensitive data, internal network structures, or even credentials that attackers can leverage for privilege escalation or lateral movement within a network [9], [10], [12]. The evolution of these techniques has paralleled the growth of web applications, making them a persistent threat that requires proactive mitigation [9], [10], [13].

C. Exposure of Sensitive Developer Artifacts and Misconfigurations

A particularly insidious aspect of web vulnerability stems from the unintentional exposure of sensitive development artifacts and misconfigurations. These exposures often occur due to oversight, misconfigured web servers, or inadequate deployment practices, leaving critical internal information accessible to the public internet. Common examples include .git repositories, .env files containing environment variables and credentials, .bash_history files (which can reveal commands executed on a server), docker-compose.yml files (detailing container configurations), backup files, and uncompiled source code [14].

The risk is profound: source code disclosure can reveal proprietary logic and vulnerabilities, while exposed credentials or configuration files can grant direct access to databases, APIs, or internal systems. The causes of such exposures are multifaceted, frequently stemming from a lack of proper .gitignore usage in development workflows, incorrect web server configurations (e.g., enabling directory listing in Apache or Nginx), flawed deployment scripts that fail to sanitize assets, or simply pushing development-specific files to production environments. Real-world incidents have repeatedly demonstrated that these seemingly minor oversights can lead to significant data breaches and system compromises [15]. While technical means to prevent these exposures exist, the persistent occurrence of such vulnerabilities highlights a deeper problem related to human practices and the implementation of security controls in the development and operations lifecycle [16].

D. Human Factors, Awareness, and Dev Sec Ops Practices

Despite advancements in security technologies, human factors remain a primary contributor to cybersecurity incidents. Studies consistently indicate that errors, lack of awareness, insufficient training, and poor adherence to security policies by individuals directly involved in software development and deployment play a significant role in introducing and perpetuating vulnerabilities [17], [18]. This underscores the importance of understanding the human element in preventing issues like artifact exposure.

Research on developer and operations awareness often reveals disparities in understanding and prioritizing security. While some developers may possess strong secure coding knowledge, they might overlook deployment-specific risks or the implications of certain configurations. The rise of DevOps has introduced methodologies aimed at accelerating software delivery through increased collaboration and automation. However, this acceleration can inadvertently bypass security checks if not explicitly integrated [19], [20].

This has led to the emergence of Dev Sec Ops, a paradigm that advocates “shifting security left”-integrating security considerations throughout the entire software development lifecycle, from design and coding to testing and deployment. Effective DevSecOps relies on automated security testing, continuous monitoring, and fostering a culture where security is a shared responsibility rather than an afterthought [21]. Training and educational initiatives are pivotal in enhancing the security posture of development and operations teams, aiming to in still a proactive security mindset and mitigate human-induced vulnerabilities.

E. Gaps in Current Research

Existing literature offers a robust technical understanding of web application vulnerabilities, including the mechanics of directory brute-forcing and the types of sensitive artifacts that can be exposed. There is also a growing body of work on Dev Sec Ops principles and the importance of human factors in cybersecurity [22]-[25].

However, a significant gap exists in qualitative research that delves deeply into the perceptions, awareness levels, and practical mitigation strategies employed by the specific individuals directly involved in web development and deployment-namely, Dev Ops engineers, system administrators, and web developers-regarding the specific risks of directory brute-forcing and the exposure of development artifacts.

While some studies touch on general security awareness, few provide in-depth, firsthand accounts of the challenges, blind spots, and decision-making processes faced by practitioners in their day-to-day operations that contribute to these vulnerabilities.

This qualitative study aims to bridge this gap by offering rich, nuanced insights into the human and practical dimensions of preventing hidden resource exposure, thereby complementing existing technical and theoretical literature.

III. METHODOLOGY

A. Research Design

This study adopted a qualitative research design with an exploratory orientation. The nature of the research question-focusing on human awareness, behavioral patterns, and operational practices-demanded a design that could provide deep, interpretive insight rather than surface-level quantification. Qualitative exploration is particularly well-suited for uncovering not just what is being done, but how and why it is done, especially in contexts where existing literature is sparse or fragmented, aligning with Braun and Clarke's framework [26].

The decision to employ this approach was influenced by the complexity of web vulnerability management, especially where technical knowledge intersects with organizational culture, deployment workflows, and personal responsibility. Rather than surveying hundreds of professionals for statistically measurable trends, the intent was to zoom in on a smaller, more focused sample of practitioners and understand their thinking, assumptions, and practices in real-world settings.

This design allowed the researcher to gather rich descriptions of how developers, DevOps engineers, and system administrators perceive risks related to directory brute-forcing and the exposure of sensitive artifacts-such as .git directories, shell history files, and environment configurations. Through natural conversations and semi-structured dialogue, it became possible to reveal gaps between assumed security practices and actual behavior, as well as the rationale behind certain decisions (or omissions) that might leave systems vulnerable.

B. Sampling Strategy

The sampling approach for this study was deliberately purposive, as defined by Jacques and Wright [27], driven by the need to engage participants with real, hands-on experience in deploying, managing, or securing web applications. The goal was not to generalize findings to a wide population but rather to engage with individuals who could offer meaningful, experience-based insights into the research questions-particularly around artifact exposure and directory brute-forcing.

A total of eleven participants were recruited, all of whom were professionals based in Nigeria and actively engaged in various areas of software development, systems administration, and Dev Ops. Some worked in formal institutions such as universities and corporate organizations, while others operated in freelance or start-up environments. This diversity of backgrounds added a useful layer of contrast to the data, highlighting differences in tooling, security culture, resource availability, and awareness levels.

Recruitment was done informally, leveraging personal and professional networks, LinkedIn, and developer community forums. Given the niche focus of the topic, participants were approached based on their visible engagement with web technologies or security-related discussions. A brief screening was conducted to ensure they had at least one year of experience working with live web deployments and were familiar with server-side configurations and CI/CD processes.

While eleven may appear numerically small, it was sufficient for this qualitative inquiry. The sample size was adequate to reach thematic saturation, where recurring ideas and concerns began to emerge across interviews-allowing the researcher to identify not just individual stories but also patterns of thought and practice across the group. Each participant brought a unique perspective, yet several shared overlapping experiences, especially regarding overlooked security gaps and why those gaps persist despite growing awareness of threats.

C. Data Collection

Data for this study were collected through semi-structured interviews, a method chosen for its flexibility and depth. This approach allowed for guided yet open-ended conversations, where participants could speak freely about their experiences, while the researcher could probe deeper when interesting or unexpected insights emerged. The semi-structured format also ensured that core topics-such as awareness of directory brute-forcing, practices around artifact management, and the use of mitigation tools-were consistently addressed across interviews.

Interviews were conducted over a span of two weeks, using virtual platforms such as Google Meet, Telegram Voice, and WhatsApp Calls, depending on participants' preferences and internet accessibility. This virtual mode of data collection was practical given geographical spread and time constraints, and it aligned well with the participants' tech-savvy nature and digital workflows. Each interview lasted between 30 and 45 minutes and was conducted in English.

Prior to each interview, participants were given a brief overview of the study's aims and reassured about confidentiality. With verbal consent, interviews were audio-recorded to ensure accuracy in later transcription and analysis. During the interviews, questions moved from general topics-such as participants' roles and experiences in deployment-to specific questions about web-based vulnerabilities, directory brute-forcing, artifact exposure, and how (or whether) these issues were addressed in their environments.

Although interviews followed a guide, the researcher intentionally allowed space for participants to explore areas they considered important. In several instances, participants volunteered stories of incidents they had witnessed or

handled, including security oversights that led to near-breaches or internal red flags. These accounts added authenticity and narrative depth, revealing not only technical but also emotional and ethical dimensions that practitioners navigate.

D. Ethical Considerations

While this study did not pass through a formal university ethics board, every effort was made to ensure it met acceptable standards of research integrity and ethical responsibility. Given the sensitivity of the topic-touching on potential security lapses and personal or organizational practices-it was crucial to approach each participant with clarity, discretion, and respect.

Participants were fully informed, prior to the start of each interview, about the purpose of the study, the kinds of questions that would be asked, and the intended use of their responses. It was emphasized that the study was academic in nature and not an audit or security assessment. They were assured that no part of their responses would be linked to their names, organizations, or specific projects in any published form. To protect identities, all personal identifiers were removed during transcription, and participants were assigned generic labels such as "Participant A," "Participant B," and so on.

Voluntary participation was a cornerstone of the process. Each individual was asked to give verbal consent before recording began and was reminded that they could skip any question or withdraw from the interview at any point-without any need to explain. Fortunately, all eleven participants completed their interviews without withdrawal.

Additionally, care was taken to avoid questions that might place participants in legally or professionally compromising situations. When discussions touched on sensitive details-such as server misconfigurations, data exposure, or inadequate practices-the researcher steered the conversation toward generalized reflection rather than specifics. The intention was never to expose flaws but to understand broader patterns, knowledge gaps, and practical constraints shaping behavior in real-world technical environments.

This ethical grounding enabled participants to speak candidly, knowing their insights were valued not as vulnerabilities to be judged but as experiences to be learned from. The confidentiality measures also supported academic rigor and personal trust-a balance essential when researching topics at the intersection of technology, accountability, and risk.

E. Data Analysis

The data analysis process followed a thematic analysis approach, which is commonly used in qualitative research to identify, interpret, and report patterns within textual data. After completing all eleven interviews, each audio recording was transcribed verbatim to preserve the richness of participants' expression, tone, and phrasing. Transcripts were then read and reread to ensure familiarity with the content before formal coding began. Initial coding was carried out manually using a hybrid approach: inductive codes emerged directly from the data, while deductive codes were informed by the research questions and existing literature on web vulnerabilities. For instance, inductive themes such as "false sense of security," "tool fatigue," and "legacy artifact neglect" surfaced naturally, whereas deductive themes like "awareness levels," "mitigation practices," and "tool usage patterns" ensured alignment with the study's focus.

Once the preliminary codes were developed, they were grouped into broader themes that captured recurring ideas across participants. For example, the theme "Inconsistent Mitigation Strategies" encompassed responses highlighting how different teams or individuals applied patches, updated configurations, or managed sensitive directories based on convenience rather than formal policies. Similarly, the theme "Tooling Gaps and Over-reliance" reflected patterns where participants either misused popular tools, failed to configure them fully, or assumed that security was entirely handled by automation pipelines.

Themes were then compared across participants' roles and organizational contexts (e.g., DevOps engineers vs. front-end developers; start-up teams vs. larger institutions). This comparative lens helped reveal how factors like team size, workload, or organizational support shaped whether vulnerabilities were addressed or overlooked.

Throughout the analysis, care was taken not to force data into predefined narratives. Where contradictions or anomalies appeared-such as a participant expressing high awareness of security risks but admitting to minimal practice-they were preserved and treated as meaningful signals rather than errors. These contradictions often revealed tensions between theory and practice, and between intention and execution, providing some of the study's most valuable insights.

IV. FINDINGS OF THE STUDY

A. Inconsistent Mitigation Practices

One of the most prominent themes was the inconsistency in how security mitigation strategies were applied across teams and environments. While participants generally agreed on the importance of securing production systems, their approaches differed widely-often influenced by time constraints, lack of formal policy, or reliance on ad hoc routines.

For instance, several participants admitted to relying heavily on their frameworks or Dev Ops pipelines to "handle most of it," while others practiced manual cleanup and validation. However, when asked whether they validated .git folders or

shell artifacts post-deployment, only 3 out of 11 reported doing this consistently.

"Honestly, it depends on the day. If we're rushing a release, security checks are sometimes skipped. Not proud of it, but it happens."- Participant C (Dev Ops Engineer, Fin tech)

Another participant from a smaller start up noted:

"We use Docker a lot, and I thought the containers isolated things enough. But during testing, we found an old .bash_history that somehow got bundled in a volume. It was embarrassing."
- Participant G (Backend Developer, Start-up)

TABLE I OBSERVED VARIANTS IN MITIGATION BEHAVIOR

Mitigation Approach	Number of Participants	Notes
Consistent and documented	2	Mainly from regulated sectors (e.g., finance)
Ad hoc/manual	5	Often based on personal discipline rather than policy
Automated via pipeline, but unchecked	3	Belief in "done by CI/CD" without audit
No dedicated strategy	1	Admitted full reliance on default server settings

These responses illustrate a crucial disconnect: although participants were technically aware of the dangers, many lacked structured, repeatable procedures to mitigate them. This inconsistency introduces opportunities for exploitation-particularly by attackers using automated tools to brute-force or enumerate hidden paths. Participants also expressed concern that their practices might not scale well or remain secure as system complexity grows.

B. Awareness of Artifact Exposure

A second major theme centered on participants' level of awareness regarding the exposure of sensitive artifacts-particularly version control directories (e.g., .git, .svn), shell history files (e.g., .bash_history, .zsh_history), and environment configuration files (e.g., .env, .profile). While most participants acknowledged the theoretical risk of leaving such files accessible on public-facing servers, actual awareness of their presence in production environments varied significantly. Several respondents expressed surprise when examples were mentioned, particularly regarding. git folders being indexed by search engines or accidentally bundled in deployments:

"Wait, git folders can be accessed from the browser if not restricted? I thought the server would just ignore that." - Participant D (Frontend Developer, mid-size company)

Only 4 out of 11 participants reported engaging in proactive behaviors, such as scanning their deployments for lingering development files or configuring .htaccess or Nginx rules to explicitly block access to these resources.
One DevOps engineer reflected:

"It's easy to forget about things like .env or .bashrc. They're just there on your local, but in shared hosting or Docker images, they creep in. We learned the hard way when someone pulled secrets from an old .env file once." -Participant H (DevOps Engineer, SaaS company)

Despite the clear security implications, the level of formal training or on boarding content addressing this issue appeared minimal.

TABLE II SUMMARY OF AWARENESS LEVELS

Artifact Type	Participants Aware of Risk	Participants Who Scan or Prevent
.git directories	9/11	4/11
.bash_history	6/11	3/11
.env, .profile	7/11	3/11
Shell aliases/config	4/11	1/11

This table illustrates a troubling gap between theoretical awareness and applied preventive action. Some participants assumed their hosting provider or CI/CD tool "took care of that," indicating an underlying overconfidence in default configurations. Overall, the data suggest that although professionals are aware of these risks, they do not routinely audit their systems to address them-particularly in fast-paced environments.

Fig. 1 Word Cloud Highlighting Key Terms and Themes from Participant Responses on Artifact Exposure and Security Awareness

C. Over-Reliance on Tools and Automation

Another recurring theme was the over-reliance on automated tools, CI/CD pipelines, and frameworks for security enforcement-often without proper validation or manual review. While automation is essential for scalability and efficiency, many participants revealed that their teams rarely audited the outputs or configurations of these tools, assuming that security tasks were being fully handled in the background. This faith in automation was particularly evident among mid-level developers and teams using modern Dev Ops stacks, such as Docker, GitHub Actions, and cloud-native deployment pipelines. However, few had configured these systems to explicitly detect or block common exposures, such as .git folders or shell artifacts.

"We use GitHub Actions and Docker for everything. I thought the linter or the Dockerfile setup would catch anything dangerous, but it turns out, unless you specifically exclude those files, they go through." - Participant A (Dev Ops Engineer, e-commerce firm) In one notable case, a participant described an instance where an internal build script-assumed to be secure-pushed a zipped directory containing both the application and its hidden. git history to a public subdomain:

"We only realized it when someone posted the link in a bug bounty forum. The automation just zipped and deployed everything in the repo."

-Participant I (Backend Developer, media startup)

The data indicate that while tools can enforce some best practices, they often lack the contextual understanding or human-level scrutiny needed to identify nuanced vulnerabilities. For example, a .git folder might not trigger a security warning unless a specific rule or plugin is configured to detect it.

TABLE III TOOL USAGE AND ASSUMPTIONS TABLE

Automation Tool Used	Assumed Secure by Default	Custom Security Config Applied	Manual Review Practiced
Git Hub Actions	8/11	2/11	3/11
Docker (for packaging)	9/11	4/11	2/11
Web Framework (e.g. Laravel, Django)	6/11	1/11	2/11

From this, it is clear that many participants trusted the default configurations too much, expecting them to cover all aspects of deployment security. The lack of awareness regarding the boundaries of these tools' responsibilities led to blind spots, particularly in handling legacy files and invisible metadata. In essence, automation was treated not as an assistant to security hygiene but as its replacement.

D. Cultural and Communication Gaps Between Roles

Beyond technical issues, a notable theme was the disconnect in communication and security culture among administrators, developers, and DevOps engineers. Participants frequently highlighted that security responsibilities were often unclear or unevenly distributed, leading to gaps in coverage and accountability.

Several participants mentioned that security knowledge tended to be siloed-developers might understand the code risks but were less aware of infrastructure exposures, while system administrators focused on network-level controls and patching, leaving file-level risks overlooked.

"We don't always talk enough between teams. Sometimes, I only hear about a security issue after it's too late. The DevOps folks think we handle the servers, but we don't check for hidden folders in deployments." -Participant F (System Administrator, financial services)

"It's a bit of a blame game sometimes. Developers say admins should lock down directories, admins say developers shouldn't commit secret files. Without a clear owner, these things slip through." - Participant B (Senior Developer, SaaS). This cultural gap was compounded by the lack of formalized training or cross-functional security policies. Although many participants expressed interest in improving awareness and practices, organizational inertia and resource constraints posed challenges.

TABLE IV SUMMARY: COMMUNICATION AND ROLE CLARITY

Issue	Frequency (Participants Mentioning)
Lack of clear ownership for artifact security	8/11
Insufficient cross-team communication	7/11
Desire for more security training & policies	9/11

Participants generally agreed that improved collaboration and clearer role definitions could reduce many of the operational security gaps related to exposed artifacts and brute-force vulnerabilities.

V. DISCUSSION

A. Technical Awareness and Practice Gaps

A key finding was the inconsistency between awareness and actual mitigation practices. While nearly all participants recognized that artifacts like .git directories pose security risks, only a minority actively scanned for or remediated these vulnerabilities. This gap between knowledge and action echoes findings from prior research (e.g., [28]-[30]), which observed that security knowledge does not always translate into consistent practice, particularly in fast-paced development environments.

The presence of hidden files, such as .bash_history and environment configuration files, in production systems further highlights operational oversights. Such files can expose sensitive command histories or credentials, serving as valuable reconnaissance targets for attackers conducting brute-force or lateral movement attacks. The sporadic attention paid to these files suggests a lack of comprehensive deployment hygiene protocols and risk assessments.

B. Over-Reliance on Automation Tools

Participants reported significant reliance on automated tools-CI/CD pipelines, linters, and deployment scripts-to manage security configurations. However, this reliance was often misplaced; many admitted that default configurations failed to exclude dangerous artifacts, and manual auditing was rare. This aligns with existing literature warning that automation, while powerful, cannot replace expert oversight (e.g., [31]-[33]).

This over-trust in tools without sufficient customization or verification can lead to false security assumptions. For example, tools may not flag. git directories unless explicitly configured to do so. Similarly, automated scans might overlook transient or legacy files if scanning rules are not continuously updated. Therefore, security automation should be viewed as a complement to-rather than a substitute for-human expertise and routine audits.

C. Organizational Culture and Communication Barriers

A notable barrier identified in the study was the lack of clear ownership regarding artifact security. Participants described scenarios in which developers, administrators, and Dev Ops engineers operated in silos, often assuming someone else was responsible for securing hidden or sensitive files.

This ambiguity led to gaps in accountability, with no single role consistently taking responsibility for checking or removing exposed artifacts. As a result, critical vulnerabilities often slipped into production unnoticed, despite good intentions and general awareness. This finding supports the growing recognition in the literature that security cannot be effectively maintained in fragmented environments. As noted in [34] and [35], embedding security as a shared responsibility across development, operations, and security teams is essential to reducing oversight. Organizational structures that promote cross-functional collaboration-such as joint deployment checklists or integrated review meetings-have been shown to enhance security readiness and ensure responsibilities are clearly communicated.

Moreover, many participants expressed a desire for more structured security training but cited competing work priorities and a lack of organizational support as barriers. While motivation existed, the absence of role-specific security programs and scheduled learning sessions prevented deeper engagement.

This is consistent with studies (e.g., [36], [37]) that emphasize the importance of continuous, tailored training to improve security literacy and operational outcomes across development teams.

D. Practical Implications for Stakeholders

For developers, the findings highlight the importance of incorporating artifact hygiene into the software development lifecycle, including explicit exclusions in .gitignore and deployment scripts. Developers should be encouraged to routinely audit their repositories for sensitive files and be trained to understand the security implications of legacy artifacts (e.g., [38], [39]).

For DevOps engineers, the study points to the need for rigorously configuring CI/CD pipelines and containerization workflows to detect and prevent unintended artifact deployment. Automated tools should be regularly updated and supplemented with manual inspection, especially in complex build environments.

For system administrators and security teams, the results suggest adopting systematic scanning of production environments for exposed directories and files, coupled with swift remediation protocols. Establishing monitoring and alerting mechanisms around unusual directory access can provide early warnings of brute-force or enumeration attempts.

At the organizational level, fostering a culture of shared security responsibility-supported by clear policies and communication channels-is essential to closing the gaps identified.

E. Limitations and Scope

While this qualitative study provided rich insights, several limitations must be acknowledged. The sample size of 11 participants, while adequate for exploratory research, limits generalizability. Participants were primarily from small to medium enterprises and start-ups, which may differ in security maturity from larger organizations. Geographic and industry diversity was also limited, potentially biasing perspectives.

Additionally, the study relied on self-reported data, which can introduce social desirability bias, as participants might overstate their security awareness or practices. Future research should consider larger, more diverse samples and employ mixed methods (combining qualitative interviews with quantitative vulnerability assessments) to validate and extend these findings.

F. Future Research Directions

Building on this work, further studies could investigate the effectiveness of specific training programs tailored to artifact security or evaluate new automated tools designed to detect and block sensitive artifact exposure. Research into organizational change strategies that enhance cross-team communication and ownership of security responsibilities could also yield valuable insights.

VI. CONCLUSION

This study examined the awareness and mitigation strategies employed by web administrators, developers, and DevOps engineers in addressing the risks associated with directory brute-forcing and the exposure of sensitive artifacts, such as .git folders and shell history files. Through qualitative interviews with 11 professionals, the research uncovered notable inconsistencies in security practices, a heavy reliance on automation tools without sufficient manual oversight, and significant communication and cultural gaps within organizations.

The findings reveal that, despite general awareness of these risks, operationalizing effective mitigation remains a challenge. Artifact exposures persist due to unclear role ownership, incomplete training, and overconfidence in automated processes. These vulnerabilities present a real attack surface that adversaries can exploit, especially when combined with other attack vectors. Addressing these issues requires a comprehensive approach that combines technical controls with organizational change. Clear delineation of security responsibilities, ongoing education tailored to different roles, and integration of manual audits with automated tooling are essential steps toward improving security hygiene.

This research contributes to understanding operational security challenges in modern web environments and highlights a critical gap between what is technically possible and what is routinely secured. It encourages organizations to prioritize artifact security as part of their broader cybersecurity strategy and calls for further research into effective interventions. By bridging the disconnect between awareness and practice, organizations can significantly reduce their vulnerability to brute-force attacks and artifact exposure, thereby enhancing their overall security posture.

VII. RECOMMENDATIONS

This study highlights the urgent need for improved operational security practices among administrators and DevOps teams. Regular configuration audits and the integration of secure defaults in deployment pipelines should be prioritized to prevent the exposure of sensitive directories and artifacts, such as .git folders and .bash_history files.

Organizations are encouraged to adopt lightweight, automated scanning tools within development and staging environments to proactively detect and alert on such exposures. Alongside tooling, targeted training programs can address the evident gaps in awareness and inconsistent mitigation practices identified in this study. Ultimately, fostering a security culture built on the principle of *least exposure*-treating all files and directories as potentially public until proven otherwise-will help teams minimize attack surfaces and improve their overall defensive posture.

Declaration of Conflicting Interests
The authors declare no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The authors received no financial support for the research, authorship, and/or publication of this article.

Use of Artificial Intelligence (AI)-Assisted Technology for Manuscript Preparation
The authors confirm that no AI-assisted technologies were used in the preparation or writing of the manuscript, and no images were altered using AI.

REFERENCES

[1] Bach-Nutman, M. (2020). Understanding the top 10 OWASP vulnerabilities. *arXiv*. https://doi.org/10.48550/arxiv.2012.09960

[2] Ezenwoye, O., & Liu, Y. (2022). Web application weakness ontology based on vulnerability data. *arXiv*. https://doi.org/10.48550/arxiv.2209.08067

[3] Cheah, C. S., & Selvarajah, V. (2021). A review of common web application breaching techniques (SQLI, XSS, CSRF). *Atlantis Highlights in Computer Sciences*. https://doi.org/10.2991/ahis.k.210913.068

[4] Suguna, N. (2014). Hunting pernicious attacks in web applications with XProber. *American Journal of Applied Sciences, 11*(7), 1164-1171. https://doi.org/10.3844/ajassp.2014.1164.1171

[5] Zhang, B., Li, J., Ren, J., & Huang, G. (2021). Efficiency and effectiveness of web application vulnerability detection approaches: A review. *ACM Computing Surveys, 54*. https://doi.org/10.1145/3474553

[6] Singh, N., Gupta, P., Singh, V., & Ranjan, R. (2021). Attacks on vulnerable web applications. In *2021 International Conference on Intelligent Technologies (CONIT)* (pp. 1-5). https://doi.org/10.1109/CONIT51480.2021.9498396

[7] Dommeti, D., & Voola, P. (2023). Identifying and mitigating common web application vulnerabilities. *South Asian Journal of Engineering and Technology*. https://doi.org/10.26524/sajet.2023.13.9

[8] Kalim, A., Jha, C., Singh, D., Tomar, D., & Tomar, D. (2020). A framework for web application vulnerability detection. *International Journal of Engineering and Advanced Technology*. https://doi.org/10.35940/ijeat.c4778.029320

[9] Farras, N., Loderick, J., Saputri, H., & Sari, A. (2024). Exploring penetration testing: A comparative analysis of brute force directory tools in vulnerability analysis phase. In *2024 2nd International Conference on Technology Innovation and Its Applications (ICTIIA)* (pp. 1-6). https://doi.org/10.1109/ICTIIA61827.2024.10761451

[10] Antonelli, D., Cascella, R., Schiano, A., Perrone, G., & Romano, S. P. (2024). 'Dirclustering': A semantic clustering approach to optimize website structure discovery during penetration testing. *Journal of Computer Virology and Hacking Techniques, 20*(4), 565-577. https://doi.org/10.1007/s11416-024-00512-6

[11] Aggarwal, V., Kaur, D., Mittal, S., Prasad, T. J. S., Batra, D., & Garg, A. (2023). A comparative study of directory fuzzing tools. In *2023 International Conference on Circuit Power and Computing Technologies (ICCPCT)* (pp. 1368-1374). https://doi.org/10.1109/ICCPCT58313.2023.10245217

[12] Antonelli, D., Cascella, R., Perrone, G., Romano, S., & Schiano, A. (2021). Leveraging AI to optimize website structure discovery during penetration testing. *arXiv preprint*. https://doi.org/10.1007/s11416-024-00512-6

[13] Castagnaro, A., Conti, M., & Pajola, L. (2024). Offensive AI: Enhancing directory brute-forcing attack with the use of language models. *arXiv*. https://doi.org/10.48550/arxiv.2404.14138

[14] Dietrich, C., Krombholz, K., Borgolte, K., & Fiebig, T. (2018). Investigating system operators' perspective on security misconfigurations. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1272-1289). https://doi.org/10.1145/3243734.3243794

[15] Hasan, M., Rozony, F. Z., Kamruzzaman, M., & Uddin, M. K. S. (2024). Common cybersecurity vulnerabilities: Software bugs, weak passwords, misconfigurations, social engineering. *Deleted Journal, 3*(4), 42-57. https://doi.org/10.62304/jieet.v3i04.193

[16] Basak, S. K., Neil, L., Reaves, B., & Williams, L. (2022). What are the practices for secret management in software artifacts? *SAGE Journals*, 69-76. https://doi.org/10.1109/secdev53368.2022.00026

[17] Akbar, M., Rafi, S., Hyrynsalmi, S., & Khan, A. (2024). Towards people maturity for secure development and operations: A vision. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*. https://doi.org/10.1145/3661167.3661238

[18] Ramaj, X., Sánchez-Gordón, M., Palacios, R., & Gkioulos, V. (2024). Training and security awareness under the lens of practitioners: A DevSecOps perspective towards risk management. In *Lecture Notes in Computer Science*. Springer. https://doi.org/10.1007/978-3-031-61382-1_6

[19] Rajapakse, R., Zahedi, M., Babar, M., & Shen, H. (2021). Challenges and solutions when adopting DevSecOps: A systematic review. *Information and Software Technology, 139*, Article 106700. https://doi.org/10.1016/j.infsof.2021.106700

[20] Naidoo, R., & Möller, N. (2022). Building software applications securely with DevSecOps: A socio-technical perspective. *European Conference on Cyber Warfare and Security*. https://doi.org/10.34190/eccws.21.1.295

[21] Tomas, N., Li, J., & Huang, H. (2019). An empirical study on culture, automation, measurement, and sharing of DevSecOps. In *2019 International Conference on Cyber Security and Protection of Digital Services* (pp. 1-8). https://doi.org/10.1109/CyberSecPODS.2019.8884935

[22] Bararia, A., & Choudhary, V. (2023). Systematic review of common web-application vulnerabilities. *International Journal of Scientific Research in Engineering and Management*. https://doi.org/10.55041/ijsrem17487

[23] Kerr-Smith, T., Tirumala, S., & Andrews, M. (2024). Assessing web application security through vulnerabilities in programming languages and environments. In *CITRENZ 2023 Conference, Auckland* (pp. 27-29). https://doi.org/10.34074/proc.240109

[24] Lombardi, F., & Fanton, A. (2023). From DevOps to DevSecOps is not enough: CyberDevOps-An extreme shifting-left architecture to bring cybersecurity within software security lifecycle pipeline. *Software Quality Journal, 31*, 619-654. https://doi.org/10.1007/s11219-023-09619-3

[25] Fadlalla, F., & Elshoush, H. (2023). Input validation vulnerabilities in web applications: Systematic review, classification, and analysis of the current state-of-the-art. *IEEE Access, 11*, 40128-40161. https://doi.org/10.1109/ACCESS.2023.3266385

[26] Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology, 3*(2), 77-101. https://doi.org/10.1191/1478088706qp063oa

[27] Jacques, S., & Wright, R. (2008). Intimacy with outlaws: The role of relational distance in recruiting, paying, and interviewing underworld research participants. *Journal of Research in Crime and Delinquency, 45*(1), 22-38. https://doi.org/10.1177/0022427807309439

[28] Yasar, H. (2018). Experiment: Sizing exposed credentials in GitHub public repositories for CI/CD. In *2018 IEEE Cybersecurity Development (SecDev)* (p. 143). https://doi.org/10.1109/SecDev.2018.00039

[29] Malatji, M. (2022). Industrial control systems cybersecurity: Back to basic cyber hygiene practices. In *2022 International Conference on Electrical, Computer and Energy Technologies (ICECET)* (pp. 1-7). https://doi.org/10.1109/ICECET55527.2022.9872810

[30] Yaseen, K. A. Y. (2022). Importance of cybersecurity in the higher education sector 2022. *Asian Journal of Computer Science and Technology, 11*(2), 20-24. https://doi.org/10.51983/ajcst-2022.11.2.3448

[31] Chen, Y., Zahedi, F. M., Abbasi, A., & Dobolyi, D. (2020). Trust calibration of automated security IT artifacts: A multi-domain study of phishing-website detection tools. *Information & Management, 58*(1), Article 103394. https://doi.org/10.1016/j.im.2020.103394

[32] Tilbury, J., & Flowerday, S. (2024). Automation bias and complacency in security operation centers. *Computers, 13*(7), Article 165. https://doi.org/10.3390/computers13070165

[33] Islam, M. S., Sajjad, M., Hasan, M. M., & Mazumder, M. S. I. (2023). Phishing attack detecting system using DNS and IP filtering. *Asian Journal of Computer Science and Technology, 12*(1), 16-20. https://doi.org/10.51983/ajcst-2023.12.1.3552

[34] Khan, M. S., Khan, A. W., Khan, F., Khan, M. A., & Whangbo, T. K. (2022). Critical challenges to adopt DevOps culture in software organizations: A systematic review. *IEEE Access, 10*, 14339-14349. https://doi.org/10.1109/access.2022.3145970

[35] Khattak, K., Qayyum, F., Naqvi, S. S. A., Mehmood, A., & Kim, J. (2023). A systematic framework for addressing critical challenges in adopting DevOps culture in software development: A PLS-SEM perspective. *IEEE Access, 11*, 120137-120156. https://doi.org/10.1109/access.2023.3325325

[36] Ghobadi, S., & Mathiassen, L. (2014). Perceived barriers to effective knowledge sharing in agile software teams. *Information Systems Journal, 26*(2), 95-125. https://doi.org/10.1111/isj.12053

[37] Blaise, O. O., Aaron, I., Alfred, U., & Amusa, A. (2024). Evaluating the ethical frameworks of information security professionals: A comparative analysis. *Asian Journal of Computer Science and Technology, 13*(2), 61-66. https://doi.org/10.70112/ajcst-2024.13.2.4289

[38] Ravichandran, S., & Rao, K. L. N. (2022). Design and development of an advancing web information stockpiling for engraved ontology in user contours. *Asian Journal of Computer Science and Technology, 11*(2), 11-15. https://doi.org/10.51983/ajcst-2022.11.2.3379

[39] Auwal, A. M., & Lazarus, S. (2024). Sociological and criminological research of victimization issues: Preliminary stage and new sphere of cybercrime categorization. *Journal of Digital Technology & Law, 2*(4), 915-942. https://doi.org/10.21202/jdtl.2024.44