IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 22 June 2023, accepted 8 August 2023, date of publication 10 August 2023, date of current version 18 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3304270

RESEARCH ARTICLE

System for Cross-Domain Identity Management (SCIM): Survey and Enhancement With RBAC

THOMAS BAUMER[1], MATHIS MÜLLER[2], AND GÜNTHER PERNUL[2], (Member, IEEE)
[1]Nexis GmbH, 93053 Regensburg, Germany
[2]Chair of Information Systems, Faculty of Informatics and Data Science, University of Regensburg, 93053 Regensburg, Germany
Corresponding author: Thomas Baumer (Thomas.Baumer@nexis-secure.com)

ABSTRACT System for Cross-domain Identity Management (SCIM) is a schema and protocol to exchange identity data across cloud-based applications utilizing a Representational State Transfer (REST) Application Programming Interface (API). Since it quickly gained decent vendor adoption, it is considered a relevant industry standard for Identity Management (IdM) and related systems. The Request for Comments (RFC) of SCIM primarily focuses on identity data but has opening points for Role-Based Access Control (RBAC). E.g., sets for roles and entitlements are specified for a user entity. However, the RFC family does not detail RBAC further, which leads to some proliferation and anomalies. E.g., the role and entitlement sets for the user are implemented in ''freestyle'' notations by vendors, and information on orphan roles or entitlements is not accessible. Moreover, some vendors and recent extensions add role and entitlement (and some other) endpoints leading to vendor-specific dialects for SCIM, which hampers simplicity and interoperability. This work contributes by proposing a RBAC profile for SCIM utilizing Design Science Research Methodology (DSRM). We thus look at present knowledge about API design, Access Control Models (ACMs), IdM and its APIs. Furthermore, we conduct a literature review on SCIM, including its specification documents, scientific contribution, and vendor implementations. An artifact combines this knowledge and improves SCIM with a RBAC profile. An open-source Swagger prototype showcases the API design. Finally, design principles formulate essential insights to guide future RBAC REST APIs.

INDEX TERMS API, design principles, IAM, IdM, RBAC, REST, SCIM.

I. INTRODUCTION

Effective creation and utilization of standards might restrict individual freedom but facilitates a safe and sound integration when multiple technologies are involved. Well-known authorities for the standardization of technologies are the National Institute of Standards and Technology (NIST), the Organization for the Advancement of Structured Information Standards (OASIS) or the Internet Engineering Task Force (IETF). These organizations' standards have strongly influenced technologies utilized nowadays.

System for Cross-domain Identity Management (SCIM) is an IETF standard for exchanging Identity Management (IdM) data. The Request for Comments (RFC) family (RFC 7642-7644) [1], [2], [3] details on this. The specifications include a protocol and resources for users, groups, and an extension for enterprise users. Utilizing a Representational State Transfer (REST) Application Programming Interface (API), SCIM targets interoperability, security, and scalability while emphasizing simplicity. The data model of SCIM derives from present protocols, like Portable Contacts [4], vCards [5], or Lightweight Directory Access Protocol (LDAP) [6]. However, it also allows extensions for custom resources and attributes. Regarding its scope, SCIM enables application provisioning and deprovisioning in an independent context from authentication. This separation allows for more flexible, automated, and comprehensive provisioning, especially for centralized and federated IdM models. SCIM thus solves issues of authentication mixed up with just-in-time provisioning, as done by Security Assertion Markup Language (SAML) [7] or OpenID Connect [8]. The identity lifecycle highlights the benefits of

The associate editor coordinating the review of this manuscript and approving it for publication was Jonathan Rodriguez.

independence. On the one hand, an application can process a joining identity before its first authentication, allowing for early interactions. On the other hand, as an identity is moving or leaving an application, SCIM can enforce updates or deprovisioning. Although its relatively recent release in 2015, SCIM has already gained decent vendor adoption [9]. Surprisingly, SCIM is barely scientifically discussed but is a hot topic for standardization efforts, see Section III-B [10].

Although SCIM focuses on identity data like users and groups, it has rudiment preparations for exposing Role-Based Access Control (RBAC) [11] data. Thus, a user resource has one collection for roles and one for entitlements. However, the RFC does neither specify a convention for these collections nor independent endpoints for these resources. This ambiguity leaves some room for interpretation by implementors and recent extensions [12]. Section III gives an overview of the different approaches. These limitations fail some premises of SCIM, like emphasizing simplicity or interoperability. Additionally, RBAC features like Mutually Exclusive Roles (MERs) are not covered.

Application scenarios and technical merits for incorporating RBAC in SCIM are evident. Because its already present vendor-adoptions SCIM prove its feasibility, especially in terms of scalability, performance, and practicability. Building upon SCIM thus utilizes present approaches and experiences, easing a real-world impact of the extension. The same also applies to RBAC. In terms of application scenarios, combining a comprehensive REST API design like SCIM and a comprehensive RBAC model allows for a broad variety of use cases. Especially interesting for the latest advances in access control might be add-on architectures [13], which rely on excellent connectivity. A universally adopted exchange protocol and data scheme for RBAC in practice allows rapid integration of scientific approaches like for data quality [13], access reviews [14], or role mining and engineering [15], [16], [17].

This work contributes by utilizing a Design Science Research Methodology (DSRM), according to [18], [19], and [20]. The primary contributions comprise:

- An overview of SCIM including a literature review on SCIM, a summary of its specification documents, scientific contributions and vendor implementations (see Table 4).
- An integration of the SCIM and RBAC data models, leading to an open-source general-purpose RBAC REST API (see Figures 6 and 7).
- A derivation of design principles guiding future RBAC REST APIs (see Figure 9).

The following Sections II and III focus on a knowledge base for API design, RBAC, and SCIM. Afterward, the DSRM is introduced in Section IV to describe, implement and evaluate a prototype for a RBAC SCIM API in the Sections V and VI. Section VII describes the lessons learned by working with the prototype and states design principles to guide present and future IdM data exchange systems utilizing RBAC. Section VIII concludes.

II. RELATED WORK AND BACKGROUND

This Section introduces relevant work for IdM APIs to collect a knowledge base in the sense of Hevner et al. [20]. Relevant topics encompass a sound API design (see Section II-A), Identity Management (IdM) (see Section II-B), Access Control Models (ACMs) (see Section II-C), and APIs for IdM (see Section II-D).

A. APPLICATION PROGRAMMING INTERFACES

APIs are a cornerstone for modern software engineering and encompass software libraries, frameworks, and web services based on REST or Simple Object Access Protocol (SOAP). It is no surprise that millions of APIs are thus available on Maven, PiYI, or npm. The rise of interconnected systems also makes APIs based on web services a common foundation of modern software engineering. Since SCIM is a REST API, we will introduce the basics of general API design and specifics for REST APIs in this Section [21].

Research interest in API design and its evolution covers maintenance, usability, documentation, performance, security, testing, verification, and integration [21]. Achieving perfection is challenging and rare, which is also evident in API design. Creators and consumers of APIs thus need to have in mind that APIs require maintenance. On the one hand, missing API updates might render it outdated or even vulnerable over time. On the other hand, frequent breaking updates impede code maintenance for API consumers. API designers thus need to act with foresight when creating or updating their APIs to keep breaking changes or ambiguities at a minimum [21]. Furthermore, from a usability perspective, multiple parties use the API. An API thus should facilitate its usage with good documentation, like providing efficient access to relevant content, giving an initial entry point for the API, or supporting different learning and development strategies [22].

In terms of APIs security and performance based on web services, APIs require protection from unauthorized access by enforcing proper authentication, authorization, geofencing, fuzzing protection, and avoidance of plain text passwords, debugging modes, or auto-incrementing identifiers. APIs also utilize timestamps and throttling to protect themselves from being overwhelmed by frequent and large requests. Given these conditions, strict input checks on the transmitted data protect the API. From a functional perspective, APIs must also be comprehensively tested and verified to ensure well-known and potential use cases. A proper API also provides tools for insights, including audit logs or best practice response codes (like [23] for REST) [24].

Since SCIM is a REST API, this work recites core principles for this API type in greater detail. These core principles encompass resource addressability, resource representations, uniform interface, statelessness, and hypermedia [25]. (i) Resource addressability denotes that resources of REST APIs are uniquely addressable utilizing Hypertext Transfer Protocol (HTTP) concepts like Uniform Resource

Identifiers (URIs) [26]. (ii) A resource is considered as a representation. It means that resources are independent of the implementing parties' internal entities. Common notations like JavaScript Object Notation (JSON) specified in RFC8259 [27] or eXtensible Markup Language (XML) specified in RFC3076 [28] often serialize resources. (iii) An uniform interface utilizes semantically valid HTTP methods. It includes using POST requests for creating resources, GET for reading, PUT or PATCH for updating, and DELETE for deleting. Additionally, HTTP response codes express a semantically valid outcome of the request, like returning the status 201 on creating resources alongside the successfully created resource. Following common considerations of RFC9110 [23] thus allows for uniform access and manipulation of resources. (iv) REST APIs do not store a shared state between multiple requests. This statelessness thus allows for independent requests when manipulating data. (v) The hypermedia concept denotes navigation and resource discovery by providing links to further related resources [25].

Finally, Swagger (https://swagger.io/) is a powerful tool for documenting, presenting, describing, and generating REST API code for popular programming languages, like Java, Python, JavaScript, or C#. This work uses Swagger to specify the API design and to precisely and conveniently communicate the API. Developers use Swagger to generate large parts of their REST API code.

B. IDENTITY MANAGEMENT: BASICS AND DISTINCTIONS

Despite some controversies between computer science and other disciplines like philosophy, an identity describes a human being. From a philosophical perspective developed via a centuries-long and nuanced discussion, this includes the entirety of attributes of a person, either perceived by the person itself or by others. From the computer science perspective, however, a (digital) identity only needs to satisfy identification: Therefore, an identity defines as a set of attributes to identify a subject within any set of subjects. Any further identity attributes are optional but yet often also present. For the remainder of this work, we thus refer to a (digital) identity interchangeably with the terms user and subject. Also, the term partial identity refers to representations of a (digital) identity for a specific context, like an account within an application [29].

The term Identity Management (IdM) refers to managing (digital) identities and controlling their access to resources. The IdM hence targets controls for identities in the domain the IdM instantiates. An IdM system provides technologies to realize IdM, like a repository, an authentication provider, access control, auditing, provisioning, or Single Sign On (SSO). (i) The repository stores logical data about the managed identities and the applied Access Control Policy (ACP). (ii) The authentication provider performs the authentication of an identity. (iii) Access control for modeling policies and assigning them to identities. Section II-C details further on access control. (iv) Auditing for tracking any updates for entities within the repository. This tracking enables forensic analysis for IdM incidents and is required by several compliance regulations, like Sarbanes-Oxley Act (SOX) [30]. (v) Provisioning refers to automating all parts of the identity lifecycle, like identifier creation, linkage to authentication providers, configuring identity attributes or policy assignments, propagation of repository data to connected systems (Service Providers (SPs)), or identity deprovisioning. (vi) SSO provides a single centralized authentication interface to access several managed applications [29], [31], [32].

The general idea for operating an IdM system is a user sending an authentication request for accessing resources of a Service Provider (SP). The SP checks whether the authentication request can be granted by forwarding this authentication request to the Identity Provider (IdP). The IdP stores and maintains identity data (e.g., the identities themselves, their credentials, and further attributes) and Access Control Policies (ACPs) to determine access grants. Regarding the decision of the IdP, the SP grants or denies the access request. The deployment of IdM distinguishes roughly into three models: the isolated, centralized, and federated model [33]. Figures 3, 2, and 3 depict the models side-by-side.

(i) For the isolated IdM model a SP fulfills its own role as SP and the role of the IdP at the same time. The SP thus needs to store its own identity data and ACP to decide the authentication requests. Figure 3 depicts the isolated IdM model. Multiple SP are also possible for this IdM model. While this approach allows for a simplistic and stand-alone SP deployment, managing identities for multiple SP without a centralized IdP quickly becomes cumbersome. For example, users must remember many passwords, or administrators struggle to keep the identities of the SPs current.

(ii) Centralized IdM models solve issues of isolated IdM models by establishing a centralized IdP for a given domain. The SPs no longer need to provide IdP functionalities by themselves and can forward authentication requests to the centralized IdP. The IdP then sends back an authentication response, on which basis the SP grants or denies access.[1] Figure 2 depicts a centralized IdM model, which is oftentimes applied to organizations like enterprises. This IdM model allows for centralized management and SSO technologies, which ease administration and boost user experience. However, the centralized IdP imposes a single point of failure, privacy issues, and low support on cross-domain access.

(iii) In contrast, the federated IdM model also integrates other domains, creating virtually a unique global domain. The federation hence equals a federated trust domain binding together SPs by agreements, standards, and technologies.

[1] In addition to Cao and Yang [33], we want to point out that some SPs still need to cache relevant IdP data for themselves, despite the centralized model. Some advanced SP provide complex and customizable services, requiring personalization and more fine-grained ACPs. E.g., some SAP systems can fine-grain customization down to single buttons for specific ACPs of users. Without caching, every UI reload might trigger several requests for the IdP, causing performance bottlenecks. Furthermore, SP might add further SP-specific attributes to the identities without the need for synchronization to the IdP.

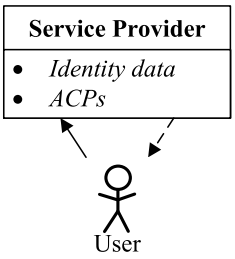

FIGURE 1. Isolated IdM.

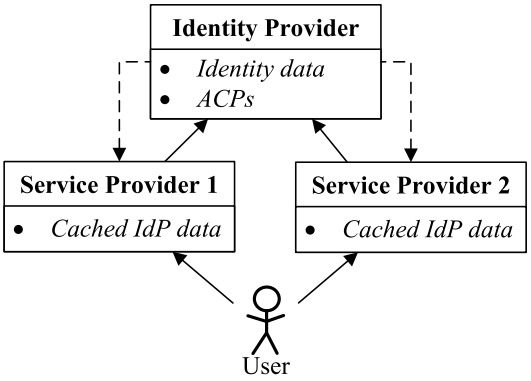

FIGURE 2. Centralized IdM.

In this IdM model, SPs can maintain their individual repositories, and authentication requests can use credentials from trusted domains. Thus, a user can utilize SSO technologies for a single domain and the trusted domains integrated by the federation. Figure 3 depicts the federated IdM model. Challenges of federated IdM models are privacy, attribute consistency, provisioning, or trust between the integrated domains.

C. ACCESS CONTROL: BASICS AND MODELS

The basic idea for access control follows a simple operational process: A user (subject) sends an access request to a reference monitor (also called an access control mechanism), deciding to grant or deny access to an object based on the available Access Control Policies (ACPs). Besides this operational process, access control requires an administrative process to determine the ACPs. Admins thus create and maintain the ACPs the reference monitor utilizes. Thereby, ACPs follow the syntax and notation of their respective Access Control Model (ACM) understood by the reference monitor. Common and widely acknowledged examples for ACMs are Discretionary Access Control (DAC), Mandatory Access Control (MAC), RBAC, and Attribute-Based Access Control (ABAC) [34], [35], [36]. The semantics for the ACPs root in the organizations' security policies and its security model: while security policies define high-level rules for access control, the security model formally collects and represents security policies. The admins thus combine the security model (semantics) with the ACM (syntax) to create a (machine-processable) ACP set for the reference monitor. Figure 4 depicts these essential building blocks for

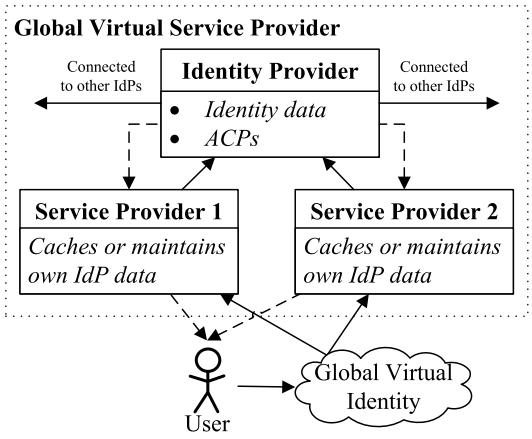

FIGURE 3. Federated IdM.

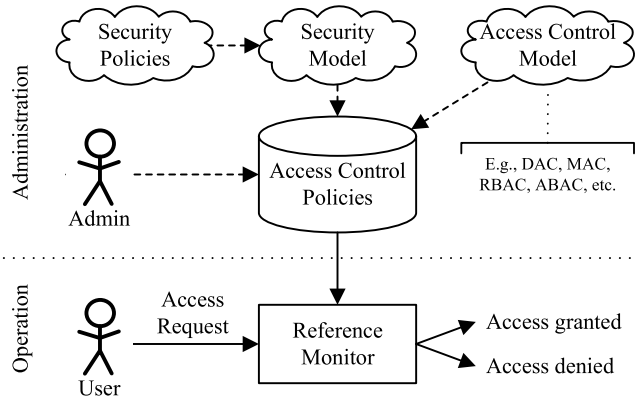

FIGURE 4. Basic building blocks for access control.

access control. A dotted line shows the distinction between the operative and administrative parts [34].

The most common Access Control Models (ACMs) encompass DAC, MAC, RBAC, and ABAC [4], [34], [35], [36], which we detail on in the following paragraphs. Besides these common ACM, latest research also covers Norm-Based [37], Activity-Centric [38], Rule-Based [39] or Relationship-Based Access Control [40], [41], [42], [43].

Discretionary Access Control (DAC) [44], [45] is the most straightforward approach for access control. The reference monitor grants (or denies) access based on an explicit triple (S, O, A), whereas S represents a set of subjects desiring access, O represents objects accessed by the subjects, A represents the allowed actions for a S and O pair (e.g., rwx for read, write, and execute). An access matrix model depicts the (S, O, A) triples, for which S forms its rows and O its columns while the entry $A[s, o]$ provides the allowed actions. E.g., if a given subject s desires access with the action a to an object o which is in $A[s, o]$, the access is granted [34].

Mandatory Access Control (MAC) uses access classifications for subjects and objects to realize access control. Access classes have two parts: a security level and a category set. For example, the security levels Top Secret (TS) > Secret (S) > Confidential (C) > Unclassified (U) define a dominance

over each other, whereas a subject with the TS clearance can access all objects of its category and one with a C-level clearance only objects of the C and U classes. The category set refines this model by separating domains from each other, as subjects with a TS clearance for an engineering category might not access other TS-level objects of another category (e.g., the back office). In this sense, two properties must be fulfilled for confidentiality (Bell-LaPadula Model [46]): (i) no-read-up of a lower security level to read (confidential) information of a higher one, and (ii) no-write-down for a higher security level to write (confidential) information in a lower one. For integrity, it is the other way around (Biba Model [47]): (i) no-read-down as objects written by subjects with a lower clearance might lack integrity, implying that a subject can only read objects from the same or higher clearance, and (ii) no-write-up as a subject with a lower clearance cannot produce objects with higher level integrity but can write objects for its or lower security levels. MAC is especially beneficial for controlling the information flow within an organization [34].

Role-Based Access Control (RBAC) [11], [35] introduces roles R with semantic meaning, as an intermediate between users U and permissions P.[2] The roles' R many-to-many assignments to P and U are denoted $PA \subseteq P \times R$ for permissions and $UA \subseteq U \times R$ for users. Users activate the roles via sessions S, which grants them access following the assigned permissions via PA. Furthermore, the RBAC family also introduces Hierarchical and Constrained RBAC. For Hierarchical RBAC, the assignment $RH \subseteq R \times R$ denotes a role hierarchy, allowing for the inheritance of roles (and ultimately permissions). For Constrained RBAC, constraints might restrict every aspect of RBAC. For example, a role might allow only certain permissions, and a Separation of Duty (SoD) might define prohibited combinations of specific roles. SoDs are enforced either statically for modifying the model itself or dynamically for adding a role to a session. RBAC is the most common ACM for enterprises [48], [49]. For example, in an enterprise context, an employee's job position reflects this semantic meaning. RBAC decreases the total number of assignments as roles bundle direct user to permission assignments to some degree. Various approaches like role mining, role engineering or hybrids generate role sets reflecting an organization's security policies [15], [17]. Overall, it facilitates more simplicity.

Attribute-Based Access Control (ABAC) [36] takes advantage of the subject and object attributes. Therefore, policies represent rules defining attribute conditions that a subject, an object, or the environment must fulfill to grant or deny access. ABAC utilizes multiple logical components (points) for realizing the concept. Thus a subject requests access to an object by a Policy Enforcement Point (PEP) that handles incoming requests and enforces the access decisions made by the Policy Decision Point (PDP). The primary task of the PDP is to query the relevant policies from the repository and check them against the subject, object, and environmental attributes provided by the Policy Information Point (PIP). Finally, administrators create and maintain policies via the Policy Administration Point (PAP). ABAC's primary benefits are flexibility and dynamics, as policies can use any desired and dynamically resolved attributes. Its flexibility goes so far that ABAC can emulate RBAC. This emulation, however, comes with drawbacks in simplicity and maturity, making RBAC still the most adopted ACM for enterprises [49], [50], [51].

[2]Permissions P are further formalized as operations OPS on objects OBS usually perceived as one unit [35].

D. IDENTITY MANAGEMENT STANDARDS AND APIS

Identity Management APIs target an efficient and standardized data exchange for the concepts presented in the previous Sections. These APIs include (non-exhaustive) LDAP [6], vCards [5], Portable Contacts [4], SAML [7], Service Provisioning Markup Language (SPML) [52], OAuth [53], eXtensible Access Control Markup Language (XACML) [54], [55], or SCIM. The standards frequently exchange similar entities [56].

Lightweight Directory Access Protocol (LDAP) [6] standardizes means to query and maintain directory services. These directory services provide information about managed network resources, like users, groups, devices, Etc. Along further attributes, LDAP Data Interchange Format (LDIF) represents LDAP entries by featuring a Distinguished Name (DN) as a unique identifier, a Common Name (CN) as a display name, Organizational Units (OUs) as organizational units, or Domain Component (DC) as part of the entry domain. An example, for a DN is ''dn: cn = John Doe, ou = security, dc = myorg, dc = org''. Besides the data exchange, users authenticate via LDAP's bind operation.

vCard [5] specifies a representation of contact information, like names, addresses, emails, telephone numbers, Etc. Its primary use case is to standardize the exchange of electronic business cards, like email or smartphone contacts. For example, a vCard can be attached to an email message by a file with the registered .vcf or .vcard filename extensions or use the Multipurpose Internet Mail Extension (MIME) type text/vcard.

Portable Contacts [4] builds upon contact information standards, like vCard [5]. The design unifies contact information and improves data retrieval via authorization, discovery, query, and response functionalities. Therefore, the RFC targets managing a user's address book across his distributed services comprehensively. Concepts of Portable Contacts strongly influence the design of SCIM.

System for Cross-domain Identity Management (SCIM) [1], [2], [3] is the focus of this study. Its RFC7642 [1] provides a general overview and concepts of SCIM, while RFC7643 [2] and RFC7644 [3] introduce its data schema and protocol. The primary scenario for SCIM is identity provisioning for centralized and federated IdM models. This focus on provisioning means that an IdP executes Create Read

Update Delete (CRUD) operations for its managed identities to synchronize with its SPs. The data exchange happens separately from authentication. It prepares its SPs for future authentication requests or, e.g., updates identity attributes not required by authentication, like an added telephone number. LDAP, vCard, and Portable Contacts influence SCIM's schema, while its protocol designs a HTTP REST API with similarities to Portable Contacts. Section III provides a deep dive into SCIM.

Service Provisioning Markup Language (SPML) [52] is a similar standard to SCIM and covers the provisioning of users and services. Via SPML, SPs receive information on the active users and their authorizations in a different context to authentication. The SPML core functions cover CRUD operations, while additional features include async, batch, password, or search capabilities. Due to a consensus of vendors on some concerns regarding SPML [57], vendors moved on by redesigning a provisioning standard, leading to the SCIM standard.

Security Assertion Markup Language (SAML) [7] specifies a standardized data exchange for security assertions for authentication or authorization. Its primary use case is Web Browser SSO: When a user attempts to request a resource of a SP via its web browser (user agent), the SP calls for a SSO service at the IdP. If the user is not yet authenticated, the IdP prompts the user for its credentials. With a valid authentication, the IdP responds with a SAML assertion. Finally, the user agent uses the SAML assertion to retrieve the protected resource of the SP. Since the SPs trust the IdP, the users only require a Single Sign On (SSO), avoiding unnecessary credentials and their prompts.

Open Authorization (OAuth) 2.0 [53] enables users to authorize third-party access without sharing credentials. For example, a user can authorize a third-party service to access its master data, like its birthday or name. The third-party service thus asks the user for an authorization grant (containing the user's credentials), which the authorization server converts to an access token. The third-party service can use the access token to access the authorized data until its expiration. Possessing an OAuth 2.0 access token is sometimes confused with an indirect authentication, although it is just authorization. To mitigate this confusion, OpenID Connect (OIDC) 1.0 [8] adds an identity layer on top of OAuth 2.0 to authenticate the user for authorization workflows. OIDC (and SAML) do not provide comprehensive provisioning, and SPs rather ''cache'' their received identity data as accounts. SP thus might operate with outdated identity data: While outdated contact data needs inconvenient repeated maintenance for each SP, accounts may become outdated themselves when an identity leaves the organization. These outdated accounts (also called orphan accounts) also impose a security risk as these might still provide access to sensitive resources.

eXtensible Access Control Markup Language (XACML) [54] specifies an architecture and a language for ABAC policies. XACML thus realizes processing and administration for ABAC policies by utilizing functional points. Upon a user's access request, the PEP receives the request and forwards it to the PDP. The PDP then decides the access request using the policies from the PAP and the subject, object and environment attributes provided by the access request or PIP. Finally, the PEP receives the response for further fulfillment. Besides ABAC, XACML theoretically can emulate RBAC [55], but some requirements like SoDs, understandability, delegation, scalability, or auditability remain unclear [51] in comparison with a more mature RBAC [48].

In summary, SCIM is an evolution of multiple standards in the field. SCIM supersedes SPML, while it builds upon standards for identity data like vCards, Portable Contacts, and LDAP. Because of SCIM's focus on provisioning, it complements seamlessly with authentication standards like SAML or OIDC as a provider for synchronized identity data. SCIM has some loose preparations for permission and role provisioning, like done for policies with XACML, but does not excel. However, complex applications like SAP or Active Directory can provide their own permission and role structures in advanced domains like large enterprises, requiring effective RBAC provisioning. This work's API design tackles the provisioning for RBAC entities.

III. SCIM: STANDARDIZATION, LITERATURE & PRACTICE

In addition to Section II with its analysis of related work and background, this Section takes a deep dive into System for Cross-domain Identity Management (SCIM) itself, further adding to a knowledge base in the sense of Hevner et al. [20]. We thus execute a structured literature review described in Section III-A. The following Sections present the result of the literature review divided into specification documents (see Section III-B), scientific literature (see Section III-C), and practice-relevant vendor-driven implementations (see Section III-D). Finally, Section III-E summarizes present work on SCIM and points out open research gaps.

A. METHOD FOR THE LITERATURE REVIEW

This Section collects relevant literature for SCIM and its extensions. This review includes specification documents, scientific literature, and practice-relevant vendor implementations to get a comprehensive SCIM overview. We apply a three-stage process to obtain these sources. Table 1 depicts an overview of the utilized search terms and results for the specification documents and scientific literature. The following enumeration summarizes the three-stage search process:

1) Search for SCIM specifications at IETF.
2) Search for literature at scientific libraries.
   a) Search at scientific libraries.
   b) Refine results with Google Scholar.
3) Search for practice-relevant implementations.

In the 1) stage, we queried SCIM specifications (of version 2.0) directly at the IETF by searching for ''scim''.[3]

[3] https://datatracker.ietf.org/doc/search?name=scim&rfcs=on&activedrafts=on&olddrafts=on

We include the RFCs and active & expired Internet-Drafts (also known as extensions) and did not consider superseded Internet-Drafts.

In the 2a) stage, we queried the high-quality online libraries IEEE Xplore,[4] AISeL,[5] ACM,[6] ScienceDirect,[7] and SpringerLink.[8] Table 1 depicts the utilized input fields, search terms, and raw & relevant results. We only included peer-reviewed contributions after the publication date of the SCIM RFCs. Results with poor quality, like an unclear contribution, were excluded.

In the 2b) stage, we refined the yielded results with Google Scholar.[9] We used the same search term as in the 2a) stage and executed a forward search based on the RFCs 7642, 7643, and 7644. We also applied the same quality considerations as in the 2a) stage to include only high-quality publications. Table 1 depicts the input fields, search terms, and raw & relevant results. Please note that we only marked not previously found results as relevant during this refinement stage to avoid duplicates. While the regular search at Google Scholar still refined the literature catalog by 14 results, a forward search using the "cited by" functionality based on the RFCs yielded no further publications. Therefore, we assume all relevant scientific publications for SCIM were found.

In the 3) stage, we searched for practice-relevant and vendor-driven SCIM implementations. While no official registry for SCIM implementations exists, a community effort, fortunately, at least loosely maintains an overview of 70 implementations of SCIM [9]. To ensure practice-relevancy, we map these implementations to successful vendors on the market, which utilize and improve the SCIM API. Therefore, we consider vendors with at least 50 ratings from large enterprises (with a company size of more than 10B USD) at Gartner[10] and a documented SCIM server implementation with further improvements besides standard SCIM. This leads to two relevant vendor implementations: SailPoint[11] and Oracle.[12]

As summarized in Table 1, this brief literature review on SCIM yielded 64 relevant results. These results include 28 (3 RFCs and 25 Internet Drafts) specification documents at the IETF. Furthermore, we identify 34 scientific publications. IEEE is the leading publisher for scientific contributions on SCIM. Finally, we also consider 2 practice-relevant vendor implementations of SCIM of SailPoint and Oracle.

For the specification documents and scientific literature, we tracked the publication year and contribution type regarding SCIM. The contribution type distinguishes publications

[4]https://ieeexplore.ieee.org/search/advanced
[5]https://aisel.aisnet.org/do/search/
[6]https://dl.acm.org/
[7]https://www.sciencedirect.com/search
[8]https://link.springer.com/search
[9]https://scholar.google.de/
[10]https://www.gartner.com/reviews/market/identity-governance-administration
[11]https://developer.sailpoint.com/iiq/api/identityiq-scim-rest-api/
[12]https://docs.oracle.com/cd/E52734_01/oim/OMDEV/ scim.htm

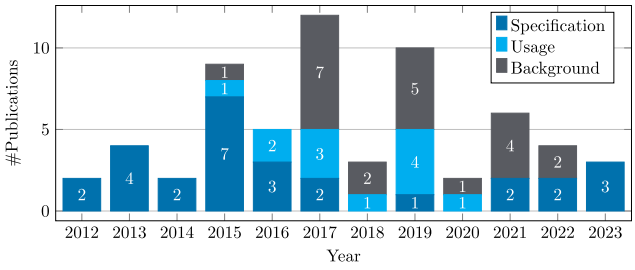

FIGURE 5. Publication over time for the specification documents and scientific literature on SCIM. A continuous flow of publications connected to SCIM with spikes in 2015, 2017, and 2019 is present.

with SCIM background references (Σ22 results), publications actively using SCIM (Σ12 results), and specification documents creating or improving SCIM (Σ28 results). Figure 5 depicts the publications' contribution type over time. Starting in 2012, the IETF community published specification documents, leading to a continuous flow of publications ever since. The plot visualizes spikes in 2015, 2017, and 2019.

B. SCIM IN STANDARDIZATION

The IETF is the driving standardization authority for SCIM. Besides its RFCs family [1], [2], [3], Internet Drafts for SCIM (also known as extensions) are present. This Section details these specification documents.

The SCIM RFC family includes three documents: RFC7642 [1] introduces SCIM and gives an overview of use cases and concepts. The specification shows the intended provisioning usage of SCIM to move identities in, out, and around centralized or federated IdM models. RFC7643 [2] defines the SCIM data model covering the resources (entities) *User* and *Group*. A User (digital identity) represents a human being and has single-value attributes like userName or nickName or multi-value attributes like emails, phoneNumbers, and addresses. SCIM additionally specifies multi-value attributes of a User for assigned groups, entitlements (permissions), and roles. The Group resource bundles a single-value attribute displayName and a multi-value attribute members, which either contains references to Users or to recursively nested Groups. Other SCIM resources are the *ServiceProviderConfig*, *ResourceType*, and *Schema*. The ServiceProviderConfig determines SCIM settings, the ResourceType for retrieving all available resource types like User, Group, Etc., and the Schema provides information on each resource's available (custom) attributes. Finally, RFC7644 [3] specifies the sophisticated SCIM protocol, including REST methods, HTTP response codes, filtering, and pagination.

At the IETF, the SCIM community published 25 extensions (Internet Drafts). For this work, we traced their primary contribution. As a first step, we categorized the extensions based on their contribution to the respective RFCs as use case (RFC7642), schema (RFC7643), or protocol (RFC7644). With this initial rough categorization, we derived more

TABLE 1. Search terms for the literature review on SCIM in three stages: 1) specification documents at the IETF are the foundation for SCIM. Therefore, we include the IETF document types RFCs, active Internet-Drafts, and expired Internet-Drafts. 2a) scientific literature from high-quality online libraries on SCIM has to be in English, published after the SCIM RFCs since 2015, and must have undergone a peer-review process. 2b) Google Scholar refines the results from stage 2a) if these were not yet found by stage 2a). 3) We also include vendor-driven SCIM implementations with high practice-relevancy.

	Library	Input Field	Search Term	Raw	Rel.
1	IETF	RFCs, Internet-Drafts	scim	30	28
2a	IEEE Xplore	Full Text & Metadata	(("Full Text & Metadata":"System for Cross-domain Identity Management"))	10	10
	AISeL	All Fields	"System for Cross-domain Identity Management"	0	0
	ACM	All	"System for Cross-domain Identity Management"	2	2
	ScienceDirect	Art. with these terms	"System for Cross-domain Identity Management"	6	6
	Springer Link	with the exact phrase	System for Cross-domain Identity Management	3	2
2b	Google Scholar		"System for Cross-domain Identity Management"	115	14
	—	RFC7642	cited by functionality	6	0
	—	RFC7643	cited by functionality	11	0
	—	RFC7644	cited by functionality	11	0
3	scim.cloud [9]	Table on website	SCIM 2.0 Implementations	70	2
			Σ	264	64

sophisticated categories: use case, discovery, mapping, constraint, event, pagination, and search. (i) Use case extension discuss SCIM applications on a higher level. (ii) Discovery extensions are tools to discover SCIM functionalities. (iii) Mapping extensions define mappings or integration of SCIM to other formats. (iv) Constraint extensions restrict SCIM attributes or assignments. (v) Event extensions cover communicating SCIM events between IdPs or SPs, like added or changed identities. (vi) Pagination extensions refer to breaking large requests into smaller ones to enhance the performance and resilience of the API. (vii) Search extensions cover advanced search and filter approaches. Table 2 maps the SCIM extensions with these categories. The following paragraphs detail on advances of each category.

(i) The use case category primary highlights SCIM application scenarios. Li [62] and Hunt et al. [63] mainly cover general SCIM provisioning use cases to move identities in, out, and around centralized and federated IdM models. Hunt and Ansari [74] present pushing and polling scenarios for SCIM events between SCIM servers and clients to notify each other about provisioning events (like CRUD operations on resources).

(ii) The discovery category primarily covers one extension [71] that describes a best-practice approach for discovering SCIM services. It utilizes the ''/.well-known'' mechanism, which supports WebFinger queries to discover SCIM services.

(iii) The mapping category refers to mapping or integrating SCIM with other specifications or models. Hunt et al. [60] closer integrate OAuth 2.0 with SCIM by adding a Client resource to automatically assign an OAuth client id and client secret via SCIM to an on-boarded application using OAuth. Hunt and Wilson [68] propose a password management extension with schemas and endpoints supporting the identity lifecycle. The proposal includes schemas for password policies, password reset & validation requests, and username generation, validation & recovery requests. Greevenbosch and Sun [64] map the vCard standard with SCIM. Shahzad et al. [81] propose schema extensions that allow the provisioning of a wide array of (Internet of Things (IoT)) devices. Hunt [61] specifies a mapping of SCIM to LDAP to enable directory services via SCIM. Grizzle et al. [73] define schemes for integrating with Privileged Access Management (PAM) software. Finally, Zollner [12] designs an extension for role and entitlement (permission) resources, including cardinalities and hierarchies of these resources. This extension is the most relevant for this work but still has several drawbacks: The roles and entitlements attributes of the user resource are still unspecified, the assignment between roles and entitlements are not covered, and advanced RBAC features like SoDs and sessions are not included.

(iv) The constraints category includes restrictions for interacting with SCIM resources. Ansari and Hunt [66] restrict the deletion of SCIM resources by distinguishing between a soft and hard deletion. Zollner [77] specifies an extension for verified domains, e.g., to constraint domains for email addresses. Zollner [12] also defines cardinalities within his roles and entitlement extension as a constraint type. Finally, the referential value location extension of Zollner [78] specifies accepted values tied to other resources. E.g., a user's manager needs to refer to a valid and present user identifier.

(v) The events category collects extensions covering means to push or poll events raised by SCIM. Hunt and Ansari [74] specify essential use cases for pushing and polling events. As the earliest proposal on SCIM events, Wahl [65] adds an extension allowing SCIM IdPs to notify

TABLE 2. Internet Draft specification documents which extend SCIM. This overview tracks the extension types by the categories use case, discovery, mapping, constraint, event, pagination, and search. The RFCs themselves are not depicted.

SCIM extension	Year	Use Case	Discov.	Mapping	Constraint	Event	Pagination	Search
Hunt et al. [58]	2012						✓	✓
Hunt et al. [59]	2012						✓	✓
Hunt et al. [60]	2013			✓				
Hunt [61]	2013			✓				
Li [62]	2013	✓						
Hunt et al. [63]	2013	✓						
Greevenbosch and Sun [64]	2014			✓				
Wahl [65]	2014					✓		
Ansari and Hunt [66]	2015				✓			
Hunt et al. [67]	2015					✓		
Hunt and Wilson [68]	2015			✓				
Hunt [69]	2015							✓
Hunt et al. [70]	2016					✓		
Hunt [71]	2016		✓					
McMurtry [72]	2016					✓		✓
Grizzle et al. [73]	2017			✓				
Hunt and Ansari [74]	2017	✓				✓		
Wahl [75]	2019					✓		
Hunt [76]	2021						✓	✓
Zollner [77]	2021				✓			
Zollner [12]	2022			✓	✓			
Zollner [78]	2022				✓			
Peterson and Zollner [79]	2023						✓	
Hunt and Cam-Winget [80]	2023					✓		
Shahzad et al. [81]	2023			✓				
Σ	25	3	1	7	4	7	4	5

its SP about changed resources. Later, Wahl [75] also specifies the other way around, allowing the SP to notify the IdP about changes. McMurtry [72] takes this idea further and enables tracing changes with watermarks to poll them periodically. Hunt et al. [67] propose subscription, event, and feed schemes and endpoints. They also define basic events improved by subsequent proposals of Hunt et al. [70] and Hunt and Cam-Winget [80]. The latest definition of events [80] distinguishes between feed events (add and remove), provisioning events (create, patch, put, delete, activate, and deactivate), signal events (authMethod and pwdReset), and miscellaneous events (asyncResp).

(vi) The pagination category refers to extensions improving a mechanism to break down large requests into several smaller ones to enhance the performance and resilience of the SCIM API. Hunt et al. propose two competing approaches to improve pagination in combination with querying the SCIM API. One extension [58] covers paginated search requests by utilizing a token that refers to an overarching search of paginated requests. The other extension [59] requires an additional endpoint /Searches allowing for creating a search request (like a database view) and querying this search request. Hunt [76] also extends pagination and filtering to multi-value attributes. This extension allows filtering and paging through assignments of SCIM resources, which is especially useful for resources with many assignments. Finally, Peterson and Zollner [79] specify cursor-based pagination as an alternative since SCIM only covers index-based pagination.

(vii) The search category includes extensions that allow sophisticated search requests. Hunt et al. propose two competing search methods (token-based [58] vs. an endpoint to store searches [59]). Hunt also takes part in an IETF discussion about adding a new HTTP method SEARCH (instead of modeling it with GET or POST), which would simplify and streamline sophisticated SCIM queries [69]. The event polling extension of [72] also allows advanced search for changed resources. Finally, the multi-value filtering extension of Hunt [76] enables searching for assignment attributes.

The active community around SCIM at IETF highlights the practice-relevance of SCIM. The extensions target real-world issues of deployed software strengthening the standard. The extensions, however, also imply that the current specification of SCIM is not yet optimal, requiring ongoing improvement efforts like this work.

TABLE 3. Results of the literature review on approaches actively using SCIM. We trace the categories architecture, application domain, and use case.

Paper	Year	Use Case	Application Domain	Architecture
Thanh et al. [101]	2015	Provisioning	eHealth	FI-STAR
Ertl et al. [102]	2016	Identity Harmonization	eScience	INDIGO
Nakandala et al. [103]	2016	Provisioning	eScience	Discussion
Bernabe et al. [104]	2017	CP-ABE	IoT	FIWARE
Ceccanti et al. [105]	2017	Identity Harmonization	eScience	INDIGO
Innerbichler et al. [106]	2017	Provisioning	IoT	NIMBLE
Hernández-Ramos et al. [107]	2018	CP-ABE	IoT	FIWARE
Corici et al. [108]	2019	Provisioning	IoT / 5G	Open5GMTC
Santos et al. [109]	2019	Provisioning	IoT / 5G	5G4IoT
Santos et al. [110]	2019	Provisioning	IoT / 5G	5G4IoT
Selvanathan et al. [111]	2019	Provisioning	Manufacturing	eFactory
Corici et al. [112]	2020	Provisioning	IoT / 5G	Open5GMTC
Σ12				

C. SCIM IN LITERATURE

The literature review revealed 34 scientific publications on SCIM with sufficient quality. We skimmed these publications and categorized their relationship to SCIM as background reference or active SCIM usage. In summary, 22 publications[13] refer to SCIM in a background reference, and 12 contributions actively use SCIM. Since the background references provide only limited insights for SCIM, we focus on the contributions actively using SCIM in this Section. Table 3 summarizes these contributions by tracing their general SCIM use case, application domain, and utilized architecture. The following paragraphs detail on these categories.

(i) Use cases for SCIM distinguish between provisioning (8 papers), identity harmonization (2 papers), and Ciphertext-Policy Attribute-Based Encryption (CP-ABE) (2 papers). Provisioning is thereby an essential IdM process allowing for synchronizing centralized identity data of an IdP with its SPs. Identity harmonization builds upon provisioning by linking heterogeneous identities of SPs or federated IdPs. Thereby, it obtains an aggregated identity with harmonized attributes which is looped back to the other IdPs and SPs. Since provisioning (and identity harmonization) are the primary use cases for SCIM, scientific contributions mostly use it the intended way. Another 2 approaches utilze SCIM to solely obtain identity attributes for CP-ABE which is especially useful for privacy-preserving encryption based on attribute claims.

(ii) Application domains for SCIM cover eHealth (1 paper), eScience (3 papers), IoT (7 papers), 5G (4 papers), and manufacturing (1 paper). For the eHealth domain, SCIM serves as an eHealth attribute provider and provisioning engine. The eScience contributions focus on collaborative science infrastructures requiring IdM systems. These systems require provisioning and identity harmonization, because of the heterogeneous SP systems in use. For IoT domains, two approaches use SCIM as an attribute provider for CP-ABE. Further approaches use SCIM as an enabler to provision managed devices with a notable overlap with 5G (or cellular networks in general) and manufacturing. Despite the contributions' distinct application domains, SCIM serves as a backbone for general-purpose provisioning API. Domain-specific and general IdM systems can utilize SCIM provisioning.

(iii) Architectures using SCIM are shaped by their application domain. The FI-STAR architecture is driven by a research project for eHealth implementing IdM functionalities. In this context, a notable contribution of Thanh et al. [101] is a suggestion for eHealth attribute mapping for SCIM users. Two contributions for the INDIGO-Datacloud use SCIM identity harmonization approaches for scientific computing. The eScience contribution of Nakandala et al. [103] also discusses common eScience architectures. The FIWARE architecture of the IoT domain is used for showcasing CP-ABE. Another IoT architecture is NIMBLE as the predecessor of eFactory of the manufacturing domain. Both use SCIM for provisioning use cases. For the 5G domains, four contributions utilize SCIM with two architectures (Open5GMTC and 5G4IoT). For Open5GMTC the Home Subscriber Server (HSS) was extended with SCIM to provision users, while 5G4IoT connects the HSS to a standalone IdM system implementing SCIM. Both Open5GMTC and 5G4IoT, however, extend the SCIM user attributes to include domain-specific attributes, like International Mobile Subscriber Identitys (IMSIs) and International Mobile Equipment Identitys (IMEIs).

D. SCIM IN PRACTICE

While writing this paper, 70 implementations, including well-known vendors like Omada, Oracle, SailPoint, Salesforce, or Saviynt, are present [9]. The amount of implementations is probably even higher due to unreported usages, which

[13] Background references: [31], [49], [56], [82], [83], [84], [85], [86], [87], [88], [89], [90], [91], [92], [93], [94], [95], [96], [97], [98], [99], [100].

TABLE 4. Comparison of SCIM APIs (C: SCIM core, O: Oracle, S: SailPoint, Z: Zollner [12]).

	Endpoint	GET	POST	PUT	PATCH	DELETE
SCIM Core	/Users	COS	COS	COS	CO	COS
	/Groups	CO	CO	CO	CO	CO
	/Me (User)	CO	CO	CO	CO	C
	/ServiceProviderConfig	COSZ				
	/ResourceTypes	COSZ				
	/Schemas	COSZ				
	/Bulk		C			
	[prefix]/.search		CO			
Oracle API	/Organizations	O	O	O	O	O
	/UserNameGenerator		O			
	/UserNameRecoverer		O			
	/UserNameValidator		O			
	/PasswordPolicies	O	O	O	O	O
	/PasswordResetterWithChallenges		O			
	/PasswordValidator		O			
	/NotificationTemplates	O	O	O	O	O
	/SystemProperties	O			O	
SailPoint API	/Accounts	S	S	S		S
	/Alerts	S	S			
	/Applications	S				
	/CheckedPolicyViolations		S			
	/Entitlements	SZ				
	/LaunchedWorkflows	S	S			
	/ObjectConfig	S				
	/PolicyViolations	S				
	/Roles	SZ				
	/TaskResults	S				
	/Workflows	S				

leverages SCIM to a vital industry standard despite its comparatively recent publication in 2015. In the following, we look closely at two leading vendors for SCIM with advances for extending SCIM. This highlights the proliferation faced in practice working with SCIM. Table 4 thus depicts resources, endpoints, and HTTP methods designed by the SCIM core (C), Oracle (O), and SailPoint (S). Because of its relevance, this work also includes the Roles and Entitlements extension by Zollner [12] (Z).

The overview given in Table 4 reflects SCIM: it reveals areas of expansion based on practical needs, theoretical shortcomings, and design strengths and flaws. First, none of the considered APIs comprehensively implements the SCIM core specification. While the APIs cover CRUD operations on the user resource (besides some ambiguity on PUT and PATCH), the support lacks groups as a core resource. Also, utility endpoints like retrieving the requester's data (Me), bulk, or search operations are missing or not implemented as specified. One can identify two major expansion streams based on the extended endpoints of Oracle and SailPoint. On the one hand, Oracle primarily extends SCIM to provide tools supporting the identity lifecycle. On the other hand, SailPoint primarily provides additional endpoints to disclose information about their policies which supports the policy lifecycle. For roles and entitlement resources, SailPoint and Zollner [12] propose extensions, which differ in their attributes and structure. E.g., SailPoint further distinguishes between business and IT roles. Surprisingly, none of the APIs provides a way to inspect or manage role-entitlement relationships. Our analysis finds further ambiguities for groups, organizations, and roles. The SCIM core API defines the group resource, which allows both Users and nested Groups as members to enable bundling of Users and enhanced access control. As mentioned earlier, present APIs do not support groups, which also blur with the organization and role resources defined by Oracle and SailPoint. While Oracle's organizations model organizational structures like departments in enterprises with further relevant attributes and no roles or entitlements assigned, SailPoint's roles can express with a type attribute to be an organization. Another aspect are SoDs which SailPoint only covers. While SailPoint provides no direct representation of SoDs, simulations or checks for violations regarding

granting access are available. Finally, note that this analysis of SCIM only scratches surface level. On the one hand, it covers reported SCIM implementations with advanced towards extending SCIM, while unreported SCIM APIs might be available. On the other hand, the attributes of the presented resources differ fundamentally, which renders a detailed analysis out of scope for this work. Despite these limitations, this section puts across a sound foundation, challenges, and potentials of SCIM.

E. RESEARCH GAPS AND POSITIONING OF THIS WORK

Section III took a deep dive into SCIM by conducting a structured literature review based on SCIM specification documents, scientific literature utilizing SCIM, and practice-relevant vendor SCIM implementations. We identified several research gaps and opportunities to improve SCIM.

One research opportunity thus comprises SCIM extensions for specific application domains. Extensions for attributes managed by SCIM or even new resources might be suitable or required. Thanh et al. [101] already suggest a mapping for patient data and approaches for the 5G domain suggest attributes like IMSIs or IMEIs. The SCIM specification defines an enterprise user extension for an enterprise domain. Shahzad et al. [81] also suggest an extension for devices, which applies for the IoT domain. Rigorous research can identify relevant resources and attributes for these and further application domains.

Another research opportunity is consolidating the current specification extensions and vendor-driven implementations. As discussed for the vendor implementations of SCIM, unconsolidated extensions lead to incompatible approaches modeling the same concepts.

Finally, two expansion streams for SCIM based on vendors and extensions can be identified: For example, Oracle expands SCIM with functionalities for the identity lifecycle, while SailPoint focuses on a policy lifecycle. A careful analysis of both lifecycles can identify further improvements for SCIM.

Based on this SCIM analysis and these research opportunities, this work contributes by consolidating RBAC features. We derive an integrated SCIM RBAC model based on the standardized NIST RBAC model [35], the SCIM roles and entitlements extension of Zollner [12], and SCIM vendor implementations. By designing a general-purpose RBAC REST API, provisioning use cases based on a policy lifecycle are covered.

IV. RESEARCH METHODOLOGY

This work follows a Design Science Research Methodology (DSRM) like proposed by Hevner et al. [20]. The approach requires an analysis of both the environment and the knowledge base (see Sections II and III) to design a problem-solving artifact (see Section V), which loops back insights to environment and knowledge base.

As Gregory and Hevner [18] point out with their *design science research knowledge contribution framework*, nothing is really novel because everything builds upon something else. The terms problem and solution maturity distinguish the situation more precisely. E.g., a sign of high solution maturity is the presence of adopted standards or guidelines. Therefore, we have already a high solution maturity for RBAC [35], REST API design [22], [23], [25], or SCIM itself (regarding to exchanging identities) [1], [2], [3]. These standards try to solve well-understood problem domains for access control, data, and identity exchange, indicating a high problem maturity for each domain individually. This work is thus not re-inventing RBAC, REST, or SCIM. Instead, the goal of this work is to tackle the low solution maturity of their *combination* (as discussed in Sections II and III) while keeping the benefits of each standard. In terms of Gregory and Hevner [18], this work classifies into two contribution types regarding the taken perspective. On the one hand, from a SCIM perspective, combining it with RBAC is an *exaptation* because a well-known solution (RBAC) is adapted for a new problem (the half-hearted specification of RFC7643 [2] towards RBAC). On the other hand, from the perspective of the currently weak advances of SCIM towards RBAC (see Section III), this work classifies into an *improvement* because present approaches especially lack a *valid* RBAC integration although RBAC's maturity.

Following this positioning based on Gregory and Hevner [18], we derive a valid combination of the RBAC and SCIM data models in Section V-A while considering the needs of present SCIM APIs and extensions. This combined RBAC SCIM data model is applied to SCIM in Section V-B. Afterward, Section V-C presents the design for interacting with the combined and applied data model. Section V-D complements the artifact design by showcasing an open-source and prototypical implementation utilizing Swagger. The evaluation in Section VI demonstrates the artifact's validity, utility, quality, and efficacy. Finally, Section VII discusses insights on the iterative development, design principles [19] for future RBAC REST APIs, and limitations.

V. DESIGNING A RBAC PROFILE FOR SCIM

This section covers the design of an artifact for a RBAC profile for SCIM. First, it includes deriving a RBAC data model based on the knowledge base. A REST API based on SCIM wraps this data model. Finally, we implement the resulting RBAC profile for SCIM with Swagger and make it open-source on GitHub.[14]

A. DERIVATION OF A RBAC MODEL FOR SCIM

The data model is the foundation of a RBAC profile for SCIM. This section derives this data model and depicts it in Figure 6. Please note that this work uses the following terms preferred by the SCIM community without further notice (we do not discuss the differences between the terms). These terms include the usage of the term *user* for (digital) identity

[14] https://rbac4scim.github.io/api, https://github.com/ThomasBaumer/RBAC_4_SCIM (long-term link)

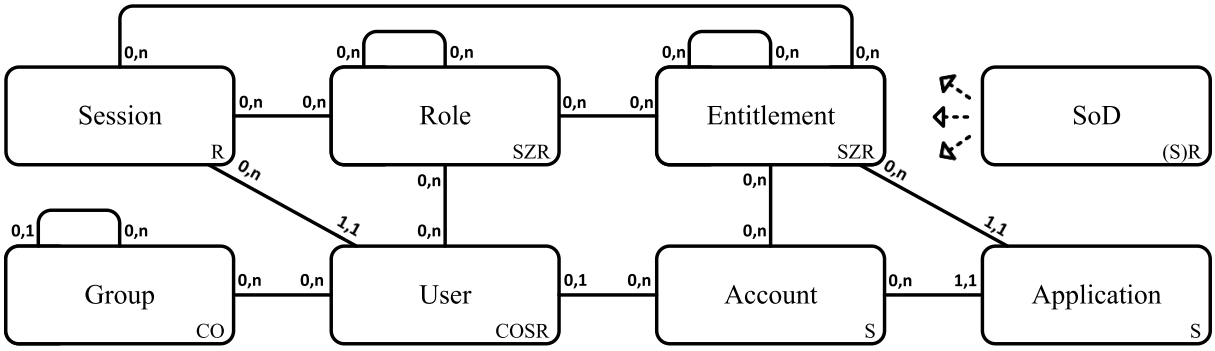

FIGURE 6. A derived data model for a RBAC profile for SCIM. The numbers on the arcs denote cardinalities as min and max. The abbreviations hint the grounding for the resource (C: SCIM core, O: Oracle, S: SailPoint, Z: Zollner [12], R: RBAC [11], [35]).

or employee, *entitlement* for permission, or *application* for target system or SP.

The SCIM core data schema [2] defines *user* and *group* entities. Groups hierarchically bundle users and nested groups to model structures like teams or departments. A group may have one parent group while several users and nested groups are assigned. Additionally, a user has a many-to-many relationship with groups. SCIM also suggests *role* and *entitlement* entities as (seemingly) many-to-many relationships with users. The standard, however, does not elaborate on the design of these relationships or entities, which is a source for serve inconsistencies of implemented SCIM APIs seen in Section III. As a recent extension, Zollner (Microsoft) [12] proposes schemata for roles and entitlements covering RBAC cardinalities and many-to-many hierarchies for roles and entitlements. The proposed data model takes these into account. The user to role, user to entitlement, and role to entitlement relationships are still not specified.

From a theoretical point of view [11], [14], [35], [56], SCIM is thus incomplete in achieving a RBAC data model. Building on the role and entitlement extension, we add a many-to-many relationship for roles and entitlements to cover RBAC. Furthermore, entitlements and accounts are application-specific, which implies some considerations. (i) The derived data model also should consider an *application* entity. (ii) An *account* represents a user for a specific application. A user thus relates to accounts in an optional one-to-many relationship. In terms of RBAC, a user activates roles, which are broken down into several application-specific accounts and entitlements. (iii) An application-specific entitlement is unique for the application. E.g., an application duplicates its entitlements if the same second application also runs on other stages (e.g., within development and production environments). (iv) Even within an application, there might be scopes making an entitlement *type* attribute reasonable [2]. The data model thus should incorporate an application entity and one-to-many relationships to entitlements and accounts. Adding a many-to-many account to entitlement relationship also allows for directly granted access, which is in line with the present design of common IdM APIs [14], [56].

Constraints are an advanced concept of RBAC to model cardinalities or *SoDs*. Cardinalities are already sufficiently covered by the extension of Zollner [12]. SoD coverage of present SCIM APIs is rare because only SailPoint offers utility endpoints to check SoD violations. A SoD is a restriction in which roles and entitlements are mutually exclusive and thus covers many-to-many relationships for the excluded entities. From a practical point of view, SoDs need to provide many-to-many allowlists which exceptionally let selected groups, users, or accounts stay unaffected by a SoD. Moreover, restrictions of SoD might block static model changes or a dynamic activation of exclusive roles or entitlements during a session requiring a type attribute (static or dynamic). Dynamic SoDs thus imply the presence of *sessions* of Core RBAC [11], [35]. Sessions are (de-)activated based on the user's required roles and entitlements, which also requires a check for dynamic SoD violations. Consequently, a session should cover a many-to-many relationship for activated roles and entitlements and an one-to-many relationship with the user, which activates its entitlements by his accounts.

Additional to the proposed data model in Figure 6 for a SCIM RBAC profile, the data model uses utility entities of SCIM like ServiceProviderConfig, Schema, or ResourceType without structural changes. This data model encompasses all relevant entities of the proposed NIST standard [35]. Section VI-A of the evaluation elaborates further on the validity of the data model.

B. REPRESENT RBAC ENTITIES AS REST RESOURCES

REST APIs utilize representations of entities, usually called *resources*, for an adequate exposure. This resource representation [25] highlights the difference between the application data model and represented API data model. While the application data model stays inside an application and includes application-specific and internal attributes, the represented data model is exchanged between applications and provides

simplicity and flexibility for custom attributes. We thus refer to represented entities as *resources* and relationships as *assignments*. This section elaborates on details for both of them for the proposed resources in Figure 6.

In general and in close alignment with the SCIM core specification [2], a resource has three essential attributes: an *id* in Universally Unique Identifier (UUID) format [113], an *externalId*, and a complex attribute called *meta*. The id uniquely identifies a resource across the whole set of resources managed by the SCIM service provider. The externalId is for the client's convenience to query its own (application-specific) resources. The meta attribute holds metadata about the resource, including the *resourceType* of the resource (e.g., User, Group, Etc.), a URI *location* of the resource, its *version*, and two timestamps *created* and *lastModified*. Additional to the SCIM core, we add a *type* attribute as the analysis in Section III revealed a common usage of types to categorize entities further. E.g., SailPoint adds a type attribute to roles to indicate a business or IT role, the SCIM core suggests the attribute for entitlements, and Zollner [12] also suggests the attribute for roles and entitlements. Please note, that the resourceType attribute distinguishes resources themselves from each other while the type attribute allows for sub-types of resources. The specific resources exposed by the REST API inherit all of the attributes from the general resource to ensure consistent behavior. Additionally, assignments utilize the general resource definition to ensure resource addressability and support the hypermedia concept of REST APIs.

Because of their semantic meaning, some resources need additional attributes and considerations discussed in this section. For the User and Group resources, using SCIM core attributes minimizes breaking changes. Especially for the User resource, this includes a wide range of name attributes and contact information [2]. The Role and Entitlement resources cover attributes to specify cardinalities, including *limitedAssignmentsPermitted* (not aware of hierarchical assignments), *totalAssignmentsPermitted* (aware of hierarchical assignments), and *totalAssignmentsUsed*. Moreover, various APIs like SailPoint, the SCIM core [2], or the extension by Zollner [12] suggest the *type* attribute to categorize resources further. For SoD, the type should denote whether its enforcement happens during the access modeling or when sessions are activated. For Groups, we suggest using the type to categorize an organizational meaning. Additionally, the ServiceProviderConfig should specify the utilization of RBAC features. These features include hierarchies of roles and entitlements or constraints based on cardinalities, static and dynamic SoDs [35].

C. PROTOCOL DESIGN

Accessing the resources of the data model is a crucial part of the REST API, including HTTP methods, status codes, and authorization. Please note, that the proposed API aims for general purpose: therefore, we do not impose a specific use case or application scenario. The API thus needs a general description of CRUD operations for all considered resources depicted in Figure 6.

For all HTTP methods (esp. GET, POST, PUT, PATCH, and DELETE) and resources (depicted in Figure 6), the REST API design ensures a typical RESTful behavior. This behavior aligns with HTTP semantics [23] and the SCIM protocol [3]. The HTTP response codes 200, 201, 204, 307, 308, 400, 401, 403, 404, 409, 412, 413, 500, and 501 are used in accordance to their semantic meaning [23]. Moreover, 400 returns specific API errors with the attribute *scimType*, which describes the error type in greater detail. E.g., we suggest returning *sodViolation* as scimType for violated SoDs. For requests altering the state of the SCIM service provider, non-writable attributes are omitted, asserted, or replaced with defaults. Additionally for specifying assignments, only the *id* attribute of the assignee needs to be provided by the request. [3]

For creating a resource, we specify a POST request for every resource, which returns for a successful operation the status code 201 and the created resource with its id set by the SCIM service provider.

For reading resources, SCIM [3] already provides a general POST search request. Additionally for each resource, SCIM specifies a specific POST search request, a GET endpoint to retrieve a known resource by its *id*, and another GET endpoint to execute a filtered query. The requests can configure pagination, sorting, and included attributes of the demanded resources and return the status code 200 on success.

For updating a resource, SCIM [3] defines a replace operation (PUT) and a partial update (PATCH). For PUT, a full representation of the updated resource is expected. On success, the status code 200 returns. For PATCH, SCIM specifies *add*, *remove*, and *replace* operations by using a path and the new value for the altered attribute. This behavior enables specific modifications without knowing the full representation of the resource. On success, the status code 200 returns. On errors with the path, the response will indicate the path error. [3]

For deleting a resource, SCIM [3] specifies DELETE endpoints with a path parameter for *id* attribute. On success, the API returns the HTTP status code 204 without a response body since the resource is no longer available. Upon querying the deleted resource with GET, the API must return the HTTP status code 404. A deleted resource does not cause the status code 409 during create operations [3].

Finally, for every CRUD operation of the API resources a specific authorization enables fine-grained control over the REST API. In this sense, both PUT and PATCH HTTP methods are accessible by an update authorization. Regardless of a filter, GET requests are available using a read authorization for the given entity. Finally, POST requests are secured by a create authorization (except the search request) and DELETE requests by one for deleting a resource. Failures in performing authentication result in the status code 401, and missing permissions result in the status code 403. For the

API prototype, we suggest the usage of OAuth2 [53], but any suitable mechanism for authorization might replace it.

D. PROTOTYPE IMPLEMENTATION (OPEN-SOURCE)

With the presented REST API design for SCIM, we developed a Swagger API prototype open-source available on GitHub.[15]

→ https://rbac4scim.github.io/api

It documents the API and allows for live try-out interactions. Figure 7 showcases a collection of role HTTP REST methods. The API details each HTTP REST method and its utilized resource. On the one hand, this includes the payload resources, like Users, Groups, Roles, Entitlements, Accounts, Applications, Sessions, and SoDs. On the other hand, utility or configuration resources, like ResourceType, Schemas, ServiceProviderConfig, and Bulk, are also available.

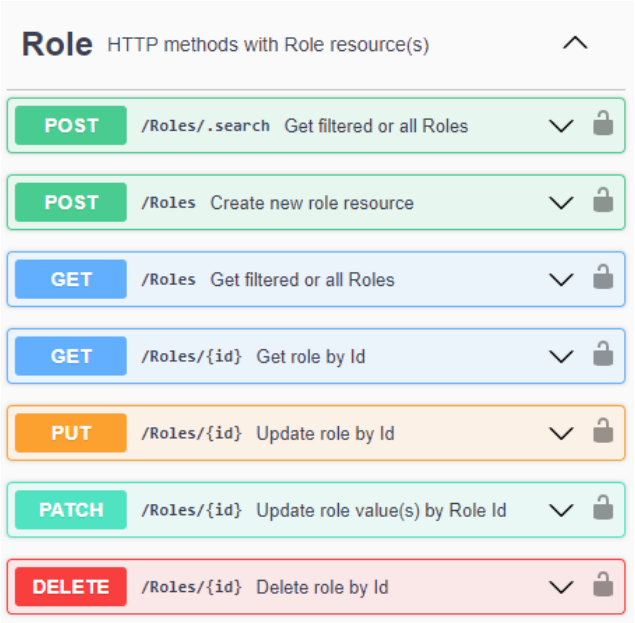

FIGURE 7. Role excerpt of the Swagger documentation for the SCIM REST API. See live on https://rbac4scim.github.io/api.

Furthermore, powerful open-source Swagger tools[16] are capable of automated code generation for a broad range of common client and server programming languages and libraries. On the one hand, this includes Java, Kotlin, Python, ASP.NET, Node.js, or Scala for servers. On the other hand, available client code generation encompasses C#, Dart, Go, HTML, Java, JavaScript, Kotlin, PHP, Python, R, Ruby, Scala, Swift, or TypeScript. Implementers of the proposed RBAC SCIM API only need to map their own CRUD methods for their data model with the specified HTTP methods and the data model shown in Figure 6 to use the API.

[15] https://github.com/ThomasBaumer/RBAC_4_SCIM (long-term link)
[16] https://editor.swagger.io/

VI. EVALUATION

Gregor and Hevner [18] suggest evaluating an artifact developed with a DSRM by its validity, utility, quality, and efficacy. The following sections consider these.

A. VALIDITY BASED ON RBAC FUNCTIONALITIES

The first evaluation aspect for the SCIM RBAC API is its validity, utilizing the proposed NIST standard for RBAC [35]. RBAC functionalities thus include administrative functions, supporting system functions, and review functions. Like in Section V, this section uses terms the SCIM community prefers. Furthermore, we omit special functions for operations and objects since applications usually manage these internally and represent them as entitlements.

Administrative functions encompass creating and maintaining RBAC resources. For simplicity, we focus on functions named by the proposed NIST standard for RBAC [35]. For Core RBAC, these include creating and deleting users and roles. Entitlements are perceived as predefined by the application and thus need no specific function except discovery. Core RBAC requires adding and removing functionalities for role to entitlement and role to user relationships. Hierarchical RBAC requires adding or deleting inheritance relationships and adding ascendants or descendants. For Constrained RBAC creating and deleting static and dynamic SoDs or altering their cardinalities are required. Moreover, functions for adding and removing role assignments for SoDs are needed. In summary, the designed RBAC REST API for SCIM fully covers administrative functions for Core, Hierarchical, and Constrained RBAC. An administrator can manage resources by utilizing the proposed HTTP methods POST, PUT, PATCH, and DELETE [35].

Supporting system functions assist RBAC. For Core RBAC, these include creating a session resource, adding or removing role to session relationships, and checking access. While Ferraiolo et al. [35] make some remarks on details for Hierarchical and Constrained RBAC, these functions remain the same. The RBAC SCIM API covers all of these functions. It is possible to create sessions via POST, alter its assignments via PUT or PATCH, and check access by querying the user session via GET [35].

Review functions let an administrator query the current resources. For Core RBAC, the role to user assignment needs a query function based on the respective user and role. The API also requires query functions to retrieve a user's entitlements and roles or active roles and entitlements of a session. For Hierarchical RBAC, additional functions for querying role to user assignments must also resolve their hierarchies. For Constrained RBAC, functions for querying static and dynamic SoDs, their assigned roles, and cardinalities are mandatory. All the required functions are present by utilizing GET HTTP methods, including their filters. However, some of the functions for Hierarchical RBAC might need multiple requests to resolve recursion [35].

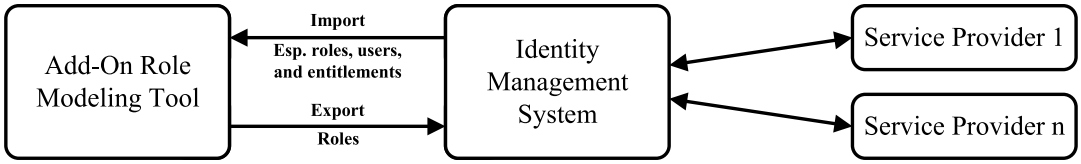

FIGURE 8. Simplified overview for the considered use case evaluation.

In summary, all functions suggested by the RBAC standard [35] are considered for the RBAC SCIM API. The proposed RBAC SCIM API is thus a valid API to cover RBAC.

B. UTILITY BASED ON A ROLE MODELING USE CASE

The design of the proposed RBAC SCIM API has no specific use case in mind to establish a general purpose RBAC REST API. This is especially beneficial since it allows multiple use cases based on the actual organizational needs. For a utility evaluation, however, we arbitrarily pick a role modeling use case similar to connecting an external PAP. Other possible use cases are application provisioning or add-on systems [13] for data quality, access reviews, or SoD firewalls. Figure 8 illustrates a simplified overview for this use case.

Let's consider an external, hybrid role modeling tool [13], [15], which is capable of role mining and role engineering. This role modeling tool thus needs to know the managed users, present roles, constraints like SoDs, and the application-specific accounts and entitlements. Each of the named resources has custom attributes, which are special for the organization applying the RBAC SCIM API. This data is available at the IdM system, which implements the proposed RBAC SCIM API on the server side. The add-on role modeling tool thus can query resources of the API from the client side. Imports and exports are triggered by the role modeling tool, which synchronizes the models comprehensively with nightly jobs. This case serves as a simplistic example for connecting said role modeling tool with an IdM system. More advanced cases, like incremental synchronizations, are also feasible. The following paragraphs consider the setup of the system, importing data from the IdM system, and returning generated roles to the IdM system.

On setup of this system, the custom attributes of each resource need consideration. SCIM allows for configuring custom attributes for each resource (or even custom resources). For example, configurations of these attributes include information about their type, mutability, uniqueness, or required and return specification. The Schema endpoint of the SCIM API returns the specification of each attribute and resource. These attribute configurations are the basis for using attribute restrictions and filtering. Let's assume for the use case a custom role attribute "factory" with the allowed values "A", "B", or "C", which enhances the semantic meaning for the role. The attribute is mutable, not-required and returned on request.

After the setup phase, the role modeling tool can import data from the IdM system. This import includes present users, roles, entitlements, accounts, applications, or SoD constraints alongside their assignments. The RBAC SCIM API allows reading these resources by a search request via POST or GET. These requests are paginated to prevent cumbersome or heavy requests for the API. Additionally, filters and attribute restrictions allow for specifying the required resources and their attributes even further. For example, requests may be restricted to only active users. The role modeling tool can thus transfer a comprehensive data RBAC data model with the means provided by the proposed API.

```
GET https://rbac4scim.github.io/api/v3/Roles?
    sortBy=displayName&sortOrder=ascending&
    startIndex=1&count=2&attributes=id,
    displayName,factory&excludedAttributes=
    schema,meta
```

LISTING 1. Example for a GET request for the first two roles with the attributes id, displayName and factory. The roles are sorted ascending by their displayName.

```
{
  "totalResults": 543,
  "startIndex": 1,
  "itemsPerPage": 2,
  "Resource": [
    {
      "id": "5a500cf5-feab-4a9b-8955-4
          f7cd01e8e8f",
      "displayName": "Blue Collar",
      "factory": "A"
    },
    {
      "id": "71cf406d-6d1d-42e6-a6ea-55
          fbcf51284e",
      "displayName": "Blue Collar Supervisor",
      "factory": "C"
    }
  ]
}
```

LISTING 2. Example for the response of the GET request of listing.

Listing 1 is an example for a GET request for roles. It demands roles sorted ascending by their displayName (sortBy and sortOrder parameters), starting with the first two roles (startIndex and count parameter), and the attributes id, displayName, and factory need to be returned while the attributes schema and meta are excluded (attributes and excludedAttributes parameters). The returned JSON in Listing 2 contains some steering attributes and the demanded

role list. The totalResults attribute denotes that 543 roles are present, while the startIndex and itemsPerPage attributes determine the current page for the pagination mechanism. The Resource attribute lists the demanded roles and attributes. In real application scenarios, we expect larger request (more attributes and more resources for each request) for all provided entities, which is configurable by raising the count parameter for more returned roles or by including more attributes with the attributes parameter.

Finally, after the role modeling tool has read the required data and can start to generate roles. When these roles are finalized, an export back to the IdM system is required. Basically, three operations need consideration: create, update, and delete. As described in Section V-C, the API offers HTTP methods for each operation: POST for creating, PUT (replace) or PATCH (partial update) for updating, and DELETE for deleting. Furthermore, a Bulk request allows for bundling multiple of these operations into one HTTP request raising its efficacy. After the export, both systems are in sync again, allowing for another iteration of hybrid role mining [15].

```
POST https://rbac4scim.github.io/api/v3/Roles?
    attributes=id

{
  "displayName": "White Collar Supervisor",
  "factory": "B"
}
```

LISTING 3. Example for a POST request to create a new role. On success only the generated id of the role returns. The body of the request contains a JSON with the displayName and factory for the new role.

```
{
  "id": "f3f3e28f-4c36-4f33-888c-034c57eb9700"
}
```

LISTING 4. Example for the response of the POST request of listing .

Listing 3 show a simplistic example for a POST request to create a new role. The role instantiates with a displayName and a factory but no assignments yet. On a successful creation of the role, only its id returns. The Listing 4 shows the successful response with the id.

In summary, the RBAC SCIM API is capable of supporting an add-on role modeling tool. The flexible setup of SCIM allows for custom attributes to match the given use case within organizations. Endpoints for querying resources support fine-grained filtering to only receive required resources and attributes. Exporting the generated roles is also possible by utilizing create, update, and delete operations.

C. QUALITY BASED ON PROPER REST API DESIGN

For a quality evaluation of the RBAC SCIM API, we closely examine realized REST API concepts, documentation, maintainability, and security considerations.

High-quality REST APIs realize REST concepts, including resource addressability, resource representations, uniform interface, statelessness, and hypermedia. (i) Resource addressability is provided by enforcing and consistently utilizing the *id* attribute in URI format as a unique address for the resource. (ii) Resource representations are implemented since entities of the implementing systems are *represented* as resources for the API. The Sections V-A and V-B elaborate on this distinction. (iii) Uniform interface is part of the RBAC SCIM profile since it uses standard HTTP methods and status codes in their semantic sense to manipulate resources. (iv) Statelessness is provided as the interactions are independent of other HTTP requests. (v) The API fully supports hypermedia as assigned resources come with a *location* attribute in its meta object, which allows additional navigation and discovery of related resources. In summary, the proposed REST API for RBAC internalized REST concepts, providing a high-quality REST API.

Guidelines for API documentation design cover three heuristics: efficient access to relevant content, an initial entry point for the API, and support for different learning strategies [22]. We examine these heuristics with the Swagger prototype described in Section V-D. (i) Efficient access to relevant content is provided by a consistent organization and navigation of the endpoints based on their resource and presenting conceptional information about the SCIM RFC family in the general description of the API. The suggested powerful search functions for the API are not present as these are a limitation of Swagger itself. (ii) An initial entry is given since the API documentation generates code examples and a recognizable API visualization. Additionally, the prototype provides a link with an overview for further background information for every API resource. (iii) Finally, different strategies for learning are supported. The documentation allows for learning about different resources independently, retrieving general and resource-based information, and providing try-out functionality for every endpoint. In summary, this prototype achieves sound documentation by utilizing Swagger's REST API documentation features.

API maintenance is a relevant topic when creating a new API or changing a present one. Changes to an API are challenging since implementing software might update only slowly or never to the latest version. It implies that API designers should avoid or carefully consider breaking changes. The RBAC profile for SCIM followed this advice however needed to apply some breaking changes to migrate to a more comprehensive data model of RBAC for SCIM. These breaking changes mainly include assignments. On the one hand, the user to entitlement assignment got removed due to the consideration of accounts. On the other hand, the internal structure of the assignments got modified to support a cleaner internalization of the hypermedia concept for REST APIs. However, other changes are additive and thus do not break the current data model and protocol of SCIM. In summary, breaking changes for the proposed API are carefully considered and only taken to extend the data model or internalize REST concepts [21].

From a security perspective, all entities depicted in Figure 6 are accessible by CRUD operations leading to a need for fine-grained permissions for the API. The assigned permissions and the suggestion to utilize OAuth2 for the API effectively protect against unauthorized access or fuzzing. Furthermore, SCIM provides tools to throttle both inbound and outbound traffic. These tools include restricting the returned amount of resources and attributes or capping bulk operations. Additional security considerations include geofencing and audit logs covered by implementing parties [24].

D. EFFICACY BASED ON EMULATING SCIM

SCIM emphasizes simplicity and scalability by various means. These include the usage of JSON [27] serialization, a pagination mechanism, attribute restrictions, filtering, bulk requests, or HTTP PATCH operations. (i) JSON serialization allows for efficient data exchange, especially in comparison to XML. (ii) A pagination mechanism splits big requests into multiple smaller requests (for example, max. 1,000 resources for a single request) and reduces efficacy risks for each request. Large requests may thus cause long processing times for servers and clients. Additionally, large requests tend to fail (exhausted memory, Etc.), raising processing and recovering times even more. (iii) Attribute restrictions contribute to efficacy twofold. On the one hand, query parameters allow restricting response attributes. On the other hand, SCIM servers can also restrict attributes' default visibility. (iv) Queries for each resource also allow for filtering by query parameters or a POST search request. Therefore, only required resources can be queried, which reduces unnecessary responses from the REST API. (v) Bulk requests enable bundling requests, which modify the state of the managed resources of the SCIM server. Avoiding atomic requests supports the SCIM server in processing bundled requests more efficiently. (vi) Finally, PATCH operations allow for decreasing the size of requests. The PATCH method only describes the required modification without requiring to send or even know all attributes of the resource itself.

At the time of writing, 70 implementors [9] adopted SCIM. Among these are big IdM vendors, like SailPoint, NetIQ, Oracle, Etc., which use SCIM in practice. These adoptions are evidence for sufficient real-world efficacy of the SCIM Core. The described means for the efficacy of the SCIM core are also considered for the SCIM extension of this work, allowing to expect the same efficacy as for the SCIM Core. In fact, the efficacy of the proposed API is primarily driven by the implementing systems, distorting an empirical evaluation of the API. These systems have to perform the querying and processing of modifications, while network conditions are a further external bias to the efficacy of the API.

VII. DISCUSSION

The discussion for the RBAC profile for SCIM covers three aspects. First, this work discloses the development cycles to show the trajectory and iterative nature of the development, as suggested by Hevner et al. [20]. Afterward, design principles formulate the insights generated by this work [19] to guide future designs of RBAC REST APIs. Finally, we present limitations of this work.

A. DISCLOSURE ON DEVELOPMENT CYCLES

As Hevner et al. [20] suggest, the development cycle is an iterative process. This work thus discloses this iterative process by outlining this project's trajectory and learning curve. Initially, this project intended no publication, nor was it targeting a full RBAC extension for SCIM. The plan was to wire up an advanced role modeling tool with an IdM system utilizing SCIM. The IdM system maintains about 30,000 users, 300,000 entitlements, 300 roles, and 10 applications. The project took on and off about 2.5 years for the productive go-live, measuring from the export step.

At first glance, an import seemed a trivial task as endpoints for crucial resources, including roles and entitlements, were present for this SCIM implementation. From a limited perspective, the missing role to entitlement relationship was considered a bug that was sloppily fixed with a database view. However, this database view was a source for various issues down the line. Despite other issues, converting from internal entities to external resources was a non-trivial task for outsiders. In this case, the root problems were uncovered REST concepts and an invalid RBAC model. On the one hand, the assignment's resource addressability and resource representation were not given. On the other hand, the RBAC model was invalid since the role to entitlement relationship of RBAC was missing.

The next step was exporting generated roles back to the IdM system. Since both the SCIM core and the implementation of SCIM lack comprehensive functionality to manipulate roles, we designed a custom plugin. This custom plugin covered a single POST endpoint which included all required functionalities at the time. The server plugin interpreted the request body to perform the desired actions. This logic violated nearly all REST concepts and merely applied for the lowest maturity for a REST API available [25]. Over time additional features and bug fixes were added. Some of the initial developers and API designers were no longer available or got replaced within a multi-national development team spanning over four involved companies. At some point, static code analysis provided more answers to questions than looking for them in the documentation. This situation eventually led to a well-documented redesign of the custom plugin following most REST concepts, improving quality and maintainability drastically. Lessons learned are boiled down to (i) a consequent utilization of REST concepts, (ii) clean and unambiguous documentation of REST APIs, and (iii) considering maintainability to avoid breaking changes for present APIs (backward) and future designs of the API (forward).

After this project, we decided to take the issue further and designed the API as described in this paper. Utilizing a theoretical background, past experiences, and a DSRM, a valid contribution to theory and practice originated.

B. GENERALIZATION BY DESIGN PRINCIPLES

Following a DSRM, the design of an artifact should loop back insights to the knowledge base and the environment. In addition to the contributions, design principles carefully formulate considerations for future RBAC REST APIs. The goal of this section is to highlight characteristics of RBAC REST APIs based on the insights of this work but independent of SCIM or the proposed artifact. Figure 9 illustrates the three identified design principles flexibility, validity, and simplicity while suggesting their interdependence. These design principles and their competing interdependence are discussed in the following [19].

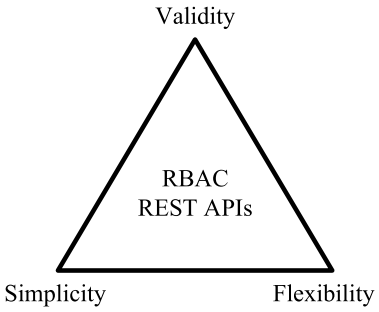

FIGURE 9. Design principles for future RBAC REST APIs.

Validity Design Principle: Provide a RBAC REST API with *a valid RBAC [11], [35] data model as derived in Figure 6 and a valid REST API design, including REST and HTTP semantics [23], [25], security considerations [24], and documentation [22]* in order for developers to *build on previous knowledge of these technologies and rely on a valid grounding based on theoretical suggestions, practical needs, and standardization*, given that *the implementing systems are interconnected, capable of processing RBAC data, and benefit from exchanging RBAC data*.

Evidence: The SCIM Core schema considers the presence of assignments users to entitlements or roles in a freestyle notation but fails in providing a comprehensive and valid RBAC model. The freestyle notation also fails a consistent resource addressability as a REST design principle. Furthermore, standalone role and entitlement endpoints are missing. Some vendors add these, but these also fail in providing, e.g. the role to entitlement assignment or further RBAC features, like sessions or (SoD) constraints. Violating validity leads to issues, e.g. missing but required RBAC resources or CRUD operation, which block certain use cases.

Simplicity Design Principle: Provide a RBAC REST API with *minimal overhead regarding the resources and their attributes, including simple and efficient means to modify resources and restrictions on the returned resources and attributes* in order for developers to *securely and efficiently share RBAC data without driving development expenses or complexity for organizations*, given that *all mandatory RBAC functions and resources are covered for the organization applying the RBAC REST API*.

Evidence: A focus of SCIM is simplicity which drives its success in practice. It's thus reasonable to adopt this simplicity for a RBAC REST API. Violating simplicity leads to unused features of the API, raising complexity and expenses for development. Resources, attributes, and HTTP endpoints thus should only be added based on shared reasoning for them across the connected applications.

Flexibility Design Principle: Provide a RBAC REST API with *flexible reusable HTTP methods allowing extensions regarding resources and their attributes to realize organization-specific, application-specific, or vendor-specific use cases* in order for developers to *efficiently extend and modify the RBAC REST API while simultaneously avoiding breaking changes to present or future extensions*, given that *these use cases and configurations are relevant and cover an application scenario for RBAC*.

Evidence: The presence of vendor-specific or even organization-specific SCIM APIs highlights the need for flexibility. On the one hand, vendors might focus on different aspects. An example of this is Oracle vs. SailPoint, which align their extensions on the identity or policy lifecycle. On the other hand, organizations (and vendors) also might need specific custom attributes which might not be reasonable for other organizations but crucial for the organization itself. Additionally, flexibility should not hamper progress by introducing breaking changes. SCIM specifies especially for this means to safely add custom attributes and resources. Violations of flexibility can lead to inabilities to enable vendor-specific or organization-specific use cases, breaking changes, or hampered adoption of the RBAC REST API.

Validity x *Simplicity Interdependence:* From the validity perspective, RBAC and REST are required, although simplicity restricts them to their basics. E.g., advanced but not yet adopted RBAC features might be relevant for certain use cases. However, for simplicity reasons, these features should be omitted for a general RBAC REST API and rather treated in dedicated extensions. From the simplicity perspective, the RBAC REST API should not be simpler as validity allows for. E.g., omitting specific CRUD operations or RBAC entities invalidates the API for its purpose. Finally, a valid and simple RBAC REST API fosters lower development expenses, while covering essential RBAC.

Validity x *Flexibility Interdependence:* From the validity perspective, RBAC and REST concepts require their usage as intended. Examples of this are repurposing roles for modeling organizational structures (like department trees) or violating validity by a different usage of HTTP semantics [23], like not using the methods POST for creating, GET for reading, PUT/PATCH for updating, or DELETE for deleting. From the flexibility perspective, a valid RBAC data model for the API needs opening points for configuration. Especially, organization-specific or vendor-specific RBAC use cases might require additional resources, attributes, or even further endpoints without violating its validity. As depicted in Table 4, vendors need and utilize this flexibility. Finally, a valid and flexible RBAC REST API fosters vendor-specific

or organization-specific application scenarios or use cases, while benefiting from the usage of valid groundings.

Simplicity x *Flexibility Interdependence:* From the simplicity perspective, simple and specified opening points for flexible extensions are required. Utilizing flexibility for adaptions thus should not bother the overall simplicity of the API. From the flexibility perspective, simplicity should not block flexibility in general, as vendor-specific or organization-specific use cases might not be covered by the essential design of the API. Finally, a simple and flexible RBAC REST API allows for efficient adaption for custom application scenarios and use cases.

C. LIMITATIONS

One methodological and one contentwise issue summarize the limitations of this work. This summary highlights open questions or current shortcomings of the proposed API design.

The methodological issue relates to Table 4. This work compares a few SCIM APIs with extensions for SCIM. Since unreported SCIM implementations may exist, further requirements may thus still lay undetected. Additionally, this work focuses on RBAC and thus rather a policy lifecycle. However, as seen for Oracle, we also detected a stream of extensions toward the identity lifecycle. Future work might further advance this identity lifecycle or ABAC.

The contentwise limitation is breaking changes. Although this work tries to avoid and minimize breaking changes, some are unavoidable in achieving a proper RBAC REST API, covering the RBAC data model or the inherent situation of present breaking vendor implementations. Breaking changes are thus mainly present for assignments or resources implemented differently by vendors. Furthermore, future extensions might add dynamic rules for determining assignments as this profile utilizes static assignments.

VIII. CONCLUSION

According to Gartner [10], SCIM runs through the *Trought of Disillusionment* reaching early mainstream. This state relentlessly reveals a clash of expectations and reality for the RFC family. In this sense, a literature discussion of SCIM is overdue. As seen in this work, a more scientific and methodological approach to design a RBAC profile for SCIM is helpful. Especially as it seems that the SCIM community tries to find a proper way to provide RBAC functionalities. Recent activities on SCIM, like the role and entitlement extension of Zollner [12] or extensions of SCIM by vendors, thus highlight shortcomings of SCIM for RBAC.

This work contributes with a RBAC profile for SCIM. Research on proper (REST) API design, SCIM specifications, literature, and implementations ensures consideration of relevant aspects. The data model (depicted in Figure 6) for the REST SCIM API is derived by RBAC and the present design of SCIM, its extensions, and implementations. Comprehensive HTTP methods allow a fine-grained management of RBAC resources covering standard RBAC functions [35]. Additionally, insights generated by this work are formulated as design principles in Section VII-B to pass them on for future RBAC API designs.

REFERENCES

[1] K. Li, P. Hunt, B. Khasnabish, A. Nadalin, and Z. Zeltsan, *System for Cross-domain Identity Management: Definitions, Overview, Concepts, and Requirements*, document RFC 7642, Sep. 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7642

[2] P. Hunt, K. Grizzle, E. Wahlstroem, and C. Mortimore, *System for Cross-domain Identity Management: Core Schema*, document RFC 7643, Sep. 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7643

[3] P. Hunt, K. Grizzle, M. Ansari, E. Wahlstroem, and C. Mortimore, *System for Cross-domain Identity Management: Protocol*, document RFC 7644, Sep. 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7644

[4] J. Smarr. (Dec. 2008). *Portable Contacts: A Common Format and Protocol for Accessing Contacts*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-smarr-vcarddav-portable-contacts-00

[5] S. Perreault, *vCard Format Specification*, document RFC 6350, Aug. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6350

[6] K. Zeilenga, *Lightweight Directory Access Protocol (LDAP): Directory Information Models*, document RFC 4512, Jun. 2006. [Online]. Available: https://www.rfc-editor.org/info/rfc4512

[7] N. Ragouzis, J. Hughes, R. Philpott, E. Maler, P. Madsen, and T. Scavo. (Mar. 2008). *Security Assertion Markup Language (SAML) V2.0 Technical Overview*. [Online]. Available: https://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf

[8] N. Sakimura, J. Bradley, M. B. Jones, B. de Medeiros, and C. Mortimore. (Nov. 2014). *OpenID Connect Core 1.0 incorporating Errata Set 1*. [Online]. Available: https://openid.net/specs/openid-connect-core-1_0.html

[9] S. Erdtman. (2023). *System for Cross-Domain Identity Management*. [Online]. Available: https://scim.cloud/#Implementations2

[10] F. Gaehtgens, "Hype cycle for digital identity, 2022," Gartner, Stamford, CT, USA, Tech. Rep. G00770428, Jul. 2022.

[11] R. S. Sandhu, "Role-based access control," *Adv. Comput.*, vol. 46, pp. 237–286, 1998. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0065245808602065

[12] D. Zollner. (Dec. 2022). *SCIM Roles and Entitlements Extension*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-scim-roles-entitlements/00/

[13] A. Puchta, S. Groll, and G. Pernul, "Leveraging dynamic information for identity and access management: An extension of current enterprise IAM architecture," in *Proc. 7th Int. Conf. Inf. Syst. Secur. Privacy*. Setúbal, Portugal: SciTePres, 2021, pp. 611–618.

[14] S. Groll, S. Kern, L. Fuchs, and G. Pernul, "Monitoring access reviews by crowd labelling," in *Trust, Privacy and Security in Digital Business*, S. Fischer-Hübner, C. Lambrinoudakis, G. Kotsis, A. M. Tjoa, and I. Khalil, Eds. Cham, Switzerland: Springer, 2021, pp. 3–17.

[15] L. Fuchs and G. Pernul, "Hydro–hybrid development of roles," in *Information Systems Security*, R. Sekar and A. K. Pujari, Eds. Berlin, Germany: Springer, 2008, pp. 287–302.

[16] L. Fuchs, G. Pernul, and R. Sandhu, "Roles in information security—A survey and classification of the research area," *Comput. Secur.*, vol. 30, no. 8, pp. 748–769, Nov. 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016740481100099X

[17] B. Mitra, S. Sural, J. Vaidya, and V. Atluri, "A survey of role mining," *ACM Comput. Surveys*, vol. 48, no. 4, pp. 1–37, May 2016, doi: 10.1145/2871148.

[18] S. Gregor and A. R. Hevner, "Positioning and presenting design science research for maximum impact," *MIS Quart.*, vol. 37, no. 2, pp. 337–356, Jun. 2013, doi: 10.25300/MISQ/2013/37.2.01.

[19] L. Chandra, S. Seidel, and S. Gregor, "Prescriptive knowledge in IS research: Conceptualizing design principles in terms of materiality, action, and boundary conditions," in *Proc. 48th Hawaii Int. Conf. Syst. Sci.*, Honolulu, HI, USA, Jan. 2015, pp. 4039–4048.

[20] R. H. Von Alan, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quart.*, vol. 28, no. 1, pp. 75–105, 2004.

[21] M. Lamothe, Y.-G. Guéhéneuc, and W. Shang, "A systematic review of API evolution literature," *ACM Comput. Surveys*, vol. 54, no. 8, pp. 1–36, Oct. 2021, doi: 10.1145/3470133.

[22] M. Meng, S. M. Steinhardt, and A. Schubert, "Optimizing API documentation: Some guidelines and effects," in *Proc. 38th ACM Int. Conf. Design Commun.* New York, NY, USA: Association for Computing Machinery, Oct. 2020, pp. 1–10, doi: 10.1145/3380851.3416759.

[23] R. T. Fielding, M. Nottingham, and J. Reschke, *HTTP Semantics*, document RFC 9110, Jun. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9110

[24] A. Lamba, "API design principles & security best practices–accelerate your business without compromising security," *Cybernomics*, vol. 1, no. 3, pp. 21–25, 2019. [Online]. Available: https://ssrn.com/abstract=3535436

[25] C. Rodríguez, M. Baez, F. Daniel, F. Casati, J. C. Trabucco, L. Canali, and G. Percannella, "Rest APIS: A large-scale analysis of compliance with principles and best practices," in *Web Engineering*, A. Bozzon, P. Cudre-Maroux, and C. Pautasso, Eds. Cham, Switzerland: Springer, 2016, pp. 21–39, doi: 10.1007/978-3-319-38791-8_2.

[26] T. Berners-Lee, R. T. Fielding, and L. M. Masinter, *Uniform Resource Identifier (URI): Generic Syntax*, document RFC 3986, Jan. 2005. [Online]. Available: https://www.rfc-editor.org/info/rfc3986

[27] T. Bray, *The JavaScript Object Notation (JSON) Data Interchange Format*, document RFC 8259, Dec. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8259

[28] J. Boyer, *Canonical XML Version 1.0*, document RFC 3076, Mar. 2001. [Online]. Available: https://www.rfc-editor.org/info/rfc3076

[29] A. Pfitzmann and M. Hansen, "A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management," TU Dresden, Dresden, Germany, Tech. Rep. 0.34, 2010. [Online]. Available: https://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf

[30] *Sarbanes-Oxley Act of 2002. Corporate responsibility*, United States Congr., Washington, DC, USA, 2002.

[31] J. Carretero, G. Izquierdo-Moreno, M. Vasile-Cabezas, and J. Garcia-Blas, "Federated identity architecture of the European eID system," *IEEE Access*, vol. 6, pp. 75302–75326, 2018.

[32] M. Just and C. Adams, *Identity Management*. Berlin, Germany: Springer, 2019, pp. 1–3. [Online]. Available: https://link.springer.com/referenceworkentry/10.1007/978-3-642-27739-9_78-2

[33] Y. Cao and L. Yang, "A survey of identity management technology," in *Proc. IEEE Int. Conf. Inf. Theory Inf. Secur.*, Dec. 2010, pp. 287–293.

[34] P. Samarati and S. C. de Vimercati, "Access control: Policies, models, and mechanisms," in *Foundations of Security Analysis and Design*, R. Focardi and R. Gorrieri, Eds. Berlin, Germany: Springer, 2001, pp. 137–196.

[35] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST standard for role-based access control," *ACM Trans. Inf. Syst. Secur.*, vol. 4, no. 3, pp. 224–274, Aug. 2001, doi: 10.1145/501978.501980.

[36] V. C. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone, "Guide to attribute based access control (ABAC) definition and considerations," U.S. Dept. Commerce, Washington, DC, USA, Tech. Rep., Tech. Rep. NIST SP 800-162, Jan. 2014, doi: 10.6028/nist.sp.800-162.

[37] O. Ulusoy and P. Yolum, "Norm-based access control," in *Proc. 25th ACM Symp. Access Control Models Technol.* New York, NY, USA: Association for Computing Machinery, Jun. 2020, pp. 35–46, doi: 10.1145/3381991.3395601.

[38] M. Gupta and R. Sandhu, "Towards activity-centric access control for smart collaborative ecosystems," in *Proc. 26th ACM Symp. Access Control Models Technol.* New York, NY, USA: Association for Computing Machinery, Jun. 2021, pp. 155–164, doi: 10.1145/3450569.3463559.

[39] A. Masoumzadeh, P. Narendran, and P. Iyer, "Towards a theory for semantics and expressiveness analysis of rule-based access control models," in *Proc. 26th ACM Symp. Access Control Models Technol.* New York, NY, USA: Association for Computing Machinery, Jun. 2021, pp. 33–43, doi: 10.1145/3450569.3463569.

[40] P. Iyer and A. Masoumzadeh, "Generalized mining of relationship-based access control policies in evolving systems," in *Proc. 24th ACM Symp. Access Control Models Technol.* New York, NY, USA: Association for Computing Machinery, May 2019, pp. 135–140, doi: 10.1145/3322431.3325419.

[41] T. Bui, S. D. Stoller, and H. Le, "Efficient and extensible policy mining for relationship-based access control," in *Proc. 24th ACM Symp. Access Control Models Technol.* New York, NY, USA: Association for Computing Machinery, May 2019, pp. 161–172, doi: 10.1145/3322431.3325106.

[42] P. Iyer and A. Masoumzadeh, "Active learning of relationship-based access control policies," in *Proc. 25th ACM Symp. Access Control Models Technol.* New York, NY, USA: Association for Computing Machinery, Jun. 2020, pp. 155–166, doi: 10.1145/3381991.3395614.

[43] P. Iyer and A. Masoumzadeh, "Effective evaluation of relationship-based access control policy mining," in *Proc. 27th ACM Symp. Access Control Models Technol.* New York, NY, USA: Association for Computing Machinery, Jun. 2022, pp. 127–138, doi: 10.1145/3532105.3535022.

[44] B. W. Lampson, "Protection," *ACM SIGOPS Operating Syst. Rev.*, vol. 8, no. 1, pp. 18–24, Jan. 1974, doi: 10.1145/775265.775268.

[45] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman, "Protection in operating systems," *Commun. ACM*, vol. 19, no. 8, pp. 461–471, Aug. 1976, doi: 10.1145/360303.360333.

[46] D. E. Bell and L. J. LaPadula, "Secure computer systems: Mathematical foundations," MITRE, Bedford, MA, USA, Tech. Rep. AD0770768, 1973.

[47] K. J. Biba, "Integrity considerations for secure computer systems," MITRE, Bedford, MA, USA, Tech. Rep. ADA039324, 1977.

[48] S. Parkinson and S. Khan, "A survey on empirical security analysis of access-control systems: A real-world perspective," *ACM Comput. Surveys*, vol. 55, no. 6, pp. 1–28, Dec. 2022, doi: 10.1145/3533703.

[49] S. Kern, T. Baumer, S. Groll, L. Fuchs, and G. Pernul, "Optimization of access control policies," *J. Inf. Secur. Appl.*, vol. 70, Nov. 2022, Art. no. 103301. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214212622001533

[50] A. Puchta, F. Böhm, and G. Pernul, "Contributing to current challenges in identity and access management with visual analytics," in *Data and Applications Security and Privacy XXXIII*. Cham, Switzerland: Springer, 2019, pp. 221–239, doi: 10.1007/978-3-030-22479-0_12.

[51] D. Servos and S. L. Osborn, "Current research and open problems in attribute-based access control," *ACM Comput. Surveys*, vol. 49, no. 4, pp. 1–45, Jan. 2017, doi: 10.1145/3007204.

[52] G. Cole. (Sep. 2005). *OASIS Service Provisioning Markup Language (SPML) Version 2*. OASIS. [Online]. Available: https://docs.oasis-open.org/provision/spml-2.0-cd-01/pstc-spml2-cd-01.pdf

[53] D. Hardt, *The OAuth 2.0 Authorization Framework*, document RFC 6749, Oct. 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6749

[54] E. Rissanen, *eXtensible Access Control Markup Language (XACML) Version 3.0*, OASIS, OASIS Standard XACML-V3.0, Jan. 2013. [Online]. Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf

[55] OASIS. (Oct. 2014). *XACML v3.0 Core and Hierarchical Role Based Access Control (RBAC) Profile Version 1.0*. [Online]. Available: http://docs.oasis-open.org/xacml/3.0/rbac/v1.0/xacml-3.0-rbac-v1.0.pdf

[56] M. Kunz, A. Puchta, S. Groll, L. Fuchs, and G. Pernul, "Attribute quality management for dynamic identity and access management," *J. Inf. Secur. Appl.*, vol. 44, pp. 64–79, Feb. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214212618301467

[57] M. Diodati. (Aug. 2010). *Consensus on the Future of Standards-Based Provisioning and SPML*. [Online]. Available: https://blogs.gartner.com/mark-diodati/2010/08/20/consensus-on-the-future-of-standards-based-provisioning-and-spml/

[58] P. Hunt, K. Grizzle, M. Ansari, and D. Olds. (Dec. 2012) *SCIM 2.0 Token Search Extension*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-hunt-scim-tokensearch/00/

[59] Internet Engineering Task Force. (Dec. 2012). *SCIM 2.0 Extended Search*. [Online]. Available: https://datatracker.ietf.org/doc/draft-hunt-scim-xsearch/00/

[60] P. Hunt, M. Ansari, and A. Nadalin. (Jul. 2013). *OAuth 2.0 SCIM Client Registration Profile*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-hunt-oauth-scim-client-reg/00/

[61] P. Hunt. (Aug. 2013). *SCIM Directory Services*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-hunt-scim-directory/01/

[62] K. Li. (Jan. 2013). *SCIM User Scenarios*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-li-scim-user-scenarios/00/

[63] P. Hunt, B. Khasnabish, A. Nadalin, Z. Zeltsan, and K. Li. (Aug. 2013). *SCIM Use Cases*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-zeltsan-scim-use-cases/02/

[64] B. Greevenbosch and R. Sun. (Dec. 2014). *SCIM and vCard Mapping*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-greevenbosch-scim-vcard-mapping/04/

[65] M. Wahl. (May 2014). *SCIM Profile For Enhancing Just-In-Time Provisioning*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-wahl-scim-jit-profile/02/

[66] M. Ansari and P. Hunt. (Mar. 2015). *SCIM Soft Delete*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-ansari-scim-soft-delete/00/

[67] P. Hunt, M. Ansari, and I. Glazer. (Mar. 2015). *SCIM Event Notification*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-hunt-scim-notify/00/

[68] P. Hunt and G. Wilson. (Mar. 2015). *SCIM Password Management Extension*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-hunt-scim-password-mgmt/00/

[69] P. Hunt. (May 2015). *SCIM HTTP Search Method Extension*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-hunt-scim-search/00/

[70] P. Hunt, W. Denniss, and M. Ansari. (Mar. 2016). *SCIM Event Extension*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-hunt-idevent-scim/00/

[71] P. Hunt. (Feb. 2016). *System for Cross-Domain Identity Management: Discovery*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-hunt-scim-discovery/00/

[72] C. McMurtry. (Apr. 2016). *SCIM Polling Protocol*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-mcmurtry-scim-polling/01/

[73] K. Grizzle, B. Yoder, J. Jones, P. Lieberman, and E. Nunez. (Oct. 2017). *SCIM Extension for Privileged Access Management*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-grizzle-scim-pam-ext/01/

[74] P. Hunt and M. Ansari. (Jul. 2017). *SCIM Use Cases for SECEVENTS*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-hunt-secevent-usecases/00/

[75] M. Wahl. (Jun. 2019). *SCIM Profile for Provisioning Users Into Relying Party Applications*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-wahl-scim-profile/00/

[76] P. Hunt. (Oct. 2021). *SCIM Protocol: Multi-Value Filtering Extension*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-hunt-scim-mv-filtering/00/

[77] D. Zollner. (Oct. 2021). *SCIM Verified Domains Extension*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-zollner-scim-domain-extension/00/

[78] Internet Engineering Task Force. (Jul. 2022). *SCIM Referential Value Location Extension*. [Online]. Available: https://datatracker.ietf.org/doc/draft-zollner-scim-referential-value-location/01/

[79] M. Peterson and D. Zollner. (Feb. 2023). *Cursor-based Pagination of SCIM Resources*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-scim-cursor-pagination/00/

[80] P. Hunt and N. Cam-Winget. (Jan. 2023). *SCIM Profile for Security Event Tokens*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-scim-events/01/

[81] M. Shahzad, H. Iqbal, and E. Lear. (Apr. 2023). *Device Schema Extensions to the SCIM Model*. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/draft-shahzad-scim-device-model/04/

[82] A. Sahi, D. Lai, and Y. Li, "A review of the state of the art in privacy and security in the eHealth cloud," *IEEE Access*, vol. 9, pp. 104127–104141, 2021.

[83] E. Bertino and K. Brancik, "Services for zero trust architectures—A research roadmap," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Sep. 2021, pp. 14–20.

[84] R. Shere, S. Srivastava, and R. K. Pateriya, "A review of federated identity management of OpenStack cloud," in *Proc. Int. Conf. Recent Innov. Signal Process. Embedded Syst. (RISE)*, Oct. 2017, pp. 516–520.

[85] T. H. Vo, W. Fuhrmann, and K.-P. Fischer-Hellmann, "How to adapt authentication and authorization infrastructure of applications for the cloud," in *Proc. IEEE 5th Int. Conf. Future Internet Things Cloud (FiCloud)*, Aug. 2017, pp. 54–61.

[86] K. Dodanduwa and I. Kaluthanthri, "Trust-based identity sharing for token grants," in *Proc. 3rd Int. Conf. Cryptography, Secur. Privacy*. New York, NY, USA: Association for Computing Machinery, Jan. 2019, pp. 168–173, doi: 10.1145/3309074.3309078.

[87] V. Beltran and E. Bertin, "Unified communications as a service and WebRTC: An identity-centric perspective," *Comput. Commun.*, vol. 68, pp. 73–82, Sep. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366415002492

[88] A. Costa, E. Sciacca, F. Vitello, U. Becciani, P. Massimino, S. Riggi, and D. Sanchez, "An integrated workspace for the Cherenkov telescope array," *Future Gener. Comput. Syst.*, vol. 94, pp. 811–819, May 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17309585

[89] R. Kumar and R. Goyal, "On cloud security requirements, threats, vulnerabilities and countermeasures: A survey," *Comput. Sci. Rev.*, vol. 33, pp. 1–48, Aug. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013718302065

[90] H. Nacer, N. Djebari, H. Slimani, and D. Aissani, "A distributed authentication model for composite web services," *Comput. Secur.*, vol. 70, pp. 144–178, Sep. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404817301153

[91] A. Majeed and A. Al-Yasiri, "Consolidate the identity management systems to identify the effective actor based on the actor relationship for the Internet of Things," in *Proc. 3rd Int. Congr. Inf. Commun. Technol.*, X.-S. Yang, S. Sherratt, N. Dey, and A. Joshi, Eds. Singapore: Springer, 2019, pp. 755–765.

[92] A. Ceccanti, E. Vianello, and F. Giacomini, "Beyond X.509: Token-based authentication and authorization in practice," in *Proc. EPJ Web Conf.*, vol. 245, 2020, p. 03021, doi: 10.1051/epjconf/202024503021.

[93] P. Djerasimović, "Geographic dependency of identity-associated data," *Automatika*, vol. 59, nos. 3–4, pp. 340–348, Oct. 2018, doi: 10.1080/00051144.2018.1530827.

[94] D. Haddon and P. Bennett, *The Emergence of Post COVID-19 Zero Trust Security Architectures*. Cham, Switzerland: Springer, 2021, pp. 335–355, doi: 10.1007/978-3-030-72120-6_13.

[95] A. A. Mahdi, "Offensive threats," *J. Univ. Babylon*, vol. 25, no. 2, pp. 364–370, 2017.

[96] P. Marillonnet, M. Ates, M. Laurent, and N. Kaaniche, "An efficient user-centric consent management design for multiservices platforms," *Secur. Commun. Netw.*, vol. 2021, pp. 1–19, Jun. 2021, doi: 10.1155/2021/5512075.

[97] H. Sato, "Authorization by documents," *J. Inf. Process.*, vol. 25, pp. 766–774, Jun. 2017.

[98] R. Shere, S. Shrivastava, and R. Pateriya, "CloudSim framework for federation of identity management in cloud computing," *Int. J. Comput. Eng. Res. Trends*, vol. 4, no. 6, pp. 269–276, Jun. 2017. [Online]. Available: http://ijcert.org/ems/ijcert_papers/V4I6011.pdf

[99] H. Truong, J. L. Hernández-Ramos, J. A. Martinez, J. B. Bernabe, W. Li, A. M. Frutos, and A. Skarmeta, "Enabling decentralized and auditable access control for IoT through blockchain and smart contracts," *Secur. Commun. Netw.*, vol. 2022, pp. 1–14, Jun. 2022, doi: 10.1155/2022/1828747.

[100] A. Ashish, A. Millar, T. Mkrtchyan, P. Fuhrmann, G. Behrmann, M. Sahakyan, O. S. Adeyemi, J. Starek, D. Litvintsev, and A. Rossi, "DCache, towards federated identities & anonymized delegation," *J. Phys., Conf. Ser.*, vol. 898, no. 10, Oct. 2017, Art. no. 102009, doi: 10.1088/1742-6596/898/10/102009.

[101] T. Q. Thanh, S. Covaci, B. Ertl, and P. Zampognano, "An integrated access control service enabler for cloud applications," in *Future Network Systems and Security*, R. Doss, S. Piramuthu, and W. Zhou, Eds. Cham, Switzerland: Springer, 2015, pp. 101–112.

[102] B. Ertl, U. Stevanovic, A. Hayrapetyan, B. Wegh, and M. Hardt, "Identity harmonization for federated HPC, grid and cloud services," in *Proc. Int. Conf. High Perform. Comput. Simul. (HPCS)*, Jul. 2016, pp. 621–627.

[103] S. Nakandala, H. Gunasinghe, S. Marru, and M. Pierce, "Apache airavata security manager: Authentication and authorization implementations for a multi-tenant escience framework," in *Proc. IEEE 12th Int. Conf. e-Science (e-Science)*, Oct. 2016, pp. 287–292.

[104] J. B. Bernabé, J. L. Hernandez-Ramos, and A. F. Gómez-Skarmeta, "Holistic privacy-preserving identity management system for the Internet of Things," *Mobile Inf. Syst.*, vol. 2017, Aug. 2017, Art. no. 6384186, doi: 10.1155/2017/6384186.

[105] A. Ceccanti, M. Hardt, B. Wegh, A. Millar, M. Caberletti, E. Vianello, and S. Licehammer, "The INDIGO-datacloud authentication and authorization infrastructure," *J. Phys., Conf. Ser.*, vol. 898, Oct. 2017, Art. no. 102016, doi: 10.1088/1742-6596/898/10/102016.

[106] J. Innerbichler, S. Gonul, V. Damjanovic-Behrendt, B. Mandler, and F. Strohmeier, "NIMBLE collaborative platform: Microservice architectural approach to federated IoT," in *Proc. Global Internet Things Summit (GIoTS)*, Jun. 2017, pp. 1–6.
[107] J. L. Hernández-Ramos, S. Pérez, C. Hennebert, J. B. Bernabé, B. Denis, A. Macabies, and A. F. Skarmeta, "Protecting personal data in IoT platform scenarios through encryption-based selective disclosure," *Comput. Commun.*, vol. 130, pp. 20–37, Oct. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366418302123
[108] A. A. Corici, Y. Shashi, M. Corici, R. Shrestha, and D. Guzman, "Enabling dynamic IoT security domains: Cellular core network and device management meet authentication framework," in *Proc. Global IoT Summit (GIoTS)*, Jun. 2019, pp. 1–6.
[109] B. de Matos Patrocínio dos Santos, B. Dzogovic, B. Feng, V. T. Do, N. Jacot, and T. Van Do, "Towards achieving a secure authentication mechanism for IoT devices in 5G networks," in *Proc. 6th IEEE Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)/ 5th IEEE Int. Conf. Edge Comput. Scalable Cloud (EdgeCom)*, Jun. 2019, pp. 130–135.
[110] B. Santos, B. Dzogovic, B. Feng, V. T. Do, N. Jacot, and T. Van Do, "Cross-federation identities for IoT devices in cellular networks," in *Proc. 24th IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2019, pp. 1745–1751.
[111] N. Selvanathan, D. Jayakody, and V. Damjanovic-Behrendt, "Federated identity management and interoperability for heterogeneous cloud platform ecosystems," in *Proc. 14th Int. Conf. Availability, Rel. Secur.* New York, NY, USA: Association for Computing Machinery, Aug. 2019, doi: 10.1145/3339252.3341492.
[112] A. A. Corici, M. Corici, E. Troudt, B. Riemer, and T. Magedanz, "Framework for trustful handover of M2M devices between security domains," in *Proc. 23rd Conf. Innov. Clouds, Internet Netw. Workshops (ICIN)*, Feb. 2020, pp. 102–109.
[113] P. Leach, M. Mealling, and R. Salz, *A Universally Unique IDentifier (UUID) URN Namespace*, document RFC 4122, Jul. 2005. [Online]. Available: https://www.rfc-editor.org/info/rfc4122

THOMAS BAUMER received the M.Sc. degree (Hons.) from the Honors Elite Program, University of Regensburg, in 2020. He is currently pursuing the Ph.D. degree with the University of Regensburg, with a research focus on maintaining an IT security level in a changing environment by grasping synergies from research and practice. He studied management information systems with a specialization in IT security with the University of Regensburg and KU Leuven. Since 2020, he has been a Software Engineer with Nexis GmbH, a spin-off from the Chair of Information Systems I (Prof. Dr. Günther Pernul, University of Regensburg) specializing in identity and access governance and analytics.

MATHIS MÜLLER received the bachelor's degree in management information systems from the University of Regensburg, where he is currently pursuing the M.Sc. degree in management information systems. After graduating from his master's studies, he will start the Ph.D. degree with the Chair of Information Systems I, University of Regensburg. Before the master's studies, he had already completed an apprenticeship as an IT Specialist with Vodafone Germany. He is with Nexis GmbH, a spin-off from the Chair of Information Systems I (Prof. Dr. Pernul, University of Regensburg). His main focus in the study is on IT security. He was able to substantiate this focus not only from a research point of view but through several internships also from a practical point of view.

GÜNTHER PERNUL (Member, IEEE) received the Diploma and Ph.D. degrees (Hons.) in business informatics from the University of Vienna, Austria. He is currently a Professor with the Department of Information Systems, University of Regensburg, Germany. Previously, he held positions with the University of Duisburg–Essen, Germany; the University of Vienna; the University of Florida, Gainesville, FL, USA; and the College of Computing, Georgia Institute of Technology, Atlanta, GA, USA. His research interests include data and information security aspects, data protection and privacy, data analytics, and advanced data-centric applications.