systems

MDPI

Article

Dependable and Non-Dependable Multi-Authentication Access Constraints to Regulate Third-Party Libraries and Plug-Ins across Platforms

Santosh Kumar Henge [1], Gnaniyan Uma Maheswari [2], Rajakumar Ramalingam [3], Sultan S. Alshamrani [4], Mamoon Rashid [5,*] and Jayalakshmi Murugan [6]

[1] Department of Computer Applications, Directorate of Online Education, Manipal University Jaipur, Jaipur 303007, India; hingesanthosh@gmail.com or santosh.henge@jaipur.manipal.edu
[2] Department of Computer Science and Engineering, RMK College of Engineering and Technology, Chennai 601206, India; umamaheswaricse@rmkcet.ac.in
[3] School of Computer Science and Engineering, Vellore Institute of Technology, Chennai 600127, India; ramukshare@gmail.com
[4] Department of Information Technology, College of Computers and Information Technology, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia; susamash@tu.edu.sa
[5] Department of Computer Engineering, Faculty of Science and Technology, Vishwakarma University, Pune 411048, India
[6] Department of Computer Science and Engineering, School of Computing, Kalasalingam Academy of Research and Education, Krishnankoil 626128, India; jayalacsmi@gmail.com
* Correspondence: mamoon.rashid@vupune.ac.in

Abstract: This article discusses the importance of cross-platform UX/UI designs and frameworks and their effectiveness in building web applications and websites. Third-party libraries (TPL) and plug-ins are also emphasized, as they can help developers quickly build and compose applications. However, using these libraries can also pose security risks, as a vulnerability in any library can compromise an entire server and customer data. The paper proposes using multi-authentication with specific parameters to analyze third-party applications and libraries used in cross-platform development. Based on multi-authentication, the proposed model will make setting up web desensitization methods and access control parameters easier. The study also uses various end-user and client-based decision-making indicators, supporting factors, and data metrics to help make accurate decisions about avoiding and blocking unwanted libraries and plug-ins. The research is based on experimentation with five web environments using specific parameters, affecting factors, and supporting data matrices.

Keywords: cloud computing; UX/UI; cross-platform; third-party libraries; decision making; rule-based security

Citation: Henge, S.K.; Maheswari, G.U.; Ramalingam, R.; Alshamrani, S.S.; Rashid, M.; Murugan, J. Dependable and Non-Dependable Multi-Authentication Access Constraints to Regulate Third-Party Libraries and Plug-Ins across Platforms. *Systems* **2023**, *11*, 262. https://doi.org/10.3390/systems11050262

Academic Editor: Fernando De la Prieta Pintado

Received: 20 April 2023
Revised: 19 May 2023
Accepted: 19 May 2023
Published: 21 May 2023

1. Introduction

Cloud computing (CC) has become increasingly popular among enterprises and technical communities because it provides on-demand access to configurable distributed resources and CC services (CC-S). CC aims to provide quick data access, security, rapid data control, efficient data-loading storage, and network-based computational CC-Ss as measurable on-demand services over the Internet [1]. CC improves scalability, interoperability, facility, alertness, availability, and flexibility, and can adapt to changes in user–company–user needs and speed up the framework for growth [2,3]. It provides cost-efficient and optimized computing resources by utilizing distributed network servers, networking devices, data-loading clusters, enterprise-level applications (apps), and supporting facilities [4–7].

However, one of the most significant barriers to its acceptance is uncertainties about security and privacy (SecPri) [8]. There needs to be more clarity about how security can be achieved at all levels of data transmission and how application-based security has moved

to the CC [9]. The lack of transparency has made it hard for data administration to prioritize data SecPri in CC [10]. The data security problem is related to the internal–external level of data storage, the dependence on shared-domain-based cyberspace, the lack of data access control, enterprise-level multi-tenancy, and the integration of external-to-internal and internal-to-external levels of security. In healthcare, CC-based specialist structures use disease-based medical data to analyze and find diseases [11]. In the design discipline, UX (user experience) and UI (user interface) are two related but separate phrases, especially in the context of digital products such as websites, apps, and software interfaces. UX/UI can also improve CC-Ss' quality in medicine, science, and engineering, integrating the multi-domain CC-Ss among various levels of customers, enterprises, end-users, and CC-S providers (CC-SP) [12].

The CC offers distinct advantages compared to traditional IT models, such as a highly distributed infrastructure and virtualized resources. However, security measures such as authorization and authentication must still be employed in CC environments [13]. CC security controls are similar to those in other IT environments, but the technology that facilitates on-demand services presents different risks to organizations. Unfortunately, implementing data security in CC solutions often makes them more complex [14]. Companies face significant concerns when moving critical applications and sensitive data to public cloud environments, where they may not have direct control over the underlying infrastructure.

To address concerns regarding security and privacy in cloud computing, a resolution supplier must make specific promises to their clients about maintaining data security and confidentiality controls for their enterprise-level software applications and supportive services. Additionally, they must confirm to clients that their association is protected and that an assessor can verify their CC-S-level conformity. Encryption and decryption techniques have been introduced, proposed, and implemented in CC to secure and manage data access between CC servers, end-users, and clients. It ensures that data are transferred reliably between the client, CC, and end-user storage providers. However, privacy concerns arise when CC providers decrypt cypher-text-based data to handle it for further stages. The data control mechanism is between organizations, data owners, CC-SPs, and Internet service providers, and the type of CC-Ss used [15], as illustrated in Figure 1.

1.1. Motivation

In software development, it is common to use external third-party libraries (TPLs) and plug-ins (PIs) to save time and effort. However, these TPLs and PIs can pose serious security threats to local systems and networks if they contain vulnerabilities, malware, or harmful injectors. TPLs and PIs can come from two sources: unintentional bugs or harmful injectors introduced by code developers or intentional bugs or harmful injectors created by malicious third-party individuals. Unwanted TPLs and PIs can cause significant damage to network resources and services, resulting in data loss and access control issues. One example of a TPL vulnerability is the Heartbleed-OpenSSL vulnerability, discovered in 2014. OpenSSL is an open-source library widely used to implement TLS-SSL functionality in web servers, local servers, hardware appliances, and operating systems. Even famous and well-secured websites such as Yahoo.com were vulnerable to this bug, which required an additional patch to fix.

In 2018, the Event Stream JavaScript (JS) library was subjected to a hack where a third-party dependency was added, which led to maliciously encrypted code injection. This vulnerability can compromise the security of data and systems. In specific scenarios, the injected code can attempt to steal bitcoins from users' wallets if combined with a particular crypto-currency-related library [16]. The resolution of these vulnerabilities can significantly impact the technical performance of designs. To prevent such issues, libraries and patches can be evaluated by incorporating measurements to ensure their security and trustworthiness.

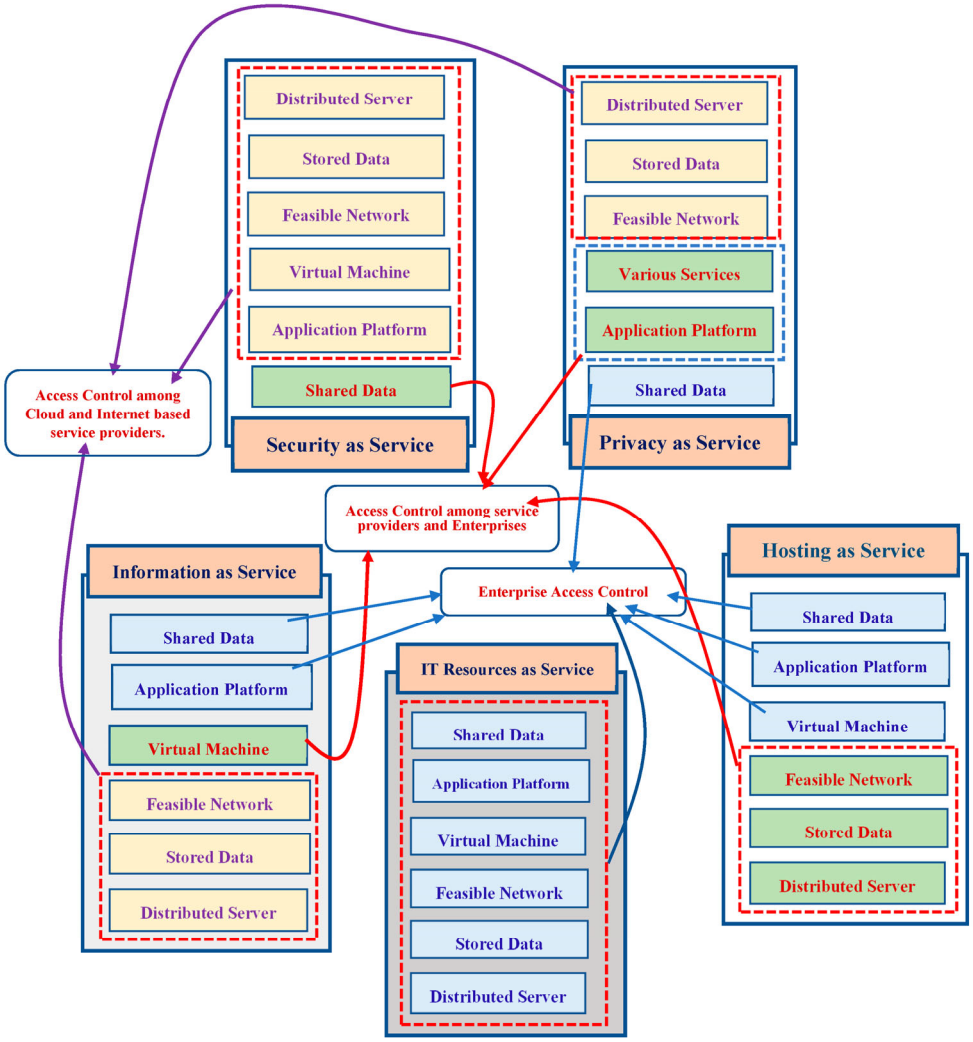

Figure 1. CC-SP and Internet service providers with the type of CC-Ss.

In Figure 1, the red colored arrow lines represent the various fundamental property to frame the access control among service providers and enterprises through the services such as shared data in the form of security as service, application platform, and supporting services in the form of privacy as service; and feasibility network, stored data, and distributed server in the form of hosting as service. The blue colored arrow lines represent the various fundamental properties that frame the enterprise access control through the services such as shared data in the form of privacy as service; virtual machine, application platform, and shared data in the form of hosting as service; application platform and shared data in the form of hosting as service; and application platform, shared data, virtual machine, stored data, and feasible network in the form of IT resource as service. The violet colored arrow lines represent the various fundamental properties that frame the access control among cloud and Internet-based service providers through the services such as stored data, feasible network, and distributed servers in the form of the information as service, application platform, virtual machine, stored data, viable network, and distributed server in the form of the security as service; and stored data, feasible network and distributed server in the form of the privacy as service. Library updates can be challenging since other projects and applications rely on them. Even minor modifications to the library can cause issues with other projects' codes.

Additionally, vulnerable libraries can depend on codes from other sources and libraries, which can be more complex and impossible to fix due to inheritance limitations. Indirect dependencies are not approved licensing models; they function as third-party dependencies'

licensing models. These technical and operational codes impact and restrict the supported licensing models.

1.2. Research Contributions

The significant contribution of this research work is listed below.

- To enhance the security of web applications by integrating dependable and non-dependable (D-ND) multi-authentication access constraints (MAAC) and specific parameters for analyzing third-party applications and libraries in CPF's development.
- Adding D-ND-MAAC-specific parameters enables web desensitization methods to be chained together for easy deployment, ensuring that libraries and local–global environments can work safely by including several dependent libraries.
- The research also looked at different end-user and client-based decision-making indicators with supporting factors and data metrics, enabling accurate decisions about how to avoid or block unwanted libraries and plug-ins.
- This proposed approach enhances the security of web applications by providing a more comprehensive and practical approach to analyzing third-party applications and libraries, ensuring that web applications remain secure in the face of evolving security threats.

This research article is organized as Section 2, which presents the related work, including the context and methodology flow of existing approaches, their advantages, and limitations in associated fields. In Section 3, the proposed methodology of the Rules-based Secure Injection Access Constraints Approach (D-ND-MAAC) is introduced, including the representation of the state level of injection of suspicious third-party libraries (TPLs), decision-making indicators (PIs), and implicated cross-platform (CPFs) development types in D-ND-MAAC through different CPFs such as Web-based apps (WbA), Hybrid-based apps (HbA), Interpreted-based apps (IbA), and Generated-based apps (GbA). The section also includes D-ND-MAAC-based cross-platform mobile development (CPMD) security analysis systems (SAS) and third-party libraries the and secure integration of third-party libraries. Section 4 provides the experimental results and discussion, while Section 5 concludes the study and presents future directions.

2. Related Work

This section articulates the various existing methodologies and models with their methodology flow, statistical aspects, theoretical–technical feasibilities, advantages, and lagging issues.

Homomorphic encryption (HE) is a cryptographic technique that enables the processing, storing, and transferring of data in CC environments. Fully homomorphic encryption (FHE), as described in [17], allows subjective computation on ciphertext-based data without decryption. Current HE methods have limited homomorphic functions such as addition and multiplication, which restrict their use in various real-world CC applications requiring basic homomorphic procedures for data security and privacy [18]. However, the processing speed and power necessary for these methods can adversely affect the end-user response time (EU-RT) and level of power consumption (PC). FHE addresses these issues by providing efficient data security and privacy among end-users and CC servers. With the increasing number of mobile platforms (PFs), mobile applications (MAs) are being developed rapidly to meet the growing demand for new content. The use of emerging mobile native app (NA) technologies, such as artificial intelligence, IoT, and geofencing, enables people to be assisted in times of need. NAs help organize mobile apps (MApps), but the main drawback is that the source code (SC) cannot be reprocessed for alternative supportive platforms. Rebuilding remarkably similar apps is necessary for the base code [19]. NAs provide essential software tools for developing and implementing business logic in real-time environments but require more technological tools, experience, expertise, and workforce than other apps.

An NA can provide a superior user experience compared to other types of applications. It is designed to run directly on the device's hardware and data storage, resulting in rapid

performance, a reliable appearance, a clear view, and complete data logging. UX measures the end-users knowledge of how to utilize gadget functionalities. The end-user must be able to control apps instantly following the installation process and anticipate from the app to the utility using a traditional technique. In addition, the user choice-based unique settings, such as the selection of regional language and language translation, are essential features considered in the version through the application-level execution process. Christoph Rieger et al. [20] proposed an analysis that evaluates various frameworks and assesses them for web apps and NA progression. The study concluded that cross-platform improvement had witnessed much development, but the challenges are ever-increasing, and further support for app designers is necessary.

Lennon C. et al. [21] presented novel authentication methods for digital systems vulnerable to insecurities. They employed standard software testing and satisfiability modulo assumptions of DSVerifier v2.0. The presented approach was used to verify the robustness of closed-loop management systems concerning finite word-length effects. On the other hand, Henning Heitkötter et al. [22] proposed expanding the model-driven cross-platform approach MD2, designed explicitly for outlining specific control structures and offline computation. This approach is focused on the business applications field and aims to maintain a precise appearance and texture while incorporating a high level of abstraction. It is integrated with an MVC-based DSL and regularly updated to support iOS and Android platforms.

Pustišek M et al. [23] conducted a study to propose and evaluate three application designs that differ in interaction, computation, storage space, and protection constraints. They also analyzed the data traffic necessary to run blockchain clients and their applications. Meanwhile, Akasiadis C et al. [24] proposed a Multi-Protocol Internet of Things (IoT) Platform Framework based on open-source frameworks. The IoT platform framework integrates multiple application layer message protocols, such as Representational State Transfer (REST) or HTTP, Message Queuing Telemetry Transport (MQTT), Advanced Message Queuing Protocol (AMQP), Constrained Application Protocol (CoAP), and WebSockets. It utilizes open-source frameworks such as RabbitMQ, Ponte, OM2M, and RDF4J.

Palviainen et al. [25] suggested a reference performance for the driver component and end-user programming, which were evaluated in cross-smart space applications. Biørn-Hansen et al. [26] developed eight mobile apps that analyzed the influence of machine hardware and the impact of modifications and activities, focusing on apps created using cross-platform expansion structures and performance measurement of animation. Hang et al. [27] integrated blockchain technology into an IoT platform to ensure data reliability detection. The platform's objective was to provide device vendors with a functional application that requires an immutable log and easy access to devices used in various fields.

Saket Acharya et al. [28] provided a comprehensive review of recent Android security concerns, improvements in security implementation, notable malware detected from 2017 to 2021, and the privacy techniques employed by malware developers, along with the current methods of detecting Android malware. On the other hand, Tahir Alyas et al. [29] proposed using Docker Security Engine (DSE) for container protection, employing container execution and vulnerability management. DSE implements four mechanisms: innovative gathering and supplementary rules to enhance container runtime and enforce container capacity, signifying claims in preparation, procedures, file system scanning, network isolation, and Docker image capabilities. Various vulnerabilities were examined, and their difficulty levels were assigned according to the Common Vulnerabilities and Exposures (CVE) classification system. The results showed that the approach is more secure for inter-container interaction, with zero vulnerabilities and an overhead of 3.45% for containers.

Elham et al. [30] proposed an attribute-based access control (AbAC) approach that utilizes the Hyperledger Fabric blockchain (HLFBC) for access control. Instead of assigning roles to all system users or establishing access control lists, AbAC grants access based on the attributes offered by the objective environment. Andreas Biørn-Hansen et al. [31] proposed an analysis of business perceptions and beliefs on cross-platform mobile envi-

ronmental growth. This analysis highlights the attractiveness, acceptance, and evolving issues of using technological expansion contexts and tools. Neline van Ginkel et al. [32] proposed NODESENTRY, a security design for server-side JS that enables an effortless implementation of web-strengthening methods and access control strategies on exchanges among their environments and supportive libraries along with dependable plug-ins. The strategy administration structure of NODESENTRY encourages secure and reliable access control on interactions between environments and supporting libraries.

P. Dhiman et al. [33] proposed two approaches for enhancing security and privacy using blockchain technology in a multi-tenant cloud environment (EMTCE). The first approach involves implementing a Blockchain Merkle-tree Ethereum approach in EMTCE, which utilizes a cypher-text policy attribute encryption algorithmic sequence through various levels of Merkle trees (MT) such as inner, outer, inner–outer, inner–outer–external, outer–inner, and external–outer–inner. The authors reported a validity and data access control (DAC) rate of 92% for this approach. In the second approach, the authors [34] proposed using a Brakerski–Gentry–Vaikuntanathan (BGV) hybrid HE with multi-factor authentication–authorization modes and a Secure Token Key in EMTCE. This approach was tested with 152 end-users by integrating six multi-tenants, five head tenants, and two enterprise levels.

Furthermore, P. Dhiman [35] proposed a study on the complications associated with cloud security and non-homomorphic and HE practices. The study provides a comprehensive overview of past methods and approaches proposed to enhance cloud security and discusses their pros and cons. Finally, the authors proposed a survey of blockchain-based secure models and advances based on different Cloud Management Tool (CMT) services by integrating HE methods to build a robust security system in a multi-tenant environment [36].

The main objective of developing cross-platform mobile applications is to achieve high performance on various operating systems (OSs), including iOS, Android, Symbian, Bada, BlackBerry, Windows Phone, Linux Maemo MeeGo, webOS, etc. The goal is to ensure compatibility with as many platforms as possible. A comparative analysis of existing CPF development approaches is presented in Table 1. It includes the key constraints, functional environment, and supportive metrics considered in the proposed process for developing cross-platform mobile applications. The goal is to optimize the performance of mobile applications while ensuring compatibility across multiple operating systems.

Table 1. Comparative analysis of existing CPF developments with their supportive environment.

Author and Citation	Proposed Approach	Key Constraints	Functional Environment	Integrated Environment	Proposed Solution
Tebaa M et al., 2012 [17]	HE method applied to CC	Data processing, storing, and transferring stages	data security and privacy	Mobile platforms (PFs) with processing–storing–transit modes	Security and privacy of data at storage, transit, CC server, and user mode
Spyros Xanthopoulos et al., 2013 [19]	Relative Analysis of CPFs Development Approaches for Mobile Applications	End-users' data in CC servers' mode	end-user EU-RT and PC for data security and privacy among the end-users and CC servers	Mobile platforms (PFs)	Security and privacy of data at CC server, storage, transit, and end-user mode
Christoph Rieger et al., 2019 [20]	Evaluation of framework for CPF app development approaches	User choice-based special settings such as the selection of regional language and language translation	Assesses web apps and NA	Web apps' and NA progression	Secure auto-integration of end-user level as well as app-level certain settings
Lennon C. et al., 2019 [21]	Verifying fragility in digital systems with uncertainties using DSVerifier v2.0	Software standard testing and satisfiability modulo assumptions of DSVerifier v2.0	Closed-loop management systems regarding finite word-length effects	Authentication procedures for digital systems	Vulnerability auto-filter and alerting mechanism

Table 1. *Cont.*

Author and Citation	Proposed Approach	Key Constraints	Functional Environment	Integrated Environment	Proposed Solution
Henning Heitkötter et al., 2015 [22]	Extending a model-driven CPF development approach to business apps, the science of computer programming	Model-driven cross-platform approach MD2. It is integrated on an MVC-based DSL and is routinely renovated into NAs for iOS and Android.	Specifically, design-certain outline with expanded control structures, and offline computation	MVC-based DSL. Business apps for iOS and Android	Multi-authentication cross-platform process for personal and commercial applications
Pustišek, M et al., 2019 [23]	Communication Constraints of Ethereum-Based Decentralized Applications	It evaluated three application designs varying in interaction, computation, storage space, and protection constraints	Blockchain clients and their applications	Integrated IoT-based devices	Secure auto-integration of end-user level as well as app-level secure settings
Akasiadis, C et al., 2019 [24]	Multi-Protocol IoT CPFs based on open-source frameworks	Chains multiple application layer message protocols such as Representational State Transfer or HTTP, Message Queuing Telemetry Transport, Advanced Message Queuing Protocol, Constrained Application Protocol, and WebSockets	It was self-possessed with open-source frameworks such as RabbitMQ, Ponte, OM2M, and RDF4J.	Integrated IoT-based devices.	Integrated the web applications and mobile apps to analyze vulnerability through the various supporting plug-in filters and auto-alerting mechanism
Palviainen, M et al., 2012 [25]	Framework for End-User Programming of Cross-Smart Space Applications	Performance of the structure, tools and methods for the driver component enhancement and end-user programming	Cross-smart space applications	Smart space applications and devices	Full, partial, and on-demand integration of web and mobile apps based on multi-authentication and access control
Biørn-Hansen et al., 2019 [26]	Animations in Cross-CPFs Mobile Applications: An Evaluation of Tools, Metrics and Performance	Analysis and evaluation to inform on the machine hardware influence and consequences triggered by modifications and activities	Prominence on Apps created using cross-PF expansion structures, along with the involvement of performance measuring of animation	Eight mobile apps	Multi-authentication cross-platform process for personal and commercial applications.
Hang, L et al., 2019 [27]	Design and Implementation of an Integrated IoT Blockchain CPFs for Sensing Data Integrity	Unchangeable log and permits easy access to their devices utilized in various fields.	PF is to offer the device vendor that requires thorough a functional application	Integrated IoT PF using blockchain technology	Secure auto-integration of data services based on the user's behavioral factors
Saket Acharya et al., 2022 [28]	A Comprehensive Review of Android Security: Threats, Vulnerabilities, Malware Detection, and Analysis	Assessment of recent Android security interests, security execution improvements, substantial malware identified from 2017 to 2021	Secrecy processes utilized by the malware designers	Android-based Apps	Vulnerability auto-filter and alerting mechanism
Tahir Alyas et al., 2022 [29]	Container Performance and Vulnerability Management for Container Security Using Docker Engine	Docker Security Engine (DSE) practices four mechanisms such as innovative gathering	Container runtime by supplementary rules	Preparation, procedures, file system, scanning of vulnerabilities, network isolation	Vulnerability auto-filter and alerting mechanism
Elham A et al., 2022 [30]	Attribute-based access control (AbAC) approach using Hyperledger Fabric blockchain (HLFBC)	Attribute-based access and HLFBC	Implications of various access controls	Role-based user systems	Multi-authentication cross-platform process approach

Table 1. *Cont.*

Author and Citation	Proposed Approach	Key Constraints	Functional Environment	Integrated Environment	Proposed Solution
Biørn-Hansen et al., 2019 [31]	An Empirical Study of cross-platform mobile development in Industry	Highlighted attractiveness, acceptance, and evolving issues associated.	Acceptance of various access controls	The use of technological expansion contexts and tool	Security and privacy of data at CC server, storage, transit, and as well as end-user mode
Neline van Ginkel et al., 2019 [32]	A Server-Side JavaScript Security Architecture for Secure Integration of third-party libraries	Web-strengthening methods and access control strategies on exchanges among their environments	Supportive libraries along with dependable plug-ins.	Data exchange among various multi-tenants	Vulnerability auto-filter and alerting mechanism
Dhiman et al., 2023 [33]	Blockchain Merkle-tree Ethereum approach in enterprise multi-tenant CC environment	It has implicated cypher-text policy attribute encryption algorithmic sequences through various levels of MT such as inner, outer, inner–outer, inner–outer–external, outer–inner, and external–outer–inner	Validity and data access control (DAC)	CE-based multi-tenants	Integrated the web applications and mobile apps to analyze vulnerability through the various supporting plug-in filters and auto-alerting mechanism
Dhiman et al., 2022 [34]	Secure Token Key in an EMT CMT using Brakerski–Gentry–Vaikuntanathan (BGV) hybrid HE	Multi-factor authentication–authorization modes.	Integration of Secure Token and Key	Security and privacy data exchange among various multi-tenants	Vulnerability filter and auto-alerting mechanism
Dhiman et al., 2022 [35]	Qualified scrutiny complications in cloud security along with non-HE and HE Practices	Comparative analysis	Integration of Secure Token, Key, and Salting techniques	security and privacy data exchange among various multi-tenants	security and privacy of data at CC server, storage, transit, and end-user mode
Dhiman et al., 2022 [36]	Analysis of BC Secure Models and Approaches Based on Various Services in Multi-tenant Environment	Multi-tenant environment	Analysis of various existing models with secure parameters	Security and privacy data exchange among various multi-tenants	Integrated the web applications and mobile apps to analyze vulnerability through the different supporting plug-in filters and auto-alerting mechanism

Developers can utilize third-party libraries that provide reusable, pre-analyzed, and pretested software apps, saving them valuable resources such as time, money, human resources, and infrastructure investments. It enables developers to concentrate on the core components and features of the applications essential to clients and end-users. However, most third-party libraries (TPLs) or non-TPLs pose security risks, as a vulnerability in any library can compromise an entire server, resulting in customers' sensitive data being compromised. This research paper proposes that third-party applications and libraries used in cross-platform development be analyzed using multi-authentication with specific parameters. Integrating multi-authentication for certain parameters makes it easier to set up web desensitization methods and access control parameters based on multi-authentication. Including multiple dependent libraries allows libraries and local–global environments to work together securely and efficiently.

This study focuses on the stages and environments involved in developing CPFs (Context-aware Policy Frameworks) and the layers of compatible devices. It explores various third-party libraries that support the building, running, deployment, and data access control in CPF-based environments. The study also evaluates existing CPF mobile app improvement models and methodologies. It considers the integration of on-demand third-party libraries, plug-ins, and patches to improve app performance while accounting for external CPFs.

The enchantment of TPL-based JavaScript source code in web environments and applications necessitates considering five risks such as:

- Execution of random malicious code on client systems;
- Loss of control over system resources on client and end-user systems;
- Loss of control over changes to the client application;
- Disclosure or leakage of sensitive information to unauthorized people;
- Compromise the local network systems, appliances, and applications.

3. Methodology: Dependable and Non-Dependable (D-ND) Multi-Authentication Access Constraints (MAAC) Approach

A vast number of TPLs are available for selection, with varying levels of risk. TPLs may contain severe vulnerabilities or unintentional PIs. Additionally, TPLs can serve specific purposes, have frequent or automatic updates, or remain in static mode. Malicious TPL code has caused security incidents in the past, infiltrating benign projects and web applications.

In Figure 2, the brown arrow lines represent the step-wise process of software application development, implementation, testing and deployment stages. The indigo arrow lines represent various vulnerabilities' entry levels at cloud-based software application installation and upgrade stages. The bi-directional arrow lines represent the correlation among multiple libraries and plug-ins required during software application testing and deployment stages. The red arrow lines represent the execution flow among the third-party libraries with their additional dependable factors and software application development stage.

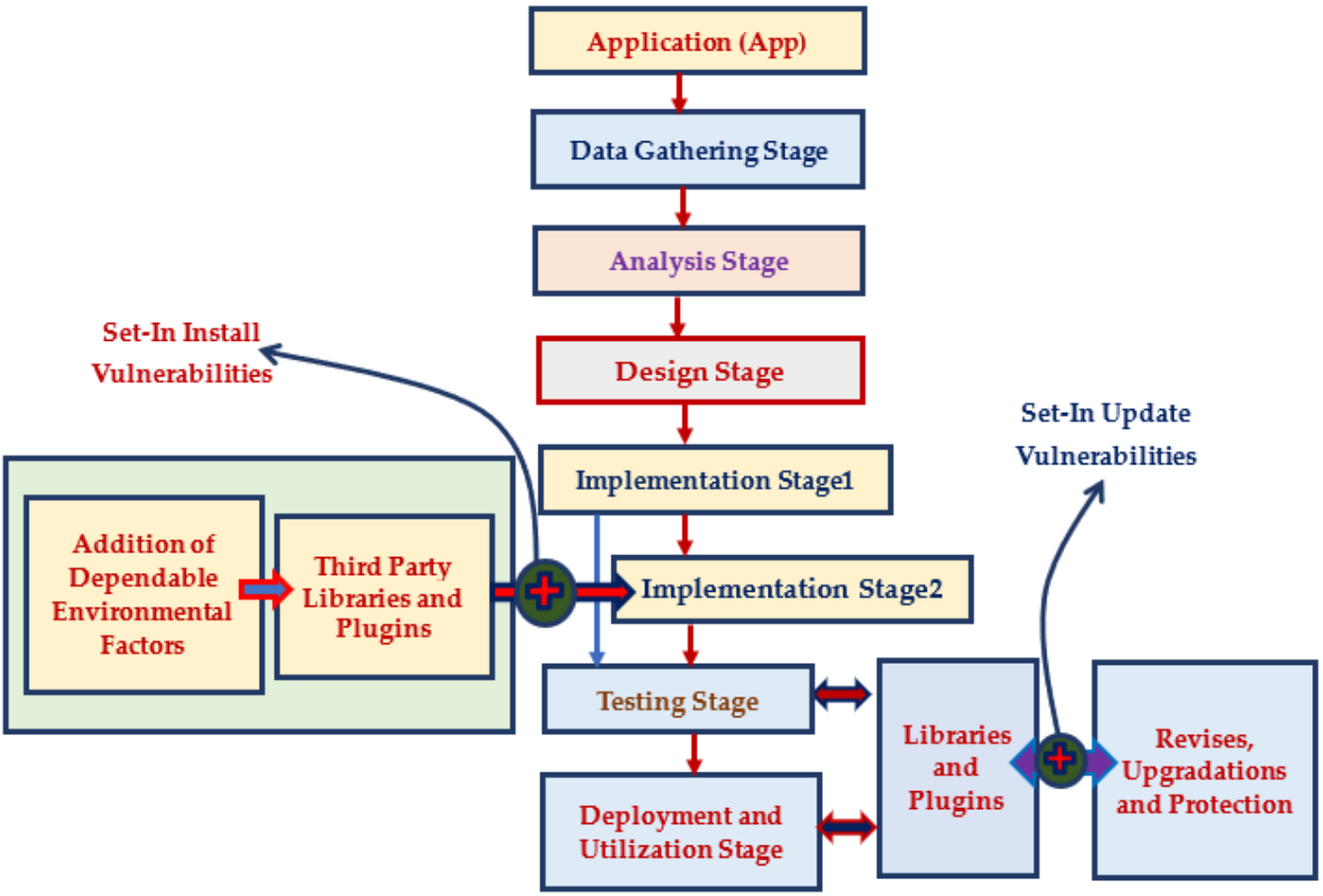

Figure 2. Involvement of vulnerabilities in various stages of application development.

3.1. Analysis of Suspicious TPLs' and PIs' Injection Levels and Their Impact on Cross-Platform (CPF) Development Types in D-ND-MAAC

There is significant growth in innovative software development (SD) environments, applications, and tools designed specifically for intelligent mobile platforms and devices. The rapid expansion of the mobile market has driven the development of CPF SD environments (SDE) that enable faster, more straightforward, more cost-effective, and resourceful software development [16]. These SDEs are primarily responsible for creating four major categories of applications: WbA, HbA, IbA, and GbA.

3.1.1. Cross-Platform (CPFs)—Web-Based Apps (WbA)

The WbA or Web-Browser-Integrated apps are software programs that are transferred and integrated into web environments. These apps are created using a combination of HTML, CSS, and JS, which are extensive cyberspace tools. However, these environments have partial data access to the primary levels of devices, hardware, and networks, and thus require additional time to render web pages and incur additional costs to download internet applications. WbA does not require other installations, and succeeding improvements can be implemented without requiring other facilities. Nowadays, software libraries such as Sencha Touch, JQuery Mobile, and JQTouch ensure the building of web apps that replicate the functionality of native applications.

3.1.2. Cross-Platform (CPFs)—Hybrid-Based Apps (HbA)

HbA is a technology that combines the benefits of WbA and native applications (NAs). It is mainly created by integrating HTML5, CSS, and JS or TypeScript without considering the specific features of the target platforms. HbA employs a thin native container with a WebView in Android and a UIWebView in iOS to embed HTML5-based code apps. The browser is still involved in HbA, which is a part of the ultimate application and can be combined with the application, unlike WbA, where security controls (SC) are taken from websites that are installed on local devices and have access to primary levels of machines, hardware, and networks through specialized dedicated APIs. HbA implements SC by integrating various technologies to develop platform frameworks (PFs). However, to achieve a native look and feel, specific progress libraries such as JQuery and other external services must be used.

3.1.3. Cross-Platform (CPFs)—Interpreted-Based Apps (IbA)

IbA automatically generates native system code to facilitate the execution of UI functionality. End-users and clients can interact with various components of platform-specific native UI. At the same time, the application's integrated logic is independently implemented using different supporting languages and technologies such as Ruby, Java, and XML. This environment is more efficient due to the self-integration of native user interfaces, but its effectiveness relies mainly on the SD environment. Depending on the supportive development environment, the HbA must incorporate new UI features and services of a new Android or other supportive versions while developing new platform-specific features and services.

3.1.4. Cross-Platform (CPFs)—Generated-Based Apps (GbA)

The GbA platform aggregates various services, such as NAs and PFs. Specific applications, such as Applause app development, are generated for each target PF. The GbAs achieve superior performance due to the efficient native code (NC) produced. Furthermore, GbAs can utilize methods to make NC integrate other external services. However, in practical experiments and real-life scenarios, the consumption of the produced NC is not straightforward due to the automated structure of the platform.

3.2. D-ND-MAAC-Based Cross-Platform Mobile Development (CPMD)-Based Security Analysis Systems (SAS) and Third-Party Libraries

The software developers employ the CPF's mobile development (CPMD) resolutions to build up mobile-based applications once, and it is expected to run or deploy on many platforms and devices. The numerous CPF solutions (CPS) are unmoving under consideration of the research process and development process. These CPS build based on different methods and approaches, such as the WbA, Cross-Compilation-based approach, and Virtual-Machine-based approach (VMbA), as shown in Figure 3.

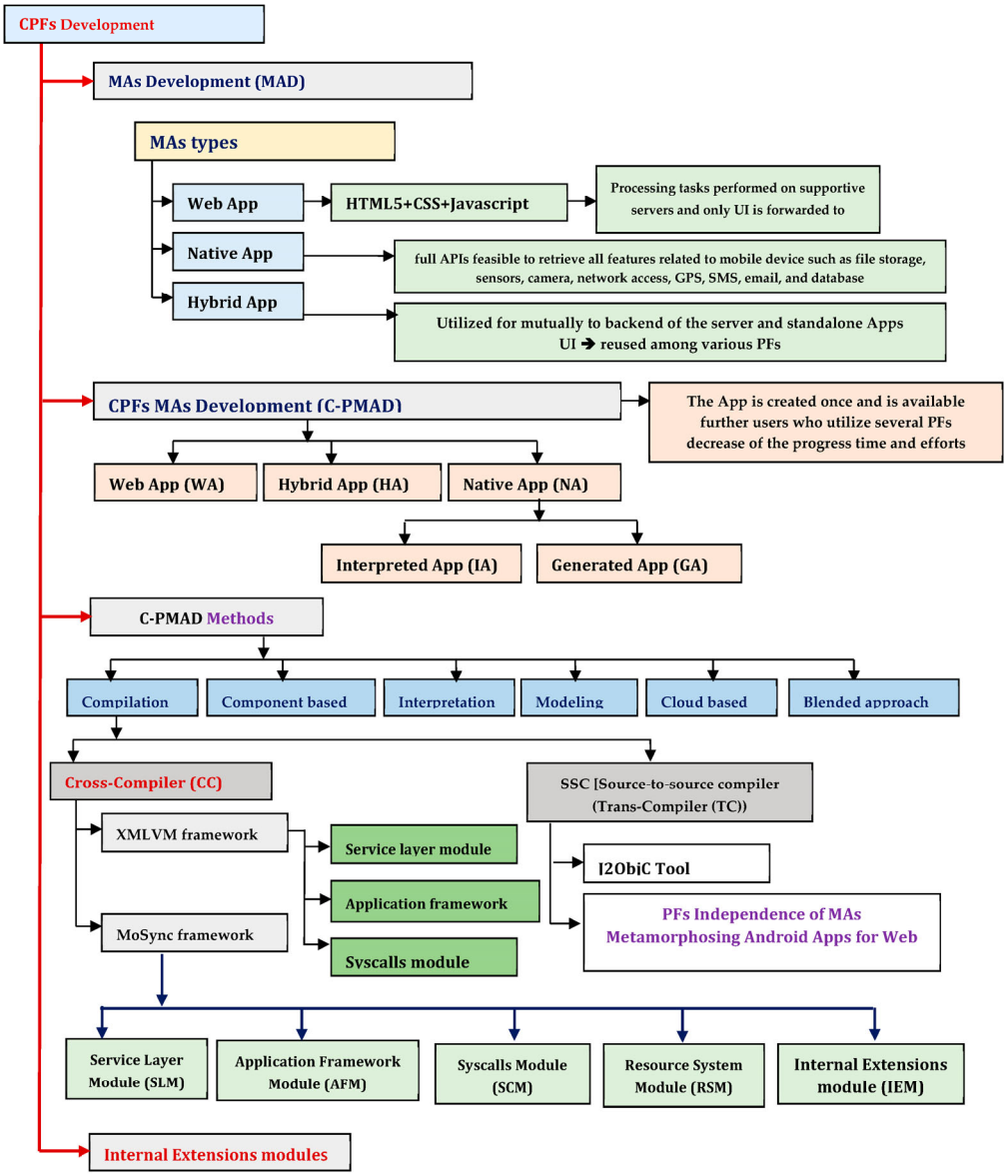

Figure 3. Implications of CPFs' Executional Stages.

In Figure 3, the brown arrow lines represent the CC-SP-based cloud platform development categories such as MAs development (MAD), CPFs MAs development (C-PMAD), C-PMAD methods and internal extensions modules. The black colour arrow lines represent further executions.

3.2.1. Cross-Platform Mobile Development (CPMD)

In literature, several proposed methods, approaches, and models for CPMD solutions have limitations due to traditional methodologies and lack the latest processes, such as the Cross-Compilation-based approach, Component-based approach (CbA), and merged-based

approach (MbA). The increasing usage of smartphones is attributed to the availability of diverse mobile applications (MApps) in various app stores, which are essential for advanced requirements in fields such as healthcare, education, telecommunication, engineering, tourism, medicine, etc. Due to the massive number of smartphone users, there is a growing demand for additional apps. These apps need to cater to the specific requirements of various smartphone platforms such as iOS, Windows Phone, BlackBerry OS, Android, Symbian, etc., which are used by different manufacturers and dealers in the market [37].

Mobile applications are classified based on four distinct forms, which include native, hybrid, devoted web-based apps, and generic mobile-based apps (GMbA). These applications are customized to explicit platforms (PFs) and can be executed on any platform, as shown in Figure 3. Mobile web development (MWD) frameworks provide common characteristics and functionalities to distributed applications by incorporating CPF models that support different platforms. These frameworks use lightweight models with smaller file sizes to accommodate bandwidth restrictions. Additionally, they include custom standards of HTML5 and CSS3 to support touch-screen strategies.

In Figure 4, the brown arrow lines represent pro-step-wise stages. The thin brown arrow lines represent the correlation among the third-party libraries, plug-ins, and the software development stages of implementation, updating, and maintenance. The red colour arrow represents the entry points of third-party vulnerabilities.

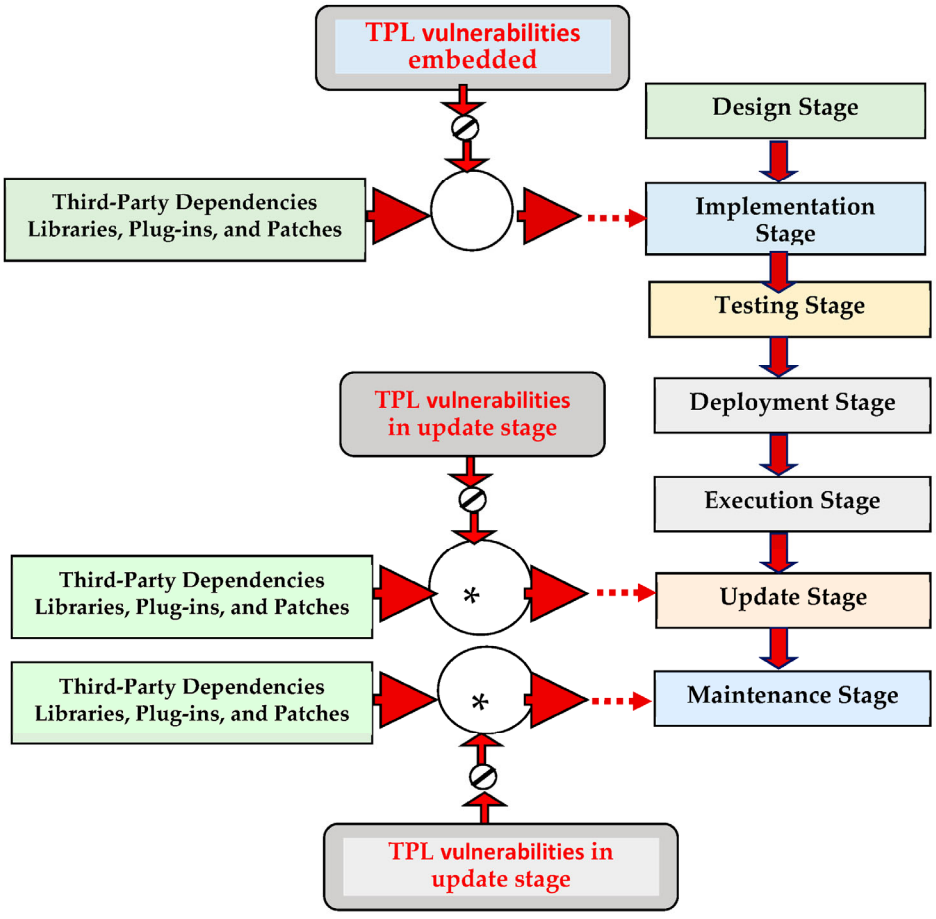

Figure 4. TPLs' and PIs' state-level injection in security analysis systems (SAS).

3.2.2. CPMD-Based Security Analysis Systems (SAS) and Third-Party Libraries

CPMD-SAS is a defensive shield that protects against internal and external attacks to break down data security and privacy. Its main objective is to auto-sense malware and authenticate application behaviour using supporting libraries [38]. CPMD-SAS is deployed at different levels and in various types across supporting devices and distributed servers.

There are two main types of CPMD-SAS methods proposed by researchers, namely static and dynamic analytical approaches [39–43]. Static analytical techniques, presented by various researchers, help in program analysis without executing the code, but they lose some data during the analysis stage. In addition, dynamic analytical approaches, also proposed by researchers, analyze the program code when it is executed, thus overcoming the limitations of the static analytical CPMD-SAS processes [44]. However, they cannot reach statically analyzable control flow. Therefore, hybrid approaches are necessary to analyze data with a defensive shield that works against internal and external attacks [45]. The mixed CPMD-SAS-based analysis presents the best effects by evading the shortcomings of static and dynamic-based CPMD-SAS usage, thus sensing all malware.

3.2.3. Secure Integration of Third-Party Libraries

The TPLs are widely used in MApps and are crucial in the Android and Windows ecosystems. However, TPLs pose both benefits and risks. On the one hand, they provide flexibility in app development, but on the other hand, they bring about security and privacy risks such as data leaks and increased attacking surfaces [46]. To address these issues, researchers have proposed various approaches to characterize TPLs, including auto-detection mechanisms, security integration, privacy filtering analysis of TPLs, and TPL analysis-based attributes supported by factors. However, existing research has not fully addressed all the issues and techniques related to TPLs. Advanced filters are necessary to ensure data security with measurable privacy parameters. These filters will help filter and classify TPLs into dependable and non-dependable libraries, required and non-required functionalities, and demand and non-demand feasibilities.

The use of TPLs has become a standard practice in the development of mobile applications. Software developers employ these libraries to monitor and analyze their apps using third-party plug-ins and advertisements. Additionally, TPLs help to integrate mobile apps with social media sites through various logins on different systems, thereby improving the services provided. JavaScript is one of the most used programming languages in web technologies and has been the primary scripting language for client-side and server-side web scripts for a long time [47]. Integrating server-side JavaScript can give enterprises and companies a wider talent pool, leading to better development outcomes [48]. JavaScript offers many benefits for clients and server systems, including building faster browsers incorporating substances and libraries from different third parties. However, this flexibility comes at a significant cost in terms of security threats (STs) and malware, which can pose a severe risk to the integrity and safety of the system [49,50].

This research integrated a cross-platform mobile development (CPMD) and security analysis systems (SAS)-based hybrid approach to analyze and regulate the data with a defensive shield that works against any internal and external attacks by deploying the weaknesses of TPLs and malware. The hybrid CPMD-SAS-based analysis presents the best effects by evading the shortcomings of static and dynamic-based CPMD-SAS usage, thus sensing all malware.

In the experimental scenario, the apps are synchronized with the rule-based API, which oversees their behavior and regulates the data, authentication, authorization, roles, and access control. New PIs and updates are upgraded based on the regulated rules and threshold values. The experimental scenario is framed through ten regulated execution stages:

Stage 1: Classified the apps into five groups based on their software and hardware functionality parameters, as shown in the Table 2.

Stage 2: All TPLs and PIs are classified based on the behavior of the five clustered app groups, technical and functional regional factors.

Stage 3: Identified the entry and exit points of TPLs and PIs. In this research, the software application deployment, upgrade, and maintenance stages were considered experimental entry points for TPLs and PIs.

Stage 4: Proposed 72 rules, including 10 cluster rules (CR1 to CR10), to manage and control all restricted TPLs and PIs based on the app's functional, technical, regional,

dependency range, and behavioral factors. These rules were developed according to the end-users and customers' requirements, active applications, and regions, as presented in Tables 4 and 5.

Stage 5: In this stage, all cluster rules are synchronized with internal regulations based on correlation sets designed to align with end-user and customer requirements, as well as active applications and regions.

Stage 6: Mapping of five different modes of apps with their range of access control, involvement, and behavior based on TPLs and PIs, which helps to filter the unauthorized TPLs and PIs. The mapping range is classified into two variations, auto-enable mode (AEM) and auto-disable mode (ADM), ranging from 1 to 100 percentage points. AEM represents 50 to 100% points, while ADM represents 0 to 49%. The ADM mode is activated when the mapping percentage is between 1 and 49, described as a cross (X) in Table 5.

Stage 7: All correlation-synchronized set rules are initiated at the installation, updating, and maintenance stages of software applications and mobile apps.

Stage 8: The new and existing TPLs and PIs entries are tuned through the correlation sets of cluster rules.

Stage 9: Newly initiated and existing TPLs and PIs are pipelined through the correlation sets of cluster rules and generate the mapped value to decide whether to allow or disallow.

Stage 10: It is an automated process; the CPMD-SAS correlation sets of cluster rules analyze the new and existing TPLs and PIs based on their functional behaviour, and the mapping value will change accordingly, helping to make operational decisions.

Table 2. A reasonable analysis of CPFs' improvement models, approaches, and methodologies.

Type of App versus Software and Hardware Parameters	User-Perceived Performance	Hardware and Data Access	User Interface and Look and Feel	Marketplace Deployment	Widespread Technologies	APIs through Software and Hardware	Auto-Integration	Software Libraries
Web-based Apps (WbA)	Low	Limited	Simulated	No	Yes	Standard Interface	Partial	JQuery Mobile, Sencha Touch, JQTouch, WebApp.net, Xui, and many others.
Hybrid-based Apps (HbA)	Medium	Limited	Simulated	Yes, but not guaranteed	Yes	Hybrid High-level Interface	Full	Native thin containers such as UIWeb view in Ios and WebView in Android. Containers such as Cordova, Capacitor, PhoneGap
Interpreted-based Apps (IbA)	Medium	Limited	Native	Yes	Yes	High-level Interface Special APIs such as SQLite API applied to store App status and data	Full	Appcelerator Titanium Mobile SDEs for designing inferred Apps.
Generated-based Apps (GbA)	High	Full Access	Native	Yes	No	High-level Interface	Full	Foremost PFs such as Android, iOS, and Windows Phone, which produces SC of Java, Objective-C, C#, Python, etc.
Model-driven software development (MDSD)	High	On-demand Access	Native	Yes	Yes	Problem domain-based development. App functionality and specifications	Full	iPhonical and applause utilize a domain-exact language (based on the XText framework) as input

4. Results and Discussion

The experimental scenarios were developed using Node JS along with integrations of MongoDB. The system architecture utilizes a rule-based environment integrating various third-party libraries and plug-ins. The Node JS architecture is designed to facilitate event-driven programming (EDP) for the development of web servers, making it easy for web developers to create highly scalable and performance-oriented server-based applications. The use of simplified methods of EDP enables the initiation of callbacks and prevents the need for working with concurrency with additional server-side programming languages. The experimental scenarios include integrating various software and hardware parameters to analyze third-party plug-ins using multiple types of apps, as depicted in Table 3.

Table 3. Integrated software- and hardware-based parameters in the proposed methodology.

	User-Perceived Performance	Hardware and Data Access	User Interface and Look and Feel	Marketplace Deployment	Widespread Technologies	APIs through Software and Hardware	Auto-Integration	Software Libraries
Web-based apps (WbA)	Low	Limited	Simulated	No	Yes	Standard Interface	Partial	JQuery Mobile, Sencha Touch, JQTouch, WebApp.net, Xui, and many others
Hybrid-based apps (HbA)	Medium	Limited	Simulated	Yes, but not guaranteed*	Yes	Hybrid High-level Interface	Full	Native thin containers such as UIWeb view in Ios and WebView in Android. The container's such as Cordova, Capacitor, PhoneGap
Interpreted-based apps (IbA)	Medium	Limited	Native	Yes	Yes	High-level Interfaces Special APIs such as SQLite API applied to store app status and data	Full	Appcelerator Titanium Mobile SDEs for designing-inferred apps.
Generated-based apps (GbA)	High	Full Access	Native	Yes	No	High-level Interface	Full	foremost PFs like Android, iOS, and Windows Phone. Which produces SC of Java, Objective-C, C#, Python, etc.
Model-driven software development (MDSD)	High	On-demand Access	Native	Yes	Yes	Problem domain-based development. App functionality and specifications	Full	iPhonical and applause utilizes a domain-exact language (based on the XText framework) as input

4.1. Secure Injection Access Constraints for Third-Party Libraries in Node JS Cross-Platform Development: A Rules-Based Approach to Prevent TPLs' and PIs' State-Level Injection in Security Analysis Systems (SAS)

Node JS is a comprehensive library that supports many functionalities, including System-I/O, networking environment support such as TLS, data bit-streams, and cryptography-based security. Its capabilities make it an ideal tool for handling binary-implicated data. However, the library is susceptible to two typical attack phases involving TPL- and PI-based vulnerabilities. These vulnerabilities can occur during the initial code design or implementation or later during the code updates and maintenance, as shown in Figure 4.

- The best option is to mitigate this risk using a local repository environment (LRE) that will not allow blindly updating dependencies remotely. For this, the LRE uses well-known secure releases that are not corrupted by receiving the code from the source code.
- The rule-based filters and checksums are used to validate the entry point of TPLs and PIs. If there is potential, verify the code reliability along with policy-based checksums if they are accessible with specific library versions, which help to keep the risk at lower levels.

- Do not be worried about minimal use of the newest version of the software libraries and patches out of practicality.
- Update the versions consciously and effectively on time and interpret the release notes carefully before advancing any TPL.
- Pining dependency versions in the source code ensure they are not in an auto-updating mode, which prevents malicious updates and backdoors from sneaking in unnoticed.
- Maintenance of a catalogue of the source code used in creation is the initial step in the defence of it—observance of the source code record on-demandable updating.

This study involved the integration of five distinct modes of mobile applications, including WbA, HbA, IbA, GbA, and MDSD. The proposed methodology was tested using different test cases through various WbA, HbA, IbA, GbA, and MDSD apps. The proposed method integrated over 95 different kinds of TPLs and PIs of multiple apps such as performance booster apps, redundant active and passive apps, alternate apps of Google, cluster workspace apps, similar deviation apps, social media apps, unauthorized entertainment apps and VPN apps, local device-supportive service apps, etc., as shown in Table 4.

Table 4. Integrated application environments, app types, TPL and PI behaviour factors.

Environment Type	Apps Type	Third-Party Libraries (TPLs)	Trigger Level	Behaviour Level	Auto-Integration of TPL Requirement	Harm-Level
Alternatives apps to Google	Bloatware apps	Auto-addable	Reduce the device's usability	It controls other device services	Not Required	High
Inactive applications	Redundant apps	Not Required	Reduce the device's performance	Auto-transfer mode of behaviour	Not Required	High
Cluster workspace apps	Google Workspace apps such as Gmail, Google Keep, Docs, Sheets, Slides, Meet, Calendar, and more.	Not Required	It will compromise the device credentials	Auto-upgradation mode of behaviour	Not Required	High
Performance booster apps	Performance booster apps such as RAM cleaners, battery savers, and game optimizers	Not Required	Reduce the device's hardware and software performance		Not Required	High
Duplicate apps holding the same functionalities.	Data backup apps, web browsers, note carrying apps, and messaging apps	Not Required	Includes critical vulnerabilities that potentially steal personal info, including credit card details, photos, and private chats	Similar nature grants different features	Not Required	High
Social media	Instagram, TikTok, or Snapchat	On-demand TPLs		Auto-popups and holding the wrong options	Not Required	
Unauthorized Personal healthcare apps	Fitness, dating, and meditation	Not Required	Bypass the access control	Auto-popups and holding the wrong options	Not Required	High

Table 4. *Cont.*

Environment Type	Apps Type	Third-Party Libraries (TPLs)	Trigger Level	Behaviour Level	Auto-Integration of TPL Requirement	Harm-Level
Unauthorized VPN apps	SuperVPN Free VPN Client apps	Not Required	includes critical vulnerabilities	Man-in-middle attacks. Exploit the vulnerabilities to take over an end-user's trusted connections to malicious websites that could further endanger user privacy and security.	Not Required	High
Local device-supportive service apps	Super Clean and Master of Cleaner apps It will automatically add unnecessary plug-ins at the time of installation and updates.	Not Required	Super Clean by Magical Dev has registered over 26 million installs on the Play Store.	Auto-spreading apps among the devices. It promises to optimize battery usage, clean junk files, and boost memory, none of which requires a third-party app.	Not Required	High
Entertainment apps	Fildo music app	Not Required	It will automatically add unnecessary plug-ins at the time of installation and updates.	Auto-downloading functionality	Not Required	High

These scenarios were designed to evaluate the performance of the applications in terms of various security and privacy parameters, such as user-perceived performance, hardware and data access, user interface and look and feel, marketplace deployment, widespread technologies, APIs through software and hardware, auto-integration, and the number of software libraries and level of integration. Each mode of the app considered six levels of variations, and the study developed 72 rules to manage and control all restricted TPLs and PIs based on these rules. These rules were designed according to the end-users' and customers' requirements, active applications, and regions. The results of the study are presented in Tables 4 and 5.

- Web-based apps (WbA) ➔{WbA1, WbA2, WbA3, WbA4, WbA5}
- Hybrid-based apps (HbA) ➔{HbA 1, HbA2, HbA3, HbA4, HbA5}
- Interpreted-based apps (IbA) ➔{IbA1, IbA2, IbA3, IbA4, IbA5}
- Generated-based apps (GbA) ➔{GbA1, GbA2, GbA3, GbA4, GbA5}
- Model-driven software development (MDSD) ➔{MDSD1, MDSD2, MDSD3, MDSD4, MDSD5}

In the above statement, the arrow representing the inference relation among the set of dependable parameters such as web-based apps (WbA) correlates with the collection of influenced parameters WbA1, WbA2, WbA3, and WbA4. In the same way, other dependable parameters, HbA, IbA, and GbA, MDSD, correlate with the set of influenced parameters HbA 1, HbA2, HbA3, HbA4; IbA1, IbA2, IbA3, IbA4; GbA1, GbA2, GbA3, GbA4; and MDSD1, MDSD2, MDSD3, MDSD 4, respectively. The modes and levels of WbA, HbA, IbA, GbA, and MDSD are correlated with various supporting mapping parameters such as UPP-P, HDA-P, UI-LF-P, MP-D-P, WST-P, APIs-SWHW-P, AI-P and SL-LI-P. Every supporting mapping parameter has been derived into various matrices with their feasible range of percentage values from 0 to 100%, which have been integrated based on the requirement of customers and end-users, as shown in Table 5.

Table 5. Integration of five different modes of apps with their execution supporting access control factors with their dependable and non-dependable parameters and matrices.

	User-Perceived Performance in Percentage (UPP-P)			Hardware and Data Access in Percentage (HDA-P)		User Interface and Look and Feel in Percentage (UI-LF-P)		Marketplace Deployment in Percentage (MP-D-P)			Widespread Technologies in Percentage (WST-P)	APIs Through Software and Hardware in Percentage (APIs-SWHW-P)					Auto-Integration in Percentage (AI-P)	No. of Software Libraries and Level of Integration (SL-LI-P)	
	Low	Medium	High	Limited	Full	On-Demand	Simulated	Native	No	Yes	Yes, but Not Guaranteed	Yes	Standard Interface	High-Level Interface	Hybrid High-Level Interface	High-level Interface Special APIs	Problem Domain-Based Development App Functionality and Specifications	Partial	Full	SL-LI-P
WbA1	50			50			60		0		80	80	85					70		80
WbA2	60			70			70		0		100	100	100					70		80
WbA3		60		50			80		0		100	80	100					60		80
WbA4		75		50			100		60		100	100			100				100	90
WbA5			80		100		90		100		100	90	100					60		80
WbA6			90		100		100		100		100	100			100				100	80
HbA1	50			50			60			0		90	100					70		100
HbA2	50			50			60			0		100			100			70		100
HbA3		60		50			80				60	95	100						100	100
HbA4		75		40				60		100		100				100			100	100
HbA5			90		100			80		100		97		100					100	100
HbA6			100			100		100		100		100					90		100	100
IbA1	50			50			60		0			70	100					70		70
IbA2	50			50			60		0			100	100						100	70
IbA3		60		50			60				60	82			100			70		70
IbA4		75		40				60		100		100				100			100	70
IbA5			90		100			80		100		94		100				70		70
IbA6			100			100		100		100		100					90		100	70
GbA1	50			50			60		0			70	100					70		100
GbA2	50			50			60		0			100	100						100	100
GbA3		60		50			80				60	82			100				100	100
GbA4		75		40				60		100		100				100			100	100
GbA5			90		100			80		100		94		100					100	100
GbA6			100			100		100		100		100					90		100	100
MDSD1	50			50			60		0			76	100					70		85
MDSD2	50			50			60		0			100	100						100	85
MDSD3		60		50			80				60	89			100				100	85
MDSD4		75		40				60		100		100				100			100	85
MDSD5			90		100			80		100		98		100					100	85
MDSD6			100			90		100		100		100					100		100	85

4.2. Mapping of TPLs' and PIs' State-Level Injections in Security Analysis Systems (SAS)

In SAS, mapping between Threat Protection Levels and PIs has been identified as a critical step for accurate decision making. The practical meaning of the rules is the process of mapping and coordination among the rules, which enables effective filtering and management of all TPLs and PIs, allowing for the control and operation of restrictions based on predefined rules. Our research integrated 72 rules, including 10 cluster rules (CR1 to CR10), designed to align with end-user and customer requirements, as well as active applications and regions. A detailed overview of these rules can be found in Tables 6 and 7. The framed cluster rules are as follows:

- Cluster Rule1 (CR1) = {WbA: HbA:IbA:GbA} ➔({WbA1, WbA2, WbA3, WbA4}) && (UPP-P:{Low, Medium, High}) && (IbA:{IbA1, IbA2, IbA3, IbA4}) ➔(IbA:{IbA1, IbA2, IbA3, IbA4}) ➔(GbA:{GbA1, GbA2, GbA3, GbA4}) ➔(MDSD):{MDSD1, MDSD2, MDSD3, DSD 4})
- Cluster Rule2 (CR2) = {WbA: HbA:IbA:GbA} ➔({HbA 1, HbA2, HbA3, HbA4}) && (UPP-P:{Low, Medium, High}) && (IbA:{IbA1, IbA2, IbA3, IbA4}) ➔(IbA:{IbA1, IbA2, IbA3, IbA4}) ➔(GbA:{GbA1, GbA2, GbA3, GbA4}) ➔(MDSD):{MDSD1, MDSD2, MDSD3, DSD 4})
- Cluster Rule 3 (CR3) = {WbA: HbA:IbA:GbA} ➔({IbA1, IbA2, IbA3, IbA4}) && (UPP-P:{Low, Medium, High}) && (IbA:{IbA1, IbA2, IbA3, IbA4}) ➔(IbA:{IbA1, IbA2, IbA3, IbA4}) ➔(GbA:{GbA1, GbA2, GbA3, GbA4}) ➔(MDSD):{MDSD1, MDSD2, MDSD3, DSD 4})
- Cluster Rule 4 (CR4) = {WbA: HbA:IbA:GbA} ➔({GbA1, GbA2, GbA3, GbA4}) && (UPP-P:{Low, Medium, High}) && (IbA:{IbA1, IbA2, IbA3, IbA4}) ➔(IbA:{IbA1, IbA2, IbA3, IbA4}) ➔(GbA:{GbA1, GbA2, GbA3, GbA4}) ➔(MDSD):{MDSD1, MDSD2, MDSD3, DSD 4})
- Cluster Rule 5 (CR5) = {WbA: HbA:IbA:GbA} ➔({MDSD1, MDSD2, MDSD3, DSD4}) && (UPP-P:{Low, Medium, High}) && (IbA:{IbA1, IbA2, IbA3, IbA4}) ➔(IbA: {IbA1, IbA2, IbA3, IbA4}) ➔(GbA:{GbA1, GbA2, GbA3, GbA4}) ➔(MDSD):{MDSD1, MDSD2, MDSD3, DSD 4})
- Cluster Rule 6 (CR6) = {WbA: HbA:IbA:GbA} ➔({WbA1, WbA2, WbA3, WbA4}) || (UPP-P:{Low, Medium, High}) && (IbA:{IbA1, IbA2, IbA3, IbA4}) ➔(IbA:{IbA1, IbA2, IbA3, IbA4}) ➔(GbA:{GbA1, GbA2, GbA3, GbA4}) ➔(MDSD):{MDSD1, MDSD2, MDSD3, DSD 4})
- Cluster Rule 7 (CR7) = {WbA: HbA:IbA:GbA} ➔({HbA 1, HbA2, HbA3, HbA4}) || (UPP-P:{Low, Medium, High}) && (IbA:{IbA1, IbA2, IbA3, IbA4}) ➔(IbA:{IbA1, IbA2, IbA3, IbA4}) ➔(GbA:{GbA1, GbA2, GbA3, GbA4}) ➔(MDSD):{MDSD1, MDSD2, MDSD3, DSD 4})
- Cluster Rule 8 (CR8) = {WbA: HbA:IbA:GbA} ➔({IbA1, IbA2, IbA3, IbA4}) || (UPP-P:{Low, Medium, High}) && (IbA:{IbA1, IbA2, IbA3, IbA4}) ➔(IbA:{IbA1, IbA2, IbA3, IbA4}) ➔(GbA:{GbA1, GbA2, GbA3, GbA4}) ➔(MDSD):{MDSD1, MDSD2, MDSD3, DSD 4})
- Cluster Rule 9 (CR9) = {WbA: HbA:IbA:GbA} ➔({GbA1, GbA2, GbA3, GbA4}) || (UPP-P:{Low, Medium, High}) && (IbA:{IbA1, IbA2, IbA3, IbA4}) ➔(IbA:{IbA1, IbA2, IbA3, IbA4}) ➔(GbA:{GbA1, GbA2, GbA3, GbA4}) ➔(MDSD):{MDSD1, MDSD2, MDSD3, DSD 4})
- Cluster Rule 10 (CR10) = {WbA: HbA:IbA:GbA} ➔({MDSD1, MDSD2, MDSD3, DSD4}) || (UPP-P:{Low, Medium, High}) && (IbA:{IbA1, IbA2, IbA3, IbA4}) ➔(IbA:{IbA1, IbA2, IbA3, IbA4}) ➔(GbA:{GbA1, GbA2, GbA3, GbA4}) ➔(MDSD):{MDSD1, MDSD2, MDSD3, DSD 4}).

The API incorporates rules for filtering the behavior of existing apps. During installation, pre-approved settings are established, and synchronized apps operate accordingly. A plug-in is required for app-level activities and is synchronized with the apps.

The apps are synchronized with the rule-based API, which oversees their behavior. New plug-ins and updates are upgraded based on rules and threshold values specified in Tables 6 and 7. Figure 5 shows the mapping scenarios among third-party dependable and non-dependable libraries and plug-ins. The x-axis represents the five modes of mobile and web apps such as WbA, HbA, IbA, GbA, and MDSD. The y-axis represents the access control involvement ranges of third-party libraries and plug-ins from 0 to 100%.

Table 6. Integration of cluster rules and their filter levels.

Custer Rule1	Filter1—Level1 Integration	Filter2—Level2 Integration	Filter3—Level3 Integration	Filter4—Level4 Integration	Filter5—Level5 Integration
Cluster Rule1 (CR1)	{WbA:HbA:IbA: GbA}	({WbA1, WbA2, WbA3, WbA4}) && (UPP-P:{Low, Medium, High}) && (IbA:{IbA1, IbA2, IbA3, IbA4})	(IbA:{IbA1, IbA2, IbA3, IbA4})	(GbA:{GbA1, GbA2, GbA3, GbA4})	(MDSD):{MDSD1, MDSD2, MDSD3, DSD 4})
Cluster Rule2 (CR2)	{WbA:HbA:IbA:GbA}	({HbA 1, HbA2, HbA3, HbA4}) && (UPP-P:{Low, Medium, High}) && (IbA:{IbA1, IbA2, IbA3, IbA4})	(IbA:{IbA1, IbA2, IbA3, IbA4})	(GbA:{GbA1, GbA2, GbA3, GbA4})	(MDSD):{MDSD1, MDSD2, MDSD3, DSD 4})
Cluster Rule 3 (CR3)	{WbA:HbA:IbA:GbA}	({IbA1, IbA2, IbA3, IbA4}) && (UPP-P:Low, Medium, High}) && (IbA:{IbA1, IbA2, IbA3, IbA4})	(IbA:{IbA1, IbA2, IbA3, IbA4})	(GbA:{GbA1, GbA2, GbA3, GbA4})	(MDSD):{MDSD1, MDSD2, MDSD3, DSD 4})
Cluster Rule 4 (CR4)	{WbA:HbA:IbA:GbA}	({GbA1, GbA2, GbA3, GbA4}) && (UPP-P:{Low, Medium, High}) && (IbA:{IbA1, IbA2, IbA3, IbA4})	(IbA:{IbA1, IbA2, IbA3, IbA4})	(GbA:{GbA1, GbA2, GbA3, GbA4})	(MDSD):{MDSD1, MDSD2, MDSD3, DSD 4})
Cluster Rule 5 (CR5)	{WbA:HbA:IbA:GbA}	({MDSD1, MDSD2, MDSD3, DSD4}) && (UPP-P:{Low, Medium, High}) && (IbA:{IbA1, IbA2, IbA3, IbA4})	(IbA:{IbA1, IbA2, IbA3, IbA4})	(GbA:{GbA1, GbA2, GbA3, GbA4})	(MDSD):{MDSD1, MDSD2, MDSD3, DSD 4})
Cluster Rule 6 (CR6)	{WbA:HbA:IbA:GbA}	({WbA1, WbA2, WbA3, WbA4}) || (UPP-:{Low, Medium, High}) && (IbA:{IbA1, IbA2, IbA3, IbA4})	(IbA:{IbA1, IbA2, IbA3, IbA4})	(GbA:{GbA1, GbA2, GbA3, GbA4})	(MDSD):{MDSD1, MDSD2, MDSD3, DSD 4})
Cluster Rule 7 (CR7)	{WbA:HbA:IbA:GbA}	({HbA 1, HbA2, HbA3, HbA4}) || (UPP-P:{Low, Medium, High}) && (IbA:{IbA1, IbA2, IbA3, IbA4})	(IbA:{IbA1, IbA2, IbA3, IbA4})	(GbA:{GbA1, GbA2, GbA3, GbA4})	(MDSD):{MDSD1, MDSD2, MDSD3, DSD 4})
Cluster Rule 8 (CR8)	{WbA:HbA:IbA:GbA}	({IbA1, IbA2, IbA3, IbA4}) || (UPP-P:{Low, Medium, High}) && (IbA:{IbA1, IbA2, IbA3, IbA4})	(IbA:{IbA1, IbA2, IbA3, IbA4})	(GbA:{GbA1, GbA2, GbA3, GbA4})	(MDSD):{MDSD1, MDSD2, MDSD3, DSD 4})
Cluster Rule 9 (CR9)	{WbA:HbA:IbA:GbA}	({GbA1, GbA2, GbA3, GbA4}) || (UPP-P:{Low, Medium, High}) && (IbA:{IbA1, IbA2, IbA3, IbA4})	(IbA:{IbA1, IbA2, IbA3, IbA4})	(GbA:{GbA1, GbA2, GbA3, GbA4})	(MDSD):{MDSD1, MDSD2, MDSD3, DSD 4})
Cluster Rule 10 (CR10)	{WbA:HbA:IbA:GbA}	({MDSD1, MDSD2, MDSD3, DSD4}) || (UPP-P:{Low, Medium, High}) && (IbA:{IbA1, IbA2, IbA3, IbA4})	(IbA:{IbA1, IbA2, IbA3, IbA4})	(GbA:{GbA1, GbA2, GbA3, GbA4})	(MDSD):{MDSD1, MDSD2, MDSD3, DSD 4})

Table 7. Mapping of five different modes of apps with their involvement of access control range of mechanism to filter the TPLs and PIs.

	UPP-P-Low	UPP-P-Medium	UPP-P-High	HDA-P-Limited	HDA-P-Full	HDA-P-On-Demand	UI-LF-P-Simulated	UI-LF-P-Native	MP-D-P-No	MP-D-P-Yes	MP-D-P-Yes, but Not Guaranteed	WST-P-Yes	APIs-SWHW-P-Standard Interface	APIs-SWHW-P-High level Interface	APIs-SWHW-P-Hybrid High-Level Interface	APIs-SWHW-P-High-level Interface Special APIs	Problem Domain-Based Development App Functionality and Specifications	AI-P-Partial	AI-P-Full	SL-LI-P
WbA1	50	X	X	50	X	X	60	X	0	X	80	80	85	X	X	X	X	70	X	80
WbA2	60	X	X	70	X	X	70	X	0	X	100	100	100	X	X	X	X	70	X	80
WbA3	X	60	X	50	X	X	80	X	0	X	100	80	100	X	X	X	X	60	X	80
WbA4	X	75	X	50	X	X	100	X	60	X	100	100	X	X	100	X	X	X	100	90
WbA5	X	X	80	X	100	X	90	X	100	X	100	90	100	X	X	X	X	60	X	80
WbA6	X	X	90	X	100	X	100	X	100	X	100	100	X	X	100	X	X	X	100	80
HbA1	50	X	X	50	X	X	60	X	X	0	X	90	100	X	X	X	X	70	X	100
HbA2	50	X	X	50	X	X	60	X	X	0	X	100	X	X	100	X	X	70	X	100
HbA3	X	60	X	50	X	X	80	X	X	X	60	95	100	X	X	X	X	X	100	100
HbA4	X	75	X	40	X	X	X	60	X	100	X	100	X	X	X	100	X	X	100	100
HbA5	X	X	90	X	100	X	X	80	X	100	X	97	X	100	X	X	X	X	100	100
HbA6	X	X	100	X	X	100	X	100	X	100	X	100	X	X	X	X	90	X	100	100
IbA1	50	X	X	50	X	X	60	X	0	X	X	70	100	X	X	X	X	70	X	70
IbA2	50	X	X	50	X	X	60	X	0	X	X	100	100	X	X	X	X	X	100	70
IbA3	X	60	X	50	X	X	60	X	X	X	60	82	X	X	100	X	X	70	X	70
IbA4	X	75	X	40	X	X	X	60	X	100	X	100	X	X	X	100	X	X	100	70
IbA5	X	X	90	X	100	X	X	80	X	100	X	94	X	100	X	X	X	70	X	70
IbA6	X	X	100	X	X	100	X	100	X	100	X	100	X	X	X	X	90	X	100	70
GbA1	50	X	X	50	X	X	60	X	0	X	X	70	100	X	X	X	X	70	X	100
GbA2	50	X	X	50	X	X	60	X	0	X	X	100	100	X	X	X	X	X	100	100
GbA3	X	60	X	50	X	X	80	X	X	X	60	82	X	X	100	X	X	X	100	100
GbA4	X	75	X	40	X	X	X	60	X	100	X	100	X	X	X	100	X	X	100	100
GbA5	X	X	90	X	100	X	X	80	X	100	X	94	X	100	X	X	X	X	100	100
GbA6	X	X	100	X	X	100	X	100	X	100	X	100	X	X	X	X	90	X	100	100
MDSD1	50	X	X	50	X	X	60	X	0	X	X	76	100	X	X	X	X	70	X	85
MDSD2	50	X	X	50	X	X	60	X	0	X	X	100	100	X	X	X	X	X	100	85
MDSD3	X	60	X	50	X	X	80	X	X	X	60	89	X	X	100	X	X	X	100	85
MDSD4	X	75	X	40	X	X	X	60	X	100	X	100	X	X	X	100	X	X	100	85
MDSD5	X	X	90	X	100	X	X	80	X	100	X	98	X	100	X	X	X	X	100	85
MDSD6	X	X	100	X	X	90	X	100	X	100	X	100	X	X	X	X	100	X	100	85

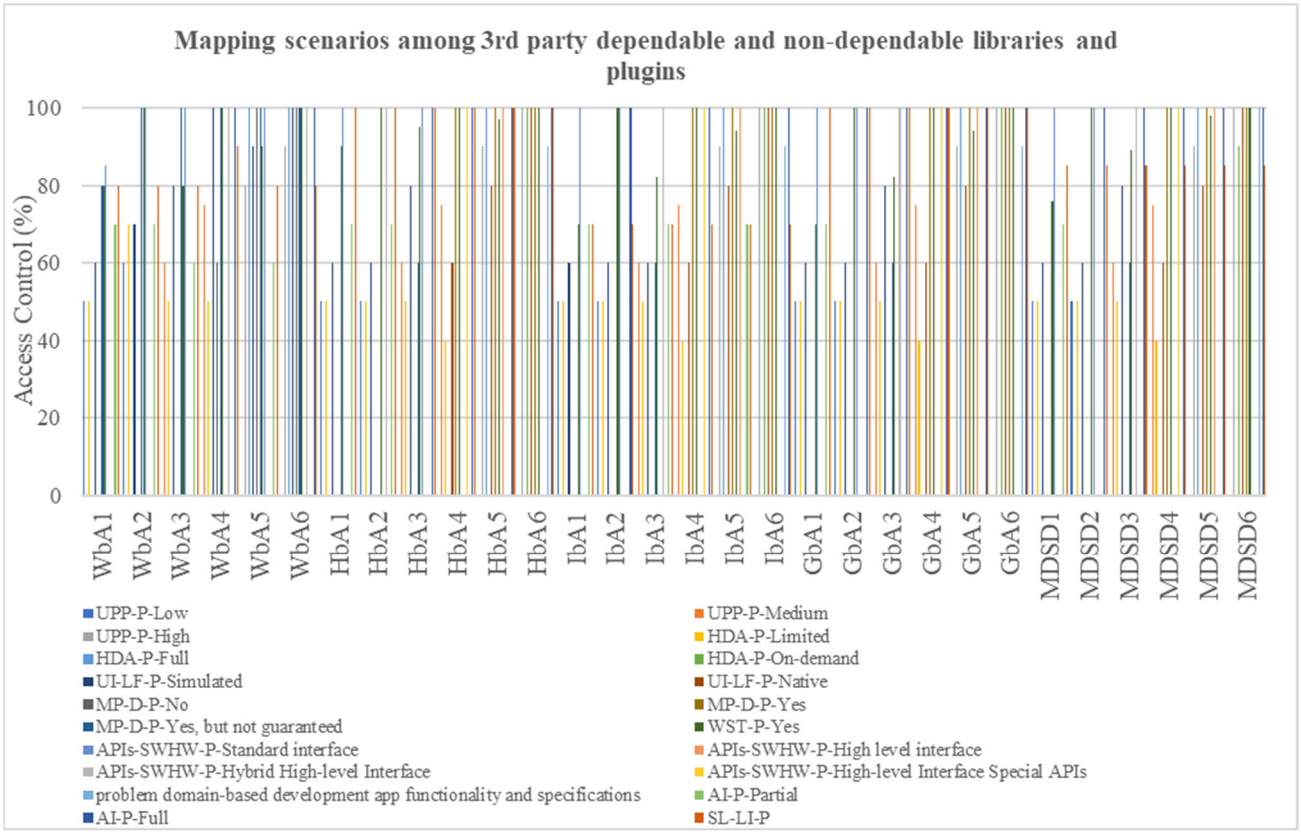

Figure 5. Mapping scenarios among 3rd TPLs' and PIs' state-level injection in SAS.

Table 6 and Figure 5 provide a visual representation of the mapping of five different modes of applications based on their access control range mechanisms for filtering TPLs and PIs. These mechanisms are supported by various parameters and data metrics such as UPP-P (low, medium, and high levels), HDA-P (limited and full versions), UI-LF-P (simulated and native), MP-D-P-No, MP-D-P-Yes, and MP-D-P-Yes but not guaranteed (binary status), WST-P-Yes, APIs-SWHW-P (standard, high level, hybrid high-level, high-level interface special APIs interfaces), AI-P (partial and full modes), and SL-LI-P. The mapping range is classified into two variations, auto-enable mode (AEM) and auto-disable mode (ADM), ranging from 1 to 100%. AEM ranges from 50% to 100%, while ADM represents 0% to 49%. The plug-ins are activated based on 72 rules, including 10 cluster rules from CR1 to CR10. The ADM mode is activated when the mapping percentage is 1% to 49%, represented as a cross mark (X) in Table 5.

An application's cluster and subcluster rules are integrated with key-log threshold values ranging from 0 to 100%, depending on the dependable and non-dependable features of the app's back end. End-users can access the app's data without needing access control of TPLs and privacy invasive PI components. A graphical representation of the mapping scenario between TPLs and PIs and state-level injections is shown in Figure 5, with an accumulated threshold of 60%. This model enables the automatic integration of feasible TPLs and PIs and the mapping of all matrices. If the mapping percentage is less than 60%, TPLs and PIs are blocked from accessing the end-user device credentials. The model was tested with over 95 types of TPLs and PIs and achieved a mapping rate of 94%.

The research analyzed end-users and client-based decision-making indicators, supporting factors, and data matrices. The goal was to facilitate accurate decisions regarding blocking unwanted libraries and plug-ins. The study involved experimenting with more than five web environments using specific rule-based parameters and supporting data matrices. To achieve this, the research proposed mapping techniques. It evaluated the accuracy of the proposed approach based on the number of different modes of apps, the

integration time of execution, the decision-making time, and the restriction of unwanted TPLs and PIs. The results showed that the proposed approach could accurately map all parameters with feasible matrices and achieved a mapping rate of 94%.

5. Conclusions

Using patterns and structures for UX/UI that can work across platforms was proposed to enhance the quality of cloud services, which can be utilized in domains such as medicine, science, and engineering. This approach can assist customers, businesses, end-users, and CC-S providers integrate multi-domain CC-Ss at different levels. A research study has suggested a methodological framework based on certain injection access constraints (D-ND-MAAC) representing state-level TPLs and PIs-based suspicious injections. This framework involves various cross-platform (CPF) development types such as WbAs, HbAs, IbAs, and GbAs in D-ND-MAAC, which are analyzed through CPMD-based security analysis systems. Experimental scenarios were built using Node JS with MongoDB integrations, third-party libraries, and plug-ins in a rule-based environment. The test cases comprised various end-users, client-based decision-making indicators, supporting factors affecting decision making, and data matrices to facilitate accurate decisions to avoid or block unwanted libraries and plug-ins. The study proposes precise mapping techniques by combining 72 and 10 cluster rules from CR1 to CR2 based on the end-user's active applications, regions, and customer needs. The mapping rate was measured based on the different modes of apps, integration execution time, decision-making time, restriction of unwanted TPLs and PIs, and the range of allowed wanted TPLs and PIs. The study successfully mapped all considered parameters with feasible matrices and achieved a 94% mapping rate.

Author Contributions: Conceptualization, S.K.H.; methodology, S.K.H. and R.R.; validation, M.R. and S.S.A.; formal analysis, G.U.M. and J.M.; writing—original draft preparation, S.K.H.; writing—review and editing, R.R. and M.R.; supervision, S.K.H., R.R. and M.R.; Funding acquisition, S.S.A. All authors have read and agreed to the published version of the manuscript.

Funding: This study was funded by the Deanship of Scientific Research, Taif University Researchers Supporting Project number (TURSP-2020/215), Taif University, Taif, Saudi Arabia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data in this research paper will be shared upon request with the corresponding author.

Acknowledgments: Authors would like to give thanks for the support of the Deanship of Scientific Research, Taif University, Taif, Saudi Arabia.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hashizume, K.; Rosado, D.G.; Fernández-Medina, E.; Fernandez, E.B. An analysis of security issues for cloud computing. *J. Internet Serv. Appl.* **2013**, *4*, 5–13. [CrossRef]
2. Zhao, G.; Liu, J.; Tang, Y.; Sun, W.; Zhang, F.; Ye, X.; Tang, N. Cloud Computing: A Statistics Aspect of Users. In *Cloud Computing, Proceedings of the First International Conference on Cloud Computing (CloudCom), Beijing, China, 1–4 December 2009*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 347–358.
3. Zhang, S.; Zhang, S.; Chen, X.; Huo, X. Cloud Computing Research and Development Trend. In Proceedings of the Second International Conference on Future Networks (ICFN'10), Sanya, China, 22–24 January 2010; IEEE Computer Society: Washington, DC, USA, 2010; pp. 93–97.
4. Cloud Security Alliance. Security Guidance for Critical Areas of Focus in Cloud Computing V3.0. 2011. Available online: https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf (accessed on 1 February 2023).
5. Marinos, A.; Briscoe, G. Community cloud computing. In *Cloud Computing, Proceedings of the First International Conference, CloudCom 2009, Beijing, China, 1–4 December 2009*; Proceedings 1; Springer: Berlin/Heidelberg, Germany, 2009; pp. 472–484.
6. Centre for the Protection of National Infrastructure. Information Security Briefing 01/2010 Cloud Computing. 2010. Available online: http://www.cpni.gov.uk/Documents/Publications/2010/2010007-ISB_cloud_computing.pdf (accessed on 1 February 2023).

7. Khalid, A. Cloud Computing: Applying issues in Small Business. In Proceedings of the International Conference on Signal Acquisition and Processing (ICSAP'10), Bangalore, India, 9–10 February 2010; pp. 278–281.
8. KPMG. From Hype to Future: KPMG's 2010 Cloud Computing Survey. 2010. Available online: http://www.techrepublic.com/whitepapers/from-hype-to-futurekpmgs-2010-cloud-computing-survey/2384291 (accessed on 1 February 2023).
9. Rosado, D.G.; Gómez, R.; Mellado, D.; Fernández-Medina, E. Security analysis in the migration to cloud environments. *Future Internet* **2012**, *4*, 469–487. [CrossRef]
10. Mather, T.; Kumaraswamy, S.; Latif, S. *Cloud Security and Privacy*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2009.
11. Kaur, G.; Vashisth, A.; Batth, R.S. X-Ortho: Fuzzy Rule Based Expert System for Diagnosing Infective Diseases of Hinge Joint Knee. In Proceedings of the 2019 International Conference on Automation, Computational and Technology Management (ICACTM), London, UK, 24–26 April 2019; IEEE: New York, NY, USA, 2019; pp. 518–524.
12. Dimple, S.; Agrawal, P.; Madaan, V. X-Tumour: Fuzzy Rule Based Medical Expert System to Detect Tumours in Gynaecology. *Int. J. Control Theory Appl.* **2016**, *9*, 5073–5084.
13. Li, W.; Ping, L. Trust model to enhance security and interoperability of Cloud environment. In *Cloud Computing, Proceedings of the 1st International conference on Cloud Computing, Beijing, China, 1–4 December 2009*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 69–79.
14. Rittinghouse, J.W.; Ransome, J.F. *Ransome. Cloud Computing: Implementation, Management, and Security*; CRC Press: Boca Raton, FL, USA, 2017.
15. Kalpana, G.; Kumar, P.V.; Krishnaiah, R.V.; Homomorphic Encryption Environment-Service Provider based Encryption and Decryption Endpoints for Third-party Cloud Provider. Int. J. *Comput. Sci. Inf. Secur.* **2017**, *15*, 7–15. Available online: https://sites.google.com/site/ijcsis/ (accessed on 1 February 2023).
16. Claesson, A. Securing Third-Party Dependencies in Development. Available online: https://www.mnemonic.no/globalassets/security-report/securing-third-party-dependencies-in-development.pdf (accessed on 2 February 2022).
17. Tebaa, M.; El Hajji, S.; El Ghazi, A. Homomorphic encryption method applied to Cloud Computing. In *National Days of Network Security and Systems (JNS2)*; IEEE Computer Society: Washington, DC, USA, 2012; pp. 86–89.
18. Naehrig, M.; Lauter, K.; Vaikuntanathan, V. Can homomorphic encryption be practical? In Proceedings of the 3rd ACM workshop on Cloud Computing Security workshop, Chicago, IL, USA, 21 October 2011; ACM: New York, NY, USA, 2011; pp. 113–124.
19. Xanthopoulos, S.; Xinogalos, S. A Comparative Analysis of Cross-platform Development Approaches for Mobile Applications. In Proceedings of the 6th Balkan Conference in Informatics, Thessaloniki, Greece, 19–21 September 2013; ACM 978-1-4503-1851-8/13/09.
20. Rieger, C.; Majchrzak, T.A. Towards the definitive evaluation framework for cross-platform app development approaches. *J. Syst. Softw.* **2019**, *153*, 175–199. [CrossRef]
21. Chaves, L.C.; Ismail, H.I.; Bessa, I.V.; Cordeiro, L.C.; de Lima Filho, E.B. Verifying fragility in digital systems with uncertainties using DSVerifier v2.0. *J. Syst. Softw.* **2019**, *153*, 22–43. [CrossRef]
22. Heitkötter, H.; Kuchen, H.; Majchrzak, T.A. Extending a model-driven cross-platform development approach for business apps. *Sci. Comput. Program.* **2015**, *97*, 31–36. [CrossRef]
23. Pustišek, M.; Umek, A.; Kos, A. Approaching the Communication Constraints of Ethereum-Based Decentralized Applications. *Sensors* **2019**, *19*, 2647. [CrossRef]
24. Akasiadis, C.; Pitsilis, V.; Spyropoulos, C.D. A Multi-Protocol IoT Platform Based on Open-Source Frameworks. *Sensors* **2019**, *19*, 4217. [CrossRef]
25. Palviainen, M.; Kuusijärvi, J.; Ovaska, E. Framework for End-User Programming of Cross-Smart Space Applications. *Sensors* **2012**, *12*, 14442–14466. [CrossRef]
26. Biørn-Hansen, A.; Grønli, T.-M.; Ghinea, G. Animations in Cross-Platform Mobile Applications: An Evaluation of Tools, Metrics and Performance. *Sensors* **2019**, *19*, 2081. [CrossRef]
27. Hang, L.; Kim, D.-H. Design and Implementation of an Integrated IoT Blockchain Platform for Sensing Data Integrity. *Sensors* **2019**, *19*, 2228. [CrossRef]
28. Acharya, S.; Rawat, U.; Bhatnagar, R. A Comprehensive Review of Android Security: Threats, Vulnerabilities, Malware Detection, and Analysis. *Secur. Commun. Netw.* **2022**, *2022*, 7775917. [CrossRef]
29. Alyas, T.; Ali, S.; Khan, H.U.; Samad, A.; Alissa, K.; Saleem, M.A. Container Performance and Vulnerability Management for Container Security Using Docker Engine. *Secur. Commun. Netw.* **2022**, *2022*, 6819002. [CrossRef]
30. Shammar, E.A.; Zahary, A.T.; Al-Shargabi, A.A. An Attribute-Based Access Control Model for Internet of Things Using Hyperledger Fabric Blockchain. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 6926408. [CrossRef]
31. Biørn-Hansen, A.; Grønli, T.-M.; Ghinea, G.; Alouneh, S. An Empirical Study of Cross-Platform Mobile Development in Industry. *Wirel. Commun. Mob. Comput.* **2019**, *2019*, 5743892. [CrossRef]
32. Van Ginkel, N.; De Groef, W.; Massacci, F.; Piessens, F. A Server-Side JavaScript Security Architecture for Secure Integration of Third-Party Libraries. *Secur. Commun. Netw.* **2019**, *2019*, 9629034. [CrossRef]
33. Dhiman, P.; Henge, S.K.; Singh, S.; Kaur, A.; Singh, P.; Hadabou, M. Blockchain merkle-tree ethereum approach in enterprise multi-tenant cloud environment. *Comput. Mater. Contin.* **2023**, *74*, 3297–3313.

34. Dhiman, P.; Henge, S.K.; Ramalingam, R.; Dumka, A.; Singh, R.; Gehlot, A.; Rashid, M.; Alshamrani, S.S.; AlGhamdi, A.S.; Alshehri, A. Secure Token–Key Implications in an Enterprise Multi-Tenancy Environment Using BGV–EHC Hybrid Homomorphic Encryption. *Electronics* **2022**, *11*, 1942. [CrossRef]
35. Dhiman, P.; Henge, S.K. Comparative Analysis of Cloud Security Complexities and Past Proposed Non-Homomorphic and Homomorphic Encryption Methodologies with Limitation. In Proceedings of the 4th International Conference on Information and Communication Technology for Competitive Strategies (ICTCS-2019), Udaipur, India, 13–14 December 2019; CRC Press: Boca Raton, FL, USA, 2019; pp. 787–799.
36. Dhiman, P.; Henge, S.K. Analysis of Blockchain Secure Models and Approaches Based on Various Services in Multi-tenant Environment. In *Recent Innovations in Computing. Lecture Notes in Electrical Engineering*; Singh, P.K., Singh, Y., Chhabra, J.K., Illés, Z., Verma, C., Eds.; Springer: Singapore, 2022; Volume 855. [CrossRef]
37. El-Kassas, W.S.; Abdullah, B.A.; Yousef, A.H.; Wahba, A.M. Taxonomy of Cross-Platform Mobile Applications Development Approaches. *Ain Shams Eng. J.* **2017**, *8*, 163–190. [CrossRef]
38. Hamza, A.A.; Halim, I.T.A.; Sobh, M.A.; Bahaa-Eldin, A.M. HSAS-MD Analyzer: A Hybrid Security Analysis System Using Model-Checking Technique and Deep Learning for Malware Detection in IoT Apps. *Sensors* **2022**, *22*, 1079. [CrossRef]
39. Nobakht, M.; Sui, Y.; Seneviratne, A.; Hu, W. PGFit: Static permission analysis of health and fitness apps in IoT programming frameworks. *J. Netw. Comput. Appl.* **2019**, *152*, 102509. [CrossRef]
40. Celik, Z.B.; McDaniel, P.; Tan, G. Soteria: Automated Iot Safety and Security Analysis. In Proceedings of the 2018 {USENIX} Annual Technical Conference, Boston, MA, USA, 11–13 July 2018; pp. 147–158.
41. Wang, Q.; Hassan, W.U.; Bates, A.; Gunter, C. Fear and Logging in the Internet of Things. In Proceedings of the Network and Distributed Systems Symposium, San Diego, CA, USA, 18–21 February 2018.
42. Celik, Z.B.; Tan, G.; McDaniel, P.D. IoTGuard: Dynamic Enforcement of Security and Safety Policy in Commodity IoT. In Proceedings of the NDSS, San Diego, CA, USA, 24–27 February 2019.
43. Tian, Y.; Zhang, N.; Lin, Y.H.; Wang, X.; Ur, B.; Guo, X.; Tague, P. Smartauth: User-Centered Authorization for the Internet of Things. In Proceedings of the 26th {USENIX} Security Symposium, Vancouver, BC, Canada, 5 May 2017; pp. 361–378.
44. Chen, J.; Diao, W.; Zhao, Q.; Zuo, C.; Lin, Z.; Wang, X.; Lau, W.C.; Sun, M.; Yang, R.; Zhang, K. IoTFuzzer: Discovering Memory Corruptions in IoT Through App-Based Fuzzing. In Proceedings of the NDSS, San Diego, CA, USA, 18–21 February 2018.
45. Roundy, K.A.; Miller, B.P. Hybrid analysis and control of malware. In *International Workshop on Recent Advances in Intrusion Detection*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 317–338.
46. Zhan, X.; Liu, T.; Fan, L.; Li, L.; Chen, S.; Luo, X.; Liu, Y. Research on Third-Party Libraries in Android Apps: A Taxonomy and Systematic Literature Review. *arXiv* **2021**, arXiv:2108.03787v1. [CrossRef]
47. Schneider, F.B. Enforceable Security Policies. *ACM Trans. Inf. Syst. Secur.* **2000**, *3*, 30–50. [CrossRef]
48. Flanagan, D. *JavaScript: The Definitive Guide*, 6th ed.; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2011.
49. Lekies, S.; Stock, B.; Johns, M. 25 Million Flows Later Large-scale Detection of DOM-based XSS. In Proceedings of the ACM Conference on Computer and Communications Security (CCS), Berlin, Germany, 4–8 November 2013.
50. Nikiforakis, N.; Invernizzi, L.; Kapravelos, A.; Van Acker, S.; Joosen, W.; Kruegel, C.; Piessens, F.; Vigna, G. You are what you include: Large-scale evaluation of remote Javascript inclusions. In Proceedings of the ACM Conference on Computer and Communications Security (CCS'12), Raleigh, CA, USA, 16–18 October 2012; pp. 736–747.