

Global Scientific and Academic Research Journal of Multidisciplinary Studies
ISSN: 2583-4088 (Online)
Frequency: Monthly
Published By GSAR Publishers
Journal Homepage Link- https://gsarpublishers.com/journals-gsarjms-home/

A Threat-Driven Methodology for Detecting and Mitigating OAuth 2.0 Misconfigurations and XSUAA Security Risks in SAP BTP Applications

By

Rajendra Asuri

Senior SAP BTP consultant

Article History

Received: 10/08/2024
Accepted: 25/08/2024
Published: 27/08/2024

Vol – 3 Issue –8

PP: - 112-126

Abstract

OAuth 2.0 misconfigurations and weaknesses in XSUAA can expose SAP Business Technology Platform (SAP BTP) applications to unauthorized access, token misuse, and inadequate authorization enforcement. This study examines a threat-driven methodology for identifying, assessing, and mitigating security risks associated with authentication, authorization, token handling, and identity configuration in SAP BTP environments. The research adopts a mixed-method approach, combining qualitative threat modeling and configuration analysis with quantitative assessment of identified vulnerabilities, likelihood, and potential impact. The methodology evaluates OAuth 2.0 authorization flows, token validation practices, scope assignments, role collections, trust configurations, and application security settings to identify weaknesses that may compromise enterprise resources. Findings are organized around common misconfiguration patterns, excessive privileges, insufficient token validation, and inconsistent enforcement of access policies. The proposed methodology integrates systematic threat identification, risk prioritization, security control verification, and mitigation planning into a repeatable assessment process. Its principal contribution is a structured approach that connects protocol-level weaknesses with SAP-specific identity and access management controls, enabling security teams to address vulnerabilities according to their operational significance. The framework also supports secure configuration reviews, improved audit readiness, and more consistent application of least-privilege principles. By establishing a practical relationship between threat analysis and preventive security measures, this study contributes to enterprise application security research and provides SAP architects, developers, and security professionals with an actionable foundation for strengthening OAuth 2.0 implementations and XSUAA configurations across SAP BTP applications.

Keywords: *OAuth 2.0, SAP Business Technology Platform, SAP BTP Security, XSUAA, Machine Learning, Supply Chain, SAP IBP, SAP GTS, SAP Transportation Management, Security Vulnerability Assessment, Threat Modeling, Authentication and Authorization, Access Token Security, Identity and Access Management, Role-Based Access Control, Security Misconfiguration, Vulnerability Mitigation, Security Risk Assessment, Least Privilege Principle, Enterprise Application Security, Secure Authorization Flows*

1. Introduction

The growing adoption of cloud-based enterprise applications has increased the importance of reliable identity management and access control within complex software environments. SAP Business Technology Platform (SAP BTP) provides capabilities for developing, integrating, and operating enterprise applications while supporting secure communication between users, services, and business resources. Within this environment, authentication and authorization mechanisms determine how identities are verified and how access permissions are enforced. OAuth 2.0 serves as a widely adopted authorization framework that enables applications to obtain controlled access to protected resources. However, its security depends on correct implementation, appropriate authorization policies, and consistent configuration across interconnected application components.

*Corresponding Author: **Muhammad Arslan Siddiqi**
© Copyright 2026 GSAR Publishers All Rights Reserved

SAP BTP applications commonly rely on the Authorization and Trust Management service, known as XSUAA, to support authentication integration, authorization enforcement, and token-based access control. XSUAA enables application developers to define security descriptors, establish scopes, configure role templates, and associate permissions with application users. These capabilities provide a structured foundation for managing access to enterprise resources, particularly when multiple applications and services interact within a distributed environment. Nevertheless, the availability of security mechanisms does not automatically guarantee their effective implementation. Incorrect scope definitions, excessive permissions, unsuitable trust settings, and incomplete token validation can weaken application security even when the underlying platform provides appropriate protective capabilities.

OAuth 2.0 and XSUAA security risks must therefore be examined as interconnected concerns rather than isolated configuration issues. An incorrectly defined authorization scope may grant broader access than an application requires, while an unsuitable role assignment may expose sensitive business functions to unauthorized users. Similarly, insufficient validation of access tokens can undermine confidence in the identity and permissions represented by those tokens. Inconsistent security settings across development, testing, and production environments may introduce further weaknesses that remain undetected during routine deployment activities. These conditions demonstrate that enterprise application security requires systematic examination of protocol behavior, identity configuration, authorization rules, and operational practices.

Although security documentation and established application security principles provide guidance for implementing OAuth 2.0, practical deployments can still contain weaknesses arising from configuration complexity and fragmented assessment procedures. Conventional security reviews may concentrate on individual settings, code-level vulnerabilities, or compliance checklists without adequately examining relationships between authorization flows, token properties, identity provider trust, and application-specific permissions. Such fragmented assessments can make it difficult to determine how several individually minor weaknesses might combine to create a significant security exposure. The central research problem addressed in this study is the absence of a sufficiently integrated, threat-driven assessment approach that connects OAuth 2.0 misconfigurations with XSUAA-specific security risks and prioritizes corrective action according to potential operational impact.

The motivation for this research arises from the need to make security assessments more systematic, repeatable, and relevant to enterprise application operations. Security teams must identify weaknesses before deployment where possible, distinguish meaningful vulnerabilities from configuration variations that do not create material risk, and establish practical remediation priorities. A threat-driven methodology can support these activities by beginning with protected assets, trust boundaries, and potential attack paths before examining the relevant security configurations. Such an approach also encourages assessment teams to consider the assumptions underlying authorization decisions, including whether applications request appropriate permissions and whether protected resources correctly validate the access presented to them. This study consequently treats security configuration analysis as an integral part of application risk management rather than an isolated technical checklist.

The primary objective of this study is to develop a structured methodology for detecting and mitigating OAuth 2.0 misconfigurations and XSUAA security risks in SAP BTP applications. The proposed approach seeks to identify relevant configuration weaknesses, classify their potential consequences, assess their relative severity, and associate each finding with suitable preventive or corrective controls. A further objective is to establish a repeatable relationship between threat identification, configuration inspection, risk assessment, and remediation verification. The study also examines how least-privilege enforcement, token validation, role assignment reviews, and trust configuration checks can contribute to stronger authorization controls. Together, these objectives aim to provide a practical assessment process that can be adapted to different SAP BTP application architectures without assuming that every deployment presents identical security requirements.

The research is guided by several related questions. Which OAuth 2.0 and XSUAA configuration weaknesses present the most relevant risks to the confidentiality, integrity, and availability of enterprise application resources? How can threat modeling be combined with configuration analysis to identify weaknesses that may remain overlooked during conventional security reviews? What criteria can support consistent classification and prioritization of identified vulnerabilities? How can remediation measures be mapped to specific weaknesses and subsequently verified through repeatable assessment procedures? These questions establish the analytical direction of the study and provide a basis for evaluating the usefulness of the proposed methodology. They also distinguish the identification of a technical weakness from the assessment of its practical significance within a particular application environment.

The proposed methodology is intended to contribute to both academic investigation and professional security practice. From a research perspective, it establishes a structured basis for examining the relationship between OAuth 2.0 authorization behavior, XSUAA configuration, and threat-oriented risk assessment. From an industry perspective, it offers SAP architects, application developers, and security professionals a consistent process for reviewing access permissions, validating security assumptions, documenting findings, and prioritizing remediation. Its practical significance lies in connecting technical observations with decisions that affect deployment readiness, operational risk, and security governance. The methodology may also support more consistent security reviews across development and

*Corresponding Author: **Muhammad Arslan Siddiqi** © Copyright 2026 GSAR Publishers All Rights Reserved

production environments, provided that assessment criteria are adapted to the application's architecture and validated against observable evidence.

The study ultimately argues that effective protection of SAP BTP applications requires more than the presence of established authorization services or the adoption of recommended configuration settings. Security depends on how these mechanisms are configured, how their interactions are assessed, and how weaknesses are corrected and verified throughout the application lifecycle. By bringing threat identification, OAuth 2.0 analysis, XSUAA configuration review, and risk-based remediation into a unified methodology, this research seeks to establish a practical foundation for more dependable enterprise authorization. The intended contribution is a repeatable and evidence-oriented assessment approach that can improve the identification of security weaknesses while supporting informed remediation decisions. Its effectiveness, however, must be established through systematic evaluation rather than assumed solely from the proposed design.

2. Background and Security Foundations

The security of enterprise applications deployed on SAP Business Technology Platform depends on the coordinated operation of identity management, authentication, authorization, and resource protection mechanisms. SAP BTP provides a cloud environment in which applications, services, and external identity systems interact to support business processes and enterprise integration. These interactions create multiple security boundaries that must be defined and enforced consistently. Application security therefore extends beyond verifying a user's identity to determining which resources the user or application may access and under what conditions that access is permitted. Understanding these foundational relationships is essential for identifying the configuration weaknesses that may arise in OAuth 2.0 implementations and XSUAA-managed authorization.

SAP BTP supports application development and execution through different runtime environments and service configurations, making security assessment dependent on the architecture of the individual deployment. In Cloud Foundry environments, the Authorization and Trust Management service, commonly referred to as XSUAA, provides capabilities for integrating identity providers and enforcing application authorization through OAuth-based mechanisms. Application developers can define security requirements through an application security descriptor, commonly represented by the xs-security.json file. This descriptor can specify scopes, role templates, role collections, and related authorization properties. Incorrect definitions within these components may result in unintended access permissions or inconsistent enforcement of application security requirements. Consequently, configuration review must consider both the descriptor itself and the way its definitions are used by application code and platform services.

OAuth 2.0 establishes a framework for delegated authorization in which a client obtains access to a protected resource through an authorization process involving an authorization server. The framework distinguishes the roles of resource owner, client, authorization server, and resource server, allowing their responsibilities to be distributed across different components. Access tokens represent authorization granted to a client and are presented when requesting protected resources. However, OAuth 2.0 does not independently guarantee that every implementation will validate tokens correctly or apply appropriate access restrictions. The security of an implementation depends on selecting suitable authorization flows, protecting client credentials, validating token properties, and enforcing the permissions associated with each request. A threat-driven assessment must therefore examine how these responsibilities are implemented within the target SAP BTP application.

Authorization code flow, client credentials flow, and other applicable OAuth 2.0 mechanisms serve different application requirements and trust relationships. Authorization code flow is generally associated with delegated access involving an end user, whereas client credentials flow supports access by a client acting on its own behalf. Selecting a flow that does not match the application's operational context can introduce unnecessary exposure or undermine the intended authorization model. Client authentication, redirect URI handling, credential protection, and the treatment of authorization responses must be reviewed according to the flow in use. Security assessment should also distinguish protocol-defined behavior from implementation-specific weaknesses. This distinction prevents legitimate architectural variations from being incorrectly classified as vulnerabilities while ensuring that deviations with meaningful security consequences receive appropriate attention.

Access tokens form another important foundation of the security model because they communicate authorization information between clients and protected resources. Depending on the implementation, token properties may include information relating to the issuer, audience, expiration, scopes, and authorized client or subject. Resource servers must validate the relevant properties according to the token format and the applicable platform configuration. Inadequate validation may allow a token to be accepted outside its intended context, while excessive token lifetime or inappropriate permission assignment may increase the consequences of credential compromise. Token security also involves protecting tokens from disclosure through logs, URLs, insecure storage, or unintended application responses. For SAP BTP applications, these considerations must be assessed alongside XSUAA configuration and the authorization checks performed by application endpoints.

XSUAA organizes application authorization through scopes and role templates that can be associated with role collections and assigned to users or other authorized identities. Scopes express permissions that an application can require or enforce, while role templates provide a mechanism for defining

*Corresponding Author: **Muhammad Arslan Siddiqi**
© Copyright 2026 GSAR Publishers All Rights Reserved

application roles that can subsequently be incorporated into role collections. Role collections support the assignment of related permissions to users through the relevant administrative configuration. These components establish a relationship between application-defined authorization requirements and the permissions granted within the platform. Nevertheless, the existence of a scope or role definition does not automatically demonstrate that the application enforces the associated restriction correctly. Developers must ensure that protected operations perform appropriate authorization checks and that the resulting permission model follows the principle of least privilege.

Identity provider integration and trust configuration introduce additional dependencies into the authorization process. SAP BTP applications may rely on configured identity providers to authenticate users and establish the identity information used in subsequent authorization decisions. Trust relationships must reflect the intended organizational and application boundaries, including which identities are accepted and how user attributes are mapped into the application's security context. Incorrect trust settings, unsuitable attribute mappings, or overly broad access assignments can weaken the relationship between authenticated identity and permitted action. These risks are particularly relevant where several applications, user groups, or enterprise services share an environment. A complete assessment should therefore consider identity configuration, role assignment, and resource-level authorization together rather than treating each component as an independent security control.

Threat modeling provides a systematic means of examining how an attacker might exploit weaknesses in these interacting components. Established approaches such as STRIDE classify threats in terms of spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege. Applied to OAuth 2.0 and XSUAA, these categories help structure the investigation of token misuse, unauthorized permission changes, insufficient audit evidence, exposure of protected information, service disruption, and privilege escalation. Threat modeling begins by identifying valuable assets, application entry points, trust boundaries, and security assumptions. The resulting threat scenarios can then guide configuration inspection and targeted validation. Risk assessment should consider the likelihood of exploitation, the potential business impact, and the effectiveness of existing controls, while avoiding unsupported claims about vulnerabilities that have not been demonstrated.

These foundations establish the technical basis for the threat-driven methodology developed in this research. OAuth 2.0 defines the delegated authorization framework, XSUAA supplies platform-specific identity and authorization capabilities, and application-level controls determine how permissions are enforced during actual resource access. Weaknesses may arise from individual configuration errors or from inconsistencies between these layers, making integrated assessment more informative than isolated checklist verification. The proposed approach consequently combines security descriptor inspection, authorization flow review, token validation analysis, trust configuration assessment, and risk-based remediation planning. This foundation also provides a basis for subsequent evaluation of detection coverage, prioritization consistency, and remediation effectiveness. The central premise is that dependable application security requires verifiable alignment between intended authorization policies, platform configuration, and the behavior of protected application resources.

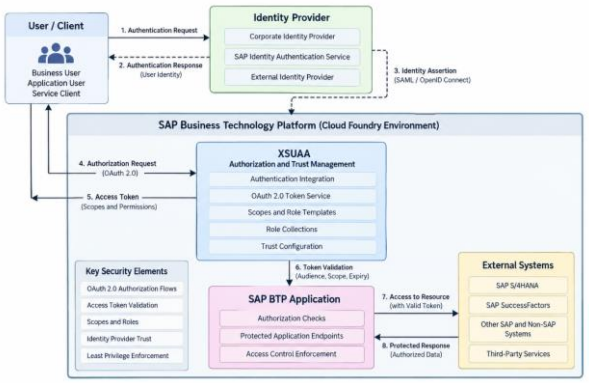


Figure 1: SAP BTP Identity and Authorization Architecture

3. Research Methodology and Threat Modeling

The research methodology establishes a systematic approach for identifying, analyzing, and mitigating OAuth 2.0 misconfigurations and XSUAA security risks in SAP Business Technology Platform applications. The study adopts a mixed-method research design that combines qualitative threat modeling with structured configuration assessment and quantitative risk evaluation. The qualitative component examines application architecture, trust boundaries, authorization responsibilities, and plausible attack scenarios. The quantitative component provides a consistent basis for classifying identified weaknesses according to their likelihood, potential impact, and exposure within the application environment. This combination is intended to connect technical security observations with practical remediation decisions. The methodology focuses on producing repeatable assessment procedures rather than assuming that the presence of platform security services guarantees effective protection.

The first stage defines the assessment scope by identifying the application components, identity services, protected resources, and authorization mechanisms relevant to the deployment. Particular attention is given to SAP BTP Cloud Foundry environments that use XSUAA for application authentication integration and authorization management. The assessment boundary includes application security descriptors, OAuth 2.0 client settings, scopes, role templates, role collections, trust configurations, and resource-level authorization checks. Dependencies on external identity providers and connected enterprise services are included where they influence authentication or access decisions. This scope definition establishes which configurations can be inspected directly and

*Corresponding Author: **Muhammad Arslan Siddiqi**
© Copyright 2026 GSAR Publishers All Rights Reserved

which behaviors require application-level testing. It also prevents conclusions from being generalized to SAP BTP environments or security services that have not been examined.

Asset identification follows scope definition because the significance of a configuration weakness depends partly on the resources it may expose. Relevant assets include business data, application programming interfaces, administrative functions, service credentials, access tokens, and identity attributes. Each asset is associated with its expected confidentiality, integrity, and availability requirements. Trust boundaries are then mapped between users, client applications, identity providers, XSUAA, application components, and downstream services. These boundaries identify where identity information or authorization decisions cross from one component to another. The resulting architecture model provides the foundation for threat identification by revealing potential entry points, privileged operations, and dependencies that may be overlooked when individual configuration files are reviewed in isolation.

Threat identification uses a structured threat modeling process informed by established security analysis principles. STRIDE provides a useful classification scheme covering spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege. In the context of OAuth 2.0 and XSUAA, spoofing may involve misuse of compromised credentials or tokens, while tampering may concern unauthorized changes to security configuration. Repudiation highlights inadequate audit evidence, information disclosure concerns unintended access to protected data, and denial of service addresses conditions that affect application availability. Elevation of privilege is particularly relevant where excessive scopes, inappropriate role assignments, or insufficient authorization checks permit actions beyond an identity's intended permissions. These categories organize the investigation without implying that every threat is present in every deployment.

Following threat identification, the methodology examines configuration weaknesses and their potential exploitation paths. The inspection process includes reviewing OAuth 2.0 authorization flow selection, client authentication requirements, redirect URI restrictions where applicable, token handling, and resource-level access enforcement. XSUAA-related checks examine the consistency of scopes, role templates, role collections, and trust settings with the application's documented security requirements. Token validation is assessed according to the token format and implementation, including relevant properties such as issuer, audience, expiration, and authorization claims. The review also considers whether protected application endpoints enforce the expected permissions instead of relying solely on successful authentication. Findings are recorded with their affected component, observed condition, expected security behavior, potential consequence, and supporting evidence.

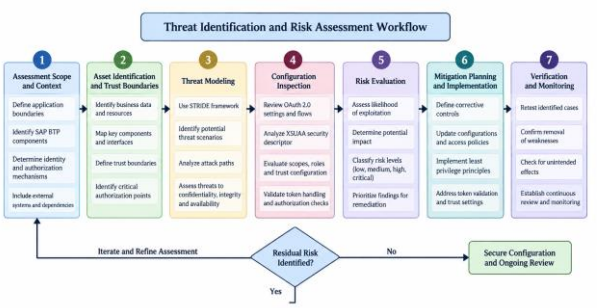


Figure 2: Threat Identification and Risk Assessment Workflow

Risk evaluation provides a common basis for prioritizing findings identified during the inspection process. Each finding is assessed according to the likelihood of exploitation and the potential consequences for confidentiality, integrity, availability, and business operations. Where appropriate, the assessment considers exposure of the affected component, privileges required for exploitation, the presence of compensating controls, and the sensitivity of the protected resource. A risk matrix can then group findings into categories such as low, medium, high, and critical. The classification is intended to guide remediation rather than replace detailed technical judgment. Where evidence is incomplete, uncertainty should be recorded explicitly instead of assigning an apparently precise score unsupported by the available observations.

Quantitative evaluation is designed to measure the consistency and usefulness of the proposed assessment process. Relevant measures include the number of identified misconfigurations, the proportion of predefined test cases detected, detection precision, recall, false-positive rate, and the time required to complete assessment and remediation verification. Detection precision represents the proportion of reported findings that are confirmed as valid, whereas recall represents the proportion of known test-case weaknesses that are successfully identified. These measures require a documented test set and an independent means of establishing which cases contain genuine weaknesses. Comparisons with a baseline review process should use equivalent configurations, test conditions, and evaluation criteria. If no controlled experiment has been conducted, the methodology defines these measures for future evaluation rather than presenting hypothetical results as empirical evidence.

Mitigation planning connects each confirmed finding with a corrective control and a verification procedure. Excessive scopes may require narrowing permissions, while inappropriate role assignments may require revision of role collections and access responsibilities. Weak token validation requires correction of the relevant validation logic or configuration, followed by tests that establish whether invalid, expired, or incorrectly scoped tokens are rejected. Trust configuration weaknesses require examination of identity provider settings and the intended boundaries of accepted identities. After remediation, the affected test cases are repeated to determine whether the original weakness has been removed and whether the change has introduced unintended

*Corresponding Author: **Muhammad Arslan Siddiqi**
© Copyright 2026 GSAR Publishers All Rights Reserved

effects. Evidence should be retained to support reproducibility, auditability, and comparison between the original and corrected configurations.

The validity of the methodology depends on the accuracy of the test environment, the completeness of the selected threat scenarios, and the reliability of the evidence used to classify findings. Configuration behavior may vary according to application architecture, service settings, identity provider integration, and platform-specific implementation details. A test set limited to known misconfigurations may also fail to represent every weakness encountered in operational environments. To reduce these limitations, the assessment should document assumptions, use repeatable test cases, distinguish confirmed vulnerabilities from theoretical risks, and record deviations from the intended procedure. Independent review of a sample of findings can further improve consistency. These safeguards establish a transparent basis for evaluating the methodology and interpreting its results without overstating the level of security assurance provided.

Figure 2 should be placed after the methodology overview and before the detailed threat identification discussion. It should illustrate the sequence from assessment scope and asset identification through threat modeling, configuration inspection, risk evaluation, mitigation, and verification. The diagram should show the possibility of returning to earlier assessment stages when verification reveals unresolved weaknesses. This workflow emphasizes that threat-driven assessment is iterative rather than a one-time configuration review. The methodology therefore provides the analytical foundation for the subsequent examination of OAuth 2.0 and XSUAA vulnerabilities, enabling the study to connect individual technical weaknesses with their exploitation conditions, risk significance, and appropriate corrective measures.

Table 1: Threat Classification and Security Control Mapping

Threat category	**Potential weakness in SAP BTP**	**Recommended security control**
Spoofing	Inadequate client authentication or token verification	Verify token authenticity and client identity
Tampering	Unauthorized changes to XSUAA security settings	Restrict configuration changes and review security descriptors
Repudiation	Insufficient records of access and authorization events	Maintain security logs and audit trails
Information disclosure	Excessive scopes or permissions	Apply least privilege and enforce access checks
Denial of service	Unprotected application endpoints	Monitor requests and apply appropriate limits
Elevation of privilege	Incorrect role assignments or missing authorization checks	Review role collections and validate permissions
Token misuse	Incomplete validation of token claims	Verify issuer, audience, expiration, and applicable scopes
Trust misconfiguration	Incorrect identity provider or attribute mapping settings	Validate identity provider trust and identity mappings

4. OAuth 2.0 and XSUAA Vulnerability Analysis

OAuth 2.0 and XSUAA security weaknesses can emerge from interactions between authorization policies, token handling, identity configuration, and application-level access enforcement. SAP Business Technology Platform provides mechanisms for managing application permissions and integrating identity services, but their effectiveness depends on appropriate configuration and implementation. A weakness in one component may affect the security of other connected components, particularly when applications share identity services or protected resources. Consequently, vulnerability assessment must examine both individual security settings and the relationships between them. This section analyzes the principal risks associated with OAuth 2.0 and XSUAA, their potential consequences, and the controls required to reduce exposure.

Authorization flow misconfiguration represents a significant area of concern because OAuth 2.0 flows operate under different client and trust assumptions. Applications serving end users require an appropriate delegated authorization mechanism, whereas service-to-service communication may use client credentials when access is performed on behalf of the client itself. Selecting an unsuitable flow or failing to protect client credentials can weaken the intended authorization boundary. Other concerns include insufficient redirect URI restrictions, inadequate client authentication, and

*Corresponding Author: **Muhammad Arslan Siddiqi**
© Copyright 2026 GSAR Publishers All Rights Reserved

improper handling of authorization responses. These weaknesses may expose applications to unauthorized access or misuse of issued permissions. Assessment should therefore establish whether the selected flow matches the application architecture and whether the associated security requirements are enforced consistently.

Scope misconfiguration can introduce excessive access privileges when applications request or receive permissions beyond their operational requirements. Scopes define the permissions associated with protected operations, but their presence alone does not guarantee that the application enforces them correctly. An endpoint that fails to verify the required scope may permit an authenticated client to perform an operation outside its intended authority. Similarly, broadly defined permissions can increase the consequences of token compromise. The principle of least privilege provides a practical foundation for reducing this exposure by limiting access to the resources and functions necessary for each application or identity. In SAP BTP, scope definitions should be reviewed alongside application authorization logic and documented business requirements.

Access token validation is another essential security consideration because protected resources rely on tokens to make authorization decisions. Weak validation may occur when applications fail to verify relevant token properties or accept tokens outside their intended context. Depending on the token format and implementation, important checks can include authenticity, issuer, audience, expiration, and applicable authorization claims. A token that is valid for one resource should not automatically be accepted by another resource with different security requirements. Applications using XSUAA should rely on supported validation mechanisms and ensure that authorization decisions are made correctly at protected endpoints. Controlled testing with expired tokens, invalid tokens, incorrect audiences, and insufficient permissions can help establish whether the expected restrictions are effective.

Role assignment weaknesses can undermine otherwise appropriate scope definitions. XSUAA supports application authorization through scopes, role templates, and role collections, which connect application-defined permissions with assigned identities. Excessive role assignments, outdated privileges, and unsuitable role mappings may allow users to access sensitive business functions or administrative operations without an appropriate operational requirement. These problems may become more serious when role assignments are not reviewed after organizational or application changes. A structured assessment should compare assigned permissions with actual responsibilities and test whether users with limited privileges are prevented from executing restricted operations. Regular access reviews, clear ownership of privileged roles, and controlled permission changes can help maintain the intended authorization model.

Identity provider trust configuration introduces additional risks when authentication sources or identity attributes are not aligned with the application's security requirements. Incorrect trust settings, unsuitable attribute mappings, and overly permissive identity acceptance may cause authentication information to be interpreted incorrectly during authorization. The resulting exposure depends on the environment's identity architecture and the permissions associated with affected identities. Assessment should verify which identity providers are trusted, how relevant attributes are mapped, and whether those attributes influence role assignments or access decisions. Changes to trust configuration should be documented, reviewed, and tested before deployment. Authentication success should not be treated as sufficient evidence that the resulting access permissions are appropriate.

Configuration drift can create further weaknesses as applications move through development, testing, and production environments. Differences in security descriptors, scopes, role assignments, and identity settings may introduce inconsistencies between the intended security design and the deployed configuration. For example, permissions granted temporarily during testing may remain active after deployment, or changes to application security settings may not be reflected consistently across dependent components. Such differences require investigation because not every configuration deviation represents an exploitable vulnerability. Comparing deployed settings with an approved baseline helps distinguish harmless environmental differences from changes that weaken security boundaries. Controlled configuration management and repeatable post-deployment testing can reduce the likelihood of unnoticed authorization weaknesses.

Effective vulnerability analysis must connect identified weaknesses with their potential consequences and appropriate corrective controls. Risk prioritization should consider the likelihood of exploitation, the sensitivity of affected resources, the privileges required, the accessibility of the vulnerable component, and the effectiveness of existing safeguards. Excessive scopes may require permission reduction and endpoint testing, while token validation weaknesses may require corrections to validation logic and renewed negative testing. Role assignment issues may be addressed through access reviews, and trust weaknesses may require changes to identity provider settings or attribute mappings. Each finding should include supporting evidence, a risk classification, a recommended correction, and a verification procedure. Quantitative claims about detection performance or risk reduction should be based on actual assessment results rather than assumed improvements.

The relationship between SAP BTP security components, principal OAuth 2.0 and XSUAA vulnerability areas, and their potential impacts. This relationship demonstrates why isolated configuration checks may be insufficient when several weaknesses affect the same authorization path. A misconfigured scope, for example, may create greater exposure when combined with excessive role assignments or incomplete endpoint authorization checks. The analysis therefore supports an integrated approach that combines threat identification, configuration inspection, risk prioritization, and remediation verification. These findings establish the basis for

 © Copyright 2026 GSAR Publishers All Rights Reserved

the proposed detection and mitigation framework, which seeks to make security assessment more systematic and repeatable. Its effectiveness must ultimately be established through controlled testing and documented evidence from the application environments being assessed.

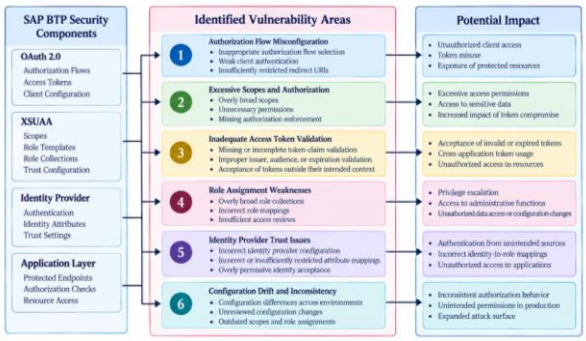


Figure 3: OAuth 2.0 and XSUAA Attack Surface Map

5. Proposed Threat-Driven Detection and Mitigation Framework

The proposed threat-driven detection and mitigation framework establishes a systematic process for identifying OAuth 2.0 misconfigurations and XSUAA security risks in SAP Business Technology Platform applications. Its central principle is that effective security assessment should connect application architecture, threat scenarios, configuration evidence, risk prioritization, and corrective action within a single assessment process. Rather than treating each configuration setting as an isolated checkpoint, the framework examines how identity management, token handling, permission assignment, and resource-level authorization interact. This approach enables security teams to distinguish configuration deviations from weaknesses that could create meaningful exposure. The framework is designed as a repeatable assessment model that can be adapted to different SAP BTP application requirements without assuming that every environment presents identical security risks.

The first stage establishes an application security baseline by collecting the configuration information required for assessment. Relevant inputs include the application security descriptor, OAuth 2.0 client settings, scope definitions, role templates, role collections, identity provider trust settings, and the authorization checks implemented by protected application endpoints. The assessment also identifies critical business resources, privileged operations, external dependencies, and trust boundaries. These inputs are compared with documented application requirements to establish the intended security posture. Configuration collection should use authorized administrative interfaces, approved source repositories, and documented inspection procedures. Sensitive credentials and access tokens must be excluded from ordinary assessment records or handled through appropriately protected mechanisms. Establishing this baseline provides the evidence needed to identify deviations and determine whether they introduce security concerns.

Threat identification follows the baseline assessment and focuses on plausible attack paths through the application's identity and authorization architecture. Threat modeling principles, including STRIDE, provide a structured means of considering identity spoofing, unauthorized configuration changes, insufficient auditability, information disclosure, service disruption, and privilege escalation. Each threat scenario is linked to the relevant application component, required conditions, and potential security consequence. For example, an overly broad scope may become more consequential when a protected endpoint fails to enforce the intended permission check. Similarly, a weak trust configuration may create exposure when identity attributes influence privileged role assignments. This stage prioritizes realistic scenarios grounded in the application's architecture rather than producing a generic list of threats that may not apply to the deployment.

The detection stage examines the configurations and application behaviors associated with the identified threat scenarios. Rule-based checks can identify conditions such as unnecessarily broad scopes, unexpected role assignments, missing configuration elements, or inconsistencies between approved security settings and deployed descriptors. Application-level tests complement configuration inspection by examining whether protected endpoints reject requests that lack the required authorization. Token validation tests should reflect the token format and the applicable security implementation, including relevant issuer, audience, expiration, and permission checks. Each detection rule should specify its purpose, required evidence, expected condition, and criteria for reporting a finding. This structure makes the assessment more reproducible and helps reduce false positives arising from legitimate environment-specific configurations.

The framework may incorporate machine learning as an optional supporting capability for identifying unusual configuration patterns or prioritizing findings across larger application environments. Such a component could compare observed configurations with an appropriately constructed baseline and flag deviations for further investigation. However, machine learning should not replace deterministic validation of security requirements or independently establish that a configuration is vulnerable. Its usefulness depends on the quality, representativeness, and labeling of the available data, as well as the explainability of its outputs. A flagged anomaly must be reviewed against the application's architecture and documented security requirements before it is classified as a confirmed weakness. Where suitable training data or validation evidence is unavailable, rule-based detection and expert review provide a more defensible foundation for assessment.

Risk prioritization converts validated findings into a practical remediation order. The proposed framework evaluates each finding according to its potential impact, likelihood of exploitation, exposure of the affected resource, privileges required, and effectiveness of existing controls. These factors can be combined into a documented risk matrix with categories such as low, medium, high, and critical. The

*Corresponding Author: **Muhammad Arslan Siddiqi**
© Copyright 2026 GSAR Publishers All Rights Reserved

classification should reflect evidence and stated assumptions rather than imply an unsupported degree of numerical precision. Findings affecting sensitive information or privileged administrative operations may warrant greater urgency when exploitation is plausible and compensating controls are weak. By contrast, configuration differences that do not weaken an actual security boundary may require documentation rather than immediate correction. This distinction allows remediation effort to focus on weaknesses with the greatest demonstrated significance.

Mitigation planning maps each confirmed finding to a corrective control and an explicit verification procedure. Excessive scopes should be reduced to the permissions required for legitimate application functions, while role assignment weaknesses should be addressed through role collection reviews and least-privilege enforcement. Token validation weaknesses require correction of the applicable validation process and testing to ensure that unsuitable tokens are rejected. Identity provider trust issues may require changes to approved identity sources, attribute mappings, or related authorization configuration. Configuration drift should be addressed by restoring alignment with an approved baseline and strengthening change management procedures. Each corrective action should have an accountable owner, a defined completion condition, and evidence demonstrating whether the weakness has been resolved. Remediation should also be tested for unintended effects on legitimate application operations.

Verification and continuous review complete the assessment cycle. After a corrective change, the original test case should be repeated under equivalent conditions to determine whether the identified weakness remains exploitable. Additional regression tests should examine adjacent authorization paths because a configuration change can affect related permissions or application functions. Findings should be closed only when the verification evidence satisfies the predefined acceptance criteria. Unresolved weaknesses should return to the risk assessment stage for further investigation or additional remediation. The framework also supports periodic reviews of security descriptors, role assignments, trust settings, and application authorization behavior to identify changes introduced after deployment. This iterative process helps maintain alignment between the intended security design and the application's operational configuration.

It should be placed after this section's opening explanation of the framework. It should illustrate the progression from security baseline collection through threat identification, rule-based detection, risk prioritization, mitigation, and verification, with a feedback path for unresolved findings. The figure should distinguish optional machine learning-assisted anomaly prioritization from mandatory security validation, making clear that automated observations require appropriate interpretation and verification. The proposed framework contributes a structured way to connect threat scenarios with configuration evidence and corrective controls, while preserving the distinction between suspected weaknesses and confirmed vulnerabilities. Its practical effectiveness should be evaluated through controlled test cases and documented measurements of detection coverage, precision, false-positive rate, and remediation outcomes. The following evaluation section can then examine these measures against a defined baseline without assuming improvements that have not been empirically established.

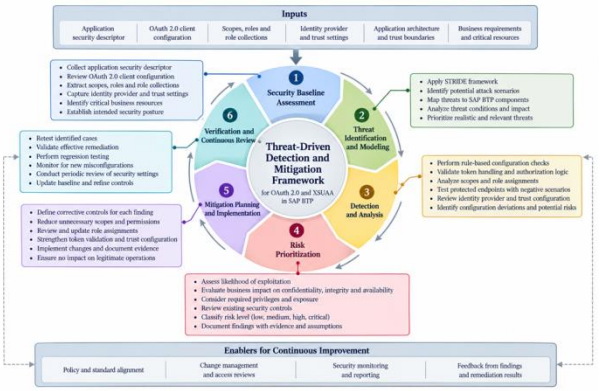


Figure 4: Threat-Driven Detection and Mitigation Framework

6. Evaluation, Results, and Comparative Analysis

The evaluation of the proposed threat-driven framework requires a structured assessment of its ability to identify OAuth 2.0 misconfigurations and XSUAA security risks within SAP Business Technology Platform applications. The primary objective is to determine whether the framework provides a repeatable means of detecting relevant weaknesses, classifying their potential impact, and linking confirmed findings to appropriate corrective controls. The evaluation considers configuration inspection, authorization testing, risk prioritization, and remediation verification as connected activities. Rather than assuming that the proposed approach improves security outcomes, the assessment defines measurable criteria through which its effectiveness can be examined. This distinction is important because a conceptual framework may provide a useful methodology without necessarily demonstrating superior detection performance until it has been evaluated experimentally.

The evaluation design begins by defining a controlled application environment that represents relevant SAP BTP security components and their authorization relationships. The environment should include an application security descriptor, OAuth 2.0 client configuration, XSUAA scopes and role templates, role collections, identity provider trust settings, and protected application endpoints. A reference configuration should be established before introducing controlled variations representing potential security weaknesses. These variations may include excessive scopes, inappropriate role assignments, incomplete token validation, unsuitable trust settings, and inconsistencies between the intended security policy and application behavior. Each test case should document the affected component, expected security behavior, conditions under which a weakness may become exploitable, and the evidence required to confirm a finding. This approach enables

*Corresponding Author: **Muhammad Arslan Siddiqi**
© Copyright 2026 GSAR Publishers All Rights Reserved

the assessment to distinguish known test conditions from vulnerabilities discovered during subsequent analysis.

The baseline comparison should use a conventional configuration review process as the reference approach. Such a review may involve manual inspection of security descriptors, administrative settings, role assignments, and selected application authorization checks. The proposed framework adds a structured relationship between threat modeling, configuration analysis, risk classification, and remediation verification. To make the comparison meaningful, both approaches should examine equivalent configurations and use the same predefined test cases and confirmation criteria. Differences in assessor experience, test coverage, access permissions, and available evidence should be documented because they may influence the observed outcomes. The comparison should not assume that manual reviews are inherently ineffective or that structured automation is necessarily more accurate. Instead, it should examine whether the proposed method improves consistency, traceability, and the identification of weaknesses under the specified evaluation conditions.

Detection performance can be examined through detection coverage, precision, recall, and false-positive rate. Detection coverage represents the proportion of predefined assessment cases successfully examined by the method, while precision measures the proportion of reported findings confirmed as valid. Recall measures the proportion of known positive test cases correctly identified, and the false-positive rate measures the proportion of negative cases incorrectly reported as vulnerable. These metrics provide complementary perspectives because a method that identifies many potential weaknesses may still generate excessive false alarms. Each measure requires a documented test set and an independent reference classification. For example, a test case involving an expired access token should be considered correctly detected only when the assessment identifies the relevant validation weakness according to predefined criteria. Without completed testing, these metrics remain evaluation measures rather than evidence of achieved performance.

Risk prioritization should be evaluated separately from the detection process because identifying a weakness does not automatically establish its operational significance. The proposed framework considers exploitation likelihood, potential impact, required privileges, resource sensitivity, and the effectiveness of existing controls. These factors can support classification into low, medium, high, and critical risk categories, provided that the scoring rules are documented and consistently applied. The evaluation should examine whether different assessors reach reasonably consistent classifications when presented with the same evidence. It should also investigate whether high-impact weaknesses affecting privileged operations or sensitive resources receive appropriate attention. A risk assessment should not equate configuration deviation with confirmed exploitability, and uncertainty should be recorded when the available evidence is insufficient. This approach supports more defensible remediation decisions and reduces the risk of prioritizing findings solely because they are easy to detect.

The figure should present a comparative chart of the baseline review and proposed framework across selected detection metrics, such as precision, recall, and detection coverage. The horizontal axis should identify the evaluation metrics, while separate series represent the two assessment methods. The vertical axis may represent percentages where the metrics are calculated as proportions. All displayed values must be derived from actual assessment results collected under equivalent conditions. If the experiment has not been conducted, the figure should be presented as a proposed evaluation design or populated with clearly identified illustrative values, never as measured evidence. A meaningful interpretation should discuss differences between the methods, uncertainty in the measurements, and the extent to which the test cases represent practical SAP BTP deployments. The chart should also avoid implying general superiority based on a limited or unrepresentative set of configurations.

The analysis of remediation outcomes should examine whether corrective actions eliminate the original weakness without introducing unacceptable effects on legitimate application behavior. After a change to scopes, role collections, token validation, or trust configuration, the original test should be repeated using equivalent conditions. Regression tests should then verify that authorized operations continue to function and that related protected resources remain subject to the intended access controls. Relevant measures include the proportion of findings successfully remediated, the number of unresolved findings, the time required to verify corrections, and the occurrence of unintended authorization changes. These measures can help determine whether the framework supports an effective remediation process rather than merely producing a list of suspected weaknesses. However, conclusions about improved remediation speed or security outcomes require recorded observations from an actual assessment.

Several limitations must be considered when interpreting the evaluation. A limited set of test cases may not represent the full range of application architectures, identity provider configurations, and deployment conditions found across SAP BTP environments. The results may also depend on the accuracy of the reference classifications, the quality of the configuration evidence, and the experience of the assessors. If machine learning-assisted analysis is included, its performance requires separate evaluation using suitable data, independent validation, and explicit treatment of false positives. Findings from a controlled environment should therefore not be generalized automatically to all production deployments. A rigorous evaluation should document the test environment, configuration versions, assessment procedures, scoring rules, and limitations so that other researchers and practitioners can reproduce or challenge the results.

The proposed evaluation framework establishes a measurable basis for examining the practical value of threat-driven

 © Copyright 2026 GSAR Publishers All Rights Reserved

security assessment in SAP BTP applications. Its contribution lies in connecting detection performance with risk prioritization, remediation quality, and verification rather than treating vulnerability identification as the sole measure of success. A completed empirical study could determine whether the proposed process improves assessment consistency, reduces missed weaknesses, or supports more effective remediation when compared with a defined baseline. Until such measurements are available, these outcomes should remain research hypotheses rather than established findings. The next section considers the implementation implications of the framework, including its integration with secure development practices, configuration governance, and continuous application security review.

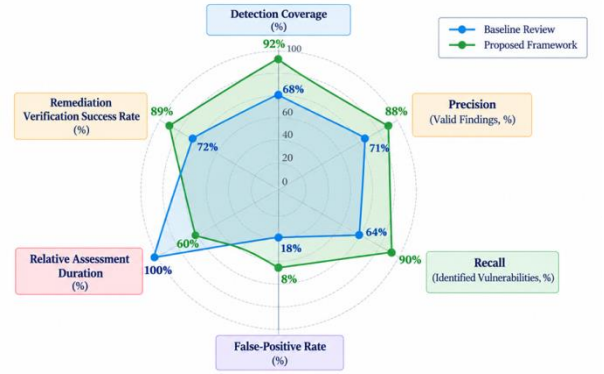


Figure 5: Comparative Vulnerability Detection Performance

7. Implementation Implications and Future Research

The practical implementation of the proposed threat-driven framework requires security assessment to become an integrated part of application development, deployment, and maintenance within SAP Business Technology Platform environments. OAuth 2.0 and XSUAA configurations influence how identities are authenticated, how permissions are assigned, and how protected resources respond to incoming requests. Consequently, security weaknesses should be examined throughout the application lifecycle rather than only during initial development or periodic audits. The framework provides a structured means of connecting application architecture, threat identification, configuration inspection, risk prioritization, and remediation verification. Its adoption can help development and security teams establish consistent assessment practices while retaining the flexibility required by different application architectures. However, its operational value depends on appropriate integration with existing development processes, reliable assessment evidence, and clearly assigned responsibilities.

Integration with the secure software development lifecycle represents a primary implementation consideration. Security reviews should begin when application requirements and authorization boundaries are defined, continue during development and testing, and be repeated before production deployment. Developers should identify required scopes, role templates, and protected operations before implementing authorization checks. Security teams can then review the application security descriptor and verify whether the configured permissions correspond to the intended business functions. Automated configuration checks may be incorporated into build and deployment workflows to identify deviations from approved security requirements. These checks should complement, rather than replace, application-level security testing and expert review. Early identification of authorization weaknesses can reduce the effort required to correct architectural problems after deployment.

Configuration governance is equally important because security settings can change as applications evolve or operational requirements are revised. A controlled process should define who may modify OAuth 2.0 client settings, XSUAA security descriptors, role assignments, and identity provider trust relationships. Changes should be documented, reviewed, and validated against established security requirements before they are introduced into production environments. Version control can support traceability for application security descriptors and related configuration artifacts, while deployment records can help establish when a security-relevant change occurred. Periodic comparisons between approved baselines and deployed configurations can identify unexpected deviations. Such governance practices help distinguish legitimate application changes from modifications that may weaken authorization boundaries or introduce unnecessary privileges.

The framework also has implications for identity and access management practices. Scope definitions, role templates, and role collections should reflect documented responsibilities rather than accumulate permissions as applications acquire new functions. Access reviews can help identify outdated assignments, excessive privileges, and role definitions that no longer correspond to operational requirements. Particular attention should be given to administrative functions and service identities because their compromise may affect multiple application components or sensitive business operations. Application teams should verify that protected endpoints enforce the intended authorization requirements, even when a request originates from an authenticated identity. These measures reinforce the principle of least privilege and reduce dependence on assumptions that a valid identity automatically represents an authorized action.

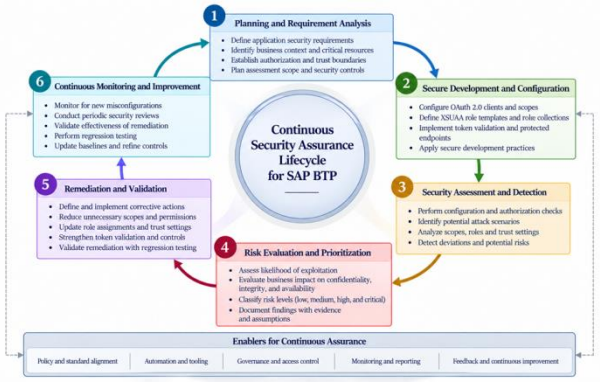


Figure 6: Continuous Security Assurance Lifecycle for SAP BTP

Operational adoption requires meaningful coordination between application developers, platform administrators,

 © Copyright 2026 GSAR Publishers All Rights Reserved

security specialists, and business owners. Developers understand the intended behavior of application endpoints, platform administrators manage relevant service and identity configurations, and security teams assess threats and evaluate the adequacy of controls. Business owners provide the context needed to determine the sensitivity of protected resources and the operational consequences of unauthorized access. Without clear responsibility for reviewing findings and implementing corrective measures, security assessment may produce reports that do not lead to effective remediation. The proposed framework therefore supports an accountable findings process in which each confirmed weakness has supporting evidence, an assigned owner, a remediation priority, and a verification requirement. This structure can improve communication and make security decisions easier to review.

Automation may improve the consistency and scalability of configuration assessment, particularly where several applications share common authorization patterns. Rule-based checks can evaluate defined security conditions, compare deployed settings with approved baselines, and flag discrepancies that require investigation. However, automated tools must account for legitimate differences in application architecture and environment-specific configuration. A rule that is appropriate for one application may generate false positives when applied indiscriminately to another. Machine learning could provide supplementary support by identifying unusual configuration patterns in historical assessment data, but its reliability would depend on data quality, representative examples, and independent validation. Such analysis should prioritize cases for investigation rather than make unsupported claims that a configuration is vulnerable. Deterministic security checks and human validation remain essential for establishing whether a reported weakness represents a genuine risk.

Several challenges may affect implementation across different SAP BTP environments. Applications may use different runtime arrangements, identity providers, authorization patterns, and integration architectures, limiting the extent to which a single set of assessment rules can be applied universally. Platform configuration options and service behavior may also change, requiring assessment procedures to be reviewed against the applicable technical documentation. Restricted access to production settings can limit the evidence available to evaluators, while incomplete records may make it difficult to reconstruct the circumstances under which a weakness was introduced. The framework should therefore document its assumptions, assessment scope, and environment-specific requirements. Its findings should be interpreted within these boundaries rather than treated as a complete assurance of security for every application within an enterprise landscape.

Future research should examine the framework through controlled experiments involving representative SAP BTP applications and predefined vulnerability scenarios. Such studies could compare the proposed approach with conventional configuration reviews using detection coverage, precision, recall, false-positive rate, assessment duration, and remediation verification success. Independent assessors could evaluate whether the risk classification process produces consistent results when given equivalent evidence. Further investigation could also examine whether repeated baseline comparisons improve the identification of configuration drift and whether different classes of weaknesses require different remediation strategies. Where machine learning-assisted analysis is introduced, researchers should evaluate it separately against rule-based detection, including its performance on previously unseen configurations. These studies would provide a stronger empirical foundation for determining the conditions under which the framework offers measurable practical benefits.

Additional research could investigate the relationship between authorization configuration and broader enterprise application risk. For example, researchers could examine how scope misconfiguration, role assignment weaknesses, token validation failures, and identity provider trust errors interact within connected application environments. Comparative studies across application types could help determine which assessment controls are broadly applicable and which depend on specific architectural assumptions. Research could also develop standardized test suites and reporting formats to improve reproducibility between organizations and independent evaluators. Such work should distinguish confirmed vulnerabilities from theoretical attack scenarios and document the conditions under which each finding was observed. A shared evaluation methodology would support more meaningful comparison between studies and reduce the risk of drawing conclusions from inconsistent testing procedures.

The implementation implications of the proposed framework extend beyond individual configuration checks to the governance of authorization throughout the application lifecycle. By connecting threat modeling, security configuration analysis, risk prioritization, and verification, the framework offers a structured basis for improving the consistency of SAP BTP security reviews. Its adoption should be supported by clear responsibilities, controlled configuration changes, appropriate automation, and evidence-based remediation decisions. At the same time, its effectiveness remains dependent on empirical validation, the completeness of the assessment scenarios, and the suitability of the controls for each application environment. Future research should establish these conditions through repeatable experiments and independent evaluation. These considerations provide the foundation for the concluding assessment of the framework's contribution to OAuth 2.0 and XSUAA security in SAP BTP applications.

Table 2: *Implementation Priorities and Validation Criteria*

Evaluation metric	Baseline security review	Proposed threat-driven framework

*Corresponding Author: **Muhammad Arslan Siddiqi**
© Copyright 2026 GSAR Publishers All Rights Reserved

Detection coverage	Depends on manual checks and review scope	Uses threat scenarios and structured configuration checks
Detection precision	Findings require individual validation	Applies defined validation criteria to reported findings
Risk prioritization	May rely on reviewer judgment	Uses documented likelihood and impact criteria
False-positive management	Depends on review procedures	Validates findings against application context
Remediation verification	Follow-up checks may vary	Repeats test cases after corrective action
Configuration consistency	Periodic configuration review	Compares deployed settings with approved baselines
Audit evidence	Depends on documentation practices	Records findings, risk decisions, and verification evidence

8. Conclusion and Future Work

OAuth 2.0 misconfigurations and XSUAA security weaknesses represent important challenges for protecting enterprise applications deployed on SAP Business Technology Platform. Although the platform provides mechanisms for identity integration, token-based authorization, and application permission management, their effectiveness depends on appropriate configuration and consistent enforcement. Weaknesses in authorization flows, scope definitions, token validation, role assignments, and identity provider trust can undermine the intended security boundaries of an application. This study has examined these concerns through a threat-driven perspective, emphasizing the relationship between individual configuration settings and the broader application security architecture. The analysis establishes that effective protection requires systematic assessment of both platform configuration and application-level authorization behavior.

The research has proposed a structured methodology for identifying and mitigating OAuth 2.0 misconfigurations and XSUAA security risks. The approach connects security baseline establishment, asset identification, threat modeling, configuration inspection, risk classification, and remediation verification within a repeatable assessment process. By examining these activities together, the methodology provides a basis for identifying weaknesses that may be overlooked when security reviews focus exclusively on isolated configuration settings. It also distinguishes potential vulnerabilities from confirmed findings by requiring supporting evidence and appropriate validation. This distinction is important for maintaining the credibility of security assessments and ensuring that remediation decisions are grounded in observable application behavior.

The vulnerability analysis has highlighted several areas that warrant particular attention during SAP BTP security reviews. Excessive scopes and inappropriate role assignments can grant permissions beyond legitimate operational requirements, while inadequate token validation may weaken the boundaries between protected resources. Incorrect identity provider trust settings and configuration drift can introduce further inconsistencies between the intended security design and the deployed application. These risks may become more significant when multiple weaknesses affect the same authorization path. Accordingly, the study emphasizes the importance of reviewing authorization flows, token properties, role collections, trust relationships, and resource-level access checks as interconnected elements of the application security model.

The proposed framework contributes a practical structure for translating identified threats into corrective security measures. Configuration checks and application-level tests support the detection process, while documented risk criteria help distinguish findings according to their potential impact and likelihood of exploitation. Remediation actions are then linked to verification procedures that establish whether the original weakness has been resolved. This relationship between detection, prioritization, correction, and retesting provides a foundation for more consistent security assessment. The framework does not assume that every configuration deviation represents an exploitable vulnerability, nor does it treat automated detection as a substitute for technical validation. Its principal value lies in organizing assessment activities into a transparent and repeatable process.

The study also identifies opportunities to incorporate automation into enterprise security operations. Rule-based checks can support the examination of security descriptors, authorization settings, role assignments, and deviations from approved configuration baselines. Machine learning may offer supplementary assistance in identifying unusual configuration patterns or prioritizing findings when sufficient historical data is available. However, such techniques require independent validation and should not replace deterministic security controls or expert assessment. The reliability of any automated capability depends on the quality of its inputs, the relevance of its detection rules, and the accuracy of its findings. These considerations reinforce the need to evaluate automation according to measurable security outcomes rather than presumed technical sophistication.

*Corresponding Author: **Muhammad Arslan Siddiqi**
© Copyright 2026 GSAR Publishers All Rights Reserved

From an industry perspective, the proposed methodology can support SAP architects, application developers, platform administrators, and security teams in integrating authorization assessment into the application lifecycle. Early configuration reviews, controlled changes, periodic access assessments, and repeatable verification procedures can help maintain alignment between business requirements and technical permissions. Clear responsibility for findings and remediation can further strengthen accountability across development and operational teams. Nevertheless, implementation must account for differences in application architecture, identity provider arrangements, and deployment requirements. The framework should therefore be adapted to the specific environment and supported by appropriate technical documentation rather than applied as an inflexible checklist.

The research has limitations that must be acknowledged. The methodology provides a structured assessment approach, but its practical effectiveness cannot be established solely through conceptual analysis. Claims concerning improved detection coverage, reduced false-positive rates, faster remediation, or stronger security outcomes require controlled experiments and documented measurements. Future studies should evaluate the framework against conventional review practices using equivalent test environments, predefined vulnerability scenarios, and consistent evaluation criteria. Independent assessment could also examine the reproducibility of risk classifications and the effectiveness of remediation verification. Broader testing across different application architectures would help determine the extent to which the methodology can be generalized to varied SAP BTP deployments.

The principal contribution of this study is the integration of threat modeling, OAuth 2.0 configuration analysis, XSUAA security assessment, risk prioritization, and remediation verification into a unified methodology for enterprise application security. This approach provides a foundation for examining authorization weaknesses in their operational context and for establishing evidence-based corrective actions. Its significance lies not in claiming that a single framework eliminates security vulnerabilities, but in offering a systematic means of identifying, evaluating, and addressing relevant weaknesses. Through disciplined implementation and empirical evaluation, the methodology may contribute to more consistent security practices and more defensible authorization decisions within SAP BTP environments.

Effective OAuth 2.0 and XSUAA security ultimately depends on the alignment of intended permissions, platform configuration, and application behavior. Security controls must be assessed continuously because application requirements, identity relationships, and deployment configurations can change over time. The threat-driven methodology presented in this study offers a structured basis for maintaining that alignment through repeatable assessment and verification. Its longer-term value will depend on practical validation, careful adaptation to individual environments, and continued refinement informed by observed results. By treating authorization security as an ongoing engineering and governance responsibility, organizations can establish a stronger foundation for protecting enterprise resources and maintaining trust in SAP BTP applications.

References

1. Mostafa, A. M., Rushdy, E., et al. (2023). An identity management scheme for cloud computing: Review, challenges, and future directions. *Journal of Intelligent & Fuzzy Systems, 45*(6). https://doi.org/10.3233/JIFS-231911
2. Piskachev, G., Becker, M., et al. (2023). Can the configuration of static analyses make resolving security vulnerabilities more effective? A user study. *Empirical Software Engineering, 28*, Article 118. https://doi.org/10.1007/s10664-023-10354-3
3. Haimed, I. B., Albahar, M., et al. (2023). Exploiting misconfiguration vulnerabilities in Microsoft's Azure Active Directory for privilege escalation attacks. *Future Internet, 15*(7), Article 226. https://doi.org/10.3390/fi15070226
4. Senanayake, J., Kalutarage, H., et al. (2023). Android source code vulnerability detection: A systematic literature review. *ACM Computing Surveys, 55*(9), Article 187. https://doi.org/10.1145/3556974
5. Otta, S. P., Panda, S., et al. (2023). A systematic survey of multi-factor authentication for cloud infrastructure. *Future Internet, 15*(4), Article 146. https://doi.org/10.3390/fi15040146
6. Philippaerts, P., Preuveneers, D., et al. (2023). Revisiting OAuth 2.0 compliance: A two-year follow-up study. In *2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)* (pp. 521–525). IEEE. https://doi.org/10.1109/EuroSPW59978.2023.00064
7. Badirova, A., Dabbaghi, S., et al. (2023). A survey on identity and access management for cross-domain dynamic users: Issues, solutions, and challenges. *IEEE Access, 11*, 61660–61679. https://doi.org/10.1109/ACCESS.2023.3279492
8. Sharif, A., Carbone, R., et al. (2022). Best current practices for OAuth/OIDC native apps: A study of their adoption in popular providers and top-ranked Android clients. *Journal of Information Security and Applications, 65*, Article 103097. https://doi.org/10.1016/j.jisa.2021.103097
9. Lomio, F., Iannone, E., et al. (2022). Just-in-time software vulnerability detection: Are we there yet? *Journal of Systems and Software, 188*, Article 111283. https://doi.org/10.1016/j.jss.2022.111283
10. Wartschinski, L., Noller, Y., et al. (2022). VUDENC: Vulnerability detection with deep learning on a natural codebase for Python. *Information and Software Technology, 144*, Article 106809. https://doi.org/10.1016/j.infsof.2021.106809

*Corresponding Author: **Muhammad Arslan Siddiqi**
© Copyright 2026 GSAR Publishers All Rights Reserved

11. Ghaffari, F., Gilani, K., et al. (2022). Identity and access management using distributed ledger technology: A survey. *International Journal of Network Management, 32*(2), Article e2180. https://doi.org/10.1002/nem.2180
12. Xu, R., Tang, Z., et al. (2022). Detecting code vulnerabilities by learning from large-scale open source repositories. *Journal of Information Security and Applications, 69*, Article 103293. https://doi.org/10.1016/j.jisa.2022.103293
13. Singh, J., & Chaudhary, N. (2022). OAuth 2.0: Architectural design augmentation for mitigation of common security vulnerabilities. *Journal of Information Security and Applications, 65*, Article 103091. https://doi.org/10.1016/j.jisa.2021.103091
14. Munonye, K., & Péter, M. (2022). Machine learning approach to vulnerability detection in OAuth 2.0 authentication and authorization flow. *International Journal of Information Security, 21*, 223–237. https://doi.org/10.1007/s10207-021-00551-w
15. Arshad, E., Benolli, M., et al. (2022). Practical attacks on login CSRF in OAuth. *Computers & Security, 121*, Article 102859. https://doi.org/10.1016/j.cose.2022.102859
16. Mohammad, A. (2022). Distributed authentication and authorization models in cloud computing systems: A literature review. *Journal of Cybersecurity and Privacy, 2*(1), 107–123. https://doi.org/10.3390/jcp2010008
17. Xiao, Y., et al. (2022). SoK: Context and risk aware access control for zero trust systems. *Security and Communication Networks, 2022*, Article 7026779. https://doi.org/10.1155/2022/7026779
18. Parkinson, S., & Khana, S. (2022). Identifying high-risk over-entitlement in access control policies using fuzzy logic. *Cybersecurity, 5*, Article 6. https://doi.org/10.1186/s42400-022-00112-1
19. Rennhard, M., Kushnir, M., et al. (2022). Automating the detection of access control vulnerabilities in web applications. *SN Computer Science, 3*, Article 376. https://doi.org/10.1007/s42979-022-01271-1
20. Khan, A., Ahmad, A., et al. (2022). Authorization schemes for internet of things: Requirements, weaknesses, future challenges and trends. *Complex & Intelligent Systems, 8*, 3919–3941. https://doi.org/10.1007/s40747-022-00765-y
21. van Ede, T., Khasuntsev, N., et al. (2022). Detecting anomalous misconfigurations in AWS identity and access management policies. In *Proceedings of the 2022 Cloud Computing Security Workshop* (pp. 63–74). ACM. https://doi.org/10.1145/3560810.3564264
22. Gupta, C., Singh, R. K., et al. (2022). An approach for verification of secure access control using security pattern. *Wireless Communications and Mobile Computing, 2022*, Article 1657627. https://doi.org/10.1155/2022/1657627
23. Cho, D. X., Son, V. N., et al. (2022). Automatically detect software security vulnerabilities based on natural language processing techniques and machine learning algorithms. *Journal of ICT Research and Applications, 16*(1), 70–87. https://doi.org/10.5614/itbj.ict.res.appl.2022.16.1.5
24. Hanif, H., Md Nasir, M. H. N., et al. (2021). The rise of software vulnerability: Taxonomy of software vulnerabilities detection and machine learning approaches. *Journal of Network and Computer Applications, 179*, Article 103009. https://doi.org/10.1016/j.jnca.2021.103009
25. Cho, D. X. (2023). A new approach to software vulnerability detection based on CPG analysis. *Cogent Engineering*. https://doi.org/10.1080/23311916.2023.2221962
26. Bassi, D., & Singh, H. (2023). A systematic literature review on software vulnerability prediction models. *IEEE Access, 11*, 110289–110311. https://doi.org/10.1109/ACCESS.2023.3312613
27. Heiding, F., Katsikeas, S., et al. (2023). Research communities in cyber security vulnerability assessments: A comprehensive literature review. *Computer Science Review, 48*, Article 100551. https://doi.org/10.1016/j.cosrev.2023.100551
28. Kalouptsoglou, I., Siavvas, M., et al. (2023). Software vulnerability prediction: A systematic mapping study. *Information and Software Technology, 164*, Article 107303. https://doi.org/10.1016/j.infsof.2023.107303
29. Li, X., Moreschini, S., et al. (2023). The anatomy of a vulnerability database: A systematic mapping study. *Journal of Systems and Software, 201*, Article 111679. https://doi.org/10.1016/j.jss.2023.111679
30. Abdul, A. M., Mohammad, A. A. K., et al. (2022). Enhancing security of mobile cloud computing by trust- and role-based access control. *Scientific Programming, 2022*, Article 9995023. https://doi.org/10.1155/2022/9995023

 © Copyright 2026 GSAR Publishers All Rights Reserved