Zero Trust Federation: Sharing Context under User Control towards Zero Trust in Identity Federation

Koudai Hatakeyama
Kyoto University
Kyoto, Japan
hatakeyama@net.ist.i.kyoto-u.ac.jp

Daisuke Kotani
Kyoto University
Kyoto, Japan
kotani@media.kyoto-u.ac.jp

Yasuo Okabe
Kyoto University
Kyoto, Japan
okabe@media.kyoto-u.ac.jp

***Abstract*—Perimeter models, which provide access control for protecting resources on networks, make authorization decisions using the source network of access requests as one of critical factors. However, such models are problematic because once a network is intruded, the attacker gains access to all of its resources. To overcome the above problem, a Zero Trust Network (ZTN) is proposed as a new security model in which access control is performed by authenticating users who request access and then authorizing such requests using various information about users and devices called contexts. To correctly make authorization decisions, this model must take a large amount of various contexts into account. However, in some cases, an access control mechanism cannot collect enough context to make decisions, e.g., when an organization that enforces access control joins the identity federation and uses systems operated by other organizations. This is because the contexts collected using the systems are stored in individual systems and no federation exists for sharing contexts. In this study, we propose the concept of a Zero Trust Federation (ZTF), which applies the concept of ZTN under the identity federation, and a method for sharing context among systems of organizations. Since context is sensitive to user privacy, we also propose a mechanism for sharing contexts under user control. We also verify context sharing by implementing a ZTF prototype.**

***Index Terms*—Access controls, Identity Federation, Zero Trust, User Managed Access**

I. Introduction

To securely restrict access to information systems and resources, organizations need access control to determine who can gain access to what resources. The perimeter model [1] is currently popular for access control. In this model, a network is divided into zones based on levels of trust. These levels of trust in the source networks of access requests are a critical factor for authorization decisions.

However, this model cannot address some problems that have recently surfaced. Due to the wide dissemination of mobile devices and remote work, internal systems are accessed not only from an organization's internal network but also from the internet. Access requests to resources can no longer be assumed to originate from an internal network. Technologies like VPNs partly solve this problem. However, with cloud computing, resources owned by organizations are now located outside such internal networks as the internet, and the need to use VPN to access the internal network is lower than before popularization of the cloud. Furthermore, once a resource inside the perimeter is hijacked, the attacker can freely access any resources in the same zone.

To address these problems, a new access control model called a Zero Trust Network (ZTN) has been proposed [2] [3]. This model does not assume such trustworthy attributes as source networks. Instead, it verifies and evaluates whether a user who requests access is trustworthy at every access request. Based on evaluation results, the decision whether to permit access is contemplated.

When verifying and evaluating an access request, the access control mechanism uses "context," which is information about the entity making an access request: information about the user, the device being employed by her, the network to which her device is connected, the physical environment surrounding her, etc. The context includes both such static information as user IDs and device vendors and dynamic information based on past behavior, such as what device was used for recent access and where it was accessed.

A various, large amount of context must be collected to perform context-based access control in ZTN. However, an access control mechanism cannot collect enough context to make decisions in some cases, e.g., when an organization that enforces access control joins the identity federation and uses systems operated by other organizations. This is because the contexts collected using other systems are stored in individual systems and no federation for sharing contexts exists.

Under the identity federation, an Identity Provider (IdP) cannot provide contexts with attributes like user's age. This is because a simple IdP cannot collect such contexts as detailed information whether the device is vulnerable and because an IdP cannot treat dynamically changing attributes like contexts in the same way as conventional static attributes. Even if an IdP can provide contexts in addition to an identity, an organization that enforces access control using these contexts and identities cannot comply with the ZTN concept because the organization relies on the trust of a single IdP. Contexts are also sensitive to user privacy and cannot simply be shared without user consent. Context must be collected from various systems operated by organizations, not just from a system operated by a single organization that enforces access control. Current ZTN implementations can only handle as much context as a single system can collect.

This study proposes a federation that shares contexts as

well as a method that shares context among systems operated by multiple organizations. We also propose a mechanism for sharing context under user control to protect user privacy. We design and implement a mechanism that shares context based on the scheme of a protocol for security-event sharing called Continuous Access Evaluation Protocol, which is currently being standardized by Shared Signals and Events WG of OpenID Foundation[1]. Access to contexts by systems is managed by an access control management protocol called User Managed Access [4] [5], which extends an authorization delegation protocol called OAuth2.0.

The main contribution of this study is that it proposes the idea of Zero Trust Federation (ZTF), which applies the ZTN concept under the identity federation, and shows how ZTF can share contexts under user control.

The following is the structure of this paper. Section 2 describes the background, and Section 3 describes the concept of Zero Trust Federation (ZTF). Section 4 proposes a system to share context under user control in ZTF. Section 5 describes its evaluation with a use case using a prototype that implements it. Finally, Section 6 summarizes this paper.

II. Background

A. Zero Trust Network (ZTN)

ZTN is an access control model whose core principle is "Never Trust, Always Verify." ZTN does not rely on trustworthy attributes for access control like source networks in traditional perimeter models ("Never Trust"). Instead, it verifies every single access request. "Context" is used for this verification. A context is a set of information about the requesting entity, which includes various data about the user who requested access, about the device being employed by the user, and so on.

1) ZTN Architecture: According to Gilman [6], ZTN Architecture consists of control and data planes. The control plane, which must determine whether to allow access to protected resources, is also called a Policy Decision Point (PDP) [3]. A data plane is the place where a user communicates with a resource and where access control is enforced. An access control enforcement point is also called a Policy Enforcement Point (PEP) [3].

2) ZTN Implementation Example: BeyondCorp is Google's implementation of ZTN in its internal network, whose concept has been summarized in several white papers [7] [8] [9]. This implementation is unique because it implements an access proxy [9] that combines PEP and PDP to enable ZTN access control. This (reverse) proxy, also known as an Identity-Aware Proxy (IAP) [2], assumes the responsibility of verifying access requests and enforcing authorization decisions. When access is granted, the IAP establishes a secure communication channel between the user and the protected resource. If denied, the IAP blocks access.

By combining PEP and PDP, policy changes can be applied quickly and consistently [9]. On the other hand, technical and administrative challenges remain when systems being operated by different organizations try to use IAPs together to collect sufficient context to determine authorization decisions. The technical challenges include an IAP, which might cause a single point of failure and a performance bottleneck because these systems communicate with users by a single IAP. The administrative challenges include managing the security policies of different organizations in one place.

B. Context-based Authentication and Authorization

As mentioned in the previous section, ZTN uses contexts to verify every access request to authenticate the user who requested access and to authorize it.

Context-based authentication is sometimes called risk-based authentication. Examples using context include IP addresses [10], mouse movements, and keystrokes [11]. In addition to using context, these authentications are characterized by continuously verifying the user's identity [11]. This is also called Continuous Authentication [12]. For example, an authentication authority, which collects the operation history of a touchscreen and continues to verify those who are currently operating it, has the same characteristics as in the previous history [12]. In addition to one-to-one authentication between a user and an authentication authority, continuous authentication under identity federation has also been studied [13] [14].

As an example of context-based authorization, perhaps only the staff members who are currently at headquarters are allowed to access a database [15]. Authorization is based on the actual environment in which a device is located [16].

C. Identity Federation

Identity federation (IdF) is a mechanism that shares user identities among organizations. Here identity is the set of information about a user, such as her ID and any affiliations/organizations to which she belongs, like a university or workplace. In the federation, an authentication authority, sometimes called an identity provider, authenticates users and issues assertions which include user identity and an authority signature. Such an entity, which employs user identities in a federation called a service provider or a Relying Party (RP), verifies the integrity of assertions and uses them to securely identify users.

An example of an inter-university IdF is Gakunin [3] through which users can seamlessly access services provided by different universities without having to register as a new user, for example, connecting to wireless LANs at different universities without registration.

Shibboleth [4] using SAML [17] or OpenID Connect [18] are technologies that establish identity federations.

[1] https://openid.net/wg/sse/
[2] https://cloud.google.com/iap/
[3] https://www.gakunin.jp/
[4] https://www.shibboleth.net/

D. Continuous Access Evaluation Protocol

Continuous Access Evaluation Protocol (CAEP), introduced by Google, is a new event-sharing protocol for Continuous Authentication in IdF [14]. With CAEP, the RP can share internally generated events about users with the IdP that authenticates them. The event includes a change of the network being used or a vulnerability that is discovered in the device being used. In other words, IdP can sense updates for user contexts that occurred at the RP with this protocol and manage continuous authentication based on these notified contexts. In addition, IdP can notify RPs of the authentication results and such additional information as events about Authenticator Assurance Level (AAL) [5]) changes due to stronger authentication for other RPs. RPs make authorization decisions using contexts about the user based on this information.

This protocol is currently being standardized in the Shared Signals and Events Working Group in the OpenID Foundation[6].

E. User Control in Federation

Sharing a user's personal information must fall under the rubric of user control. For example, Gakunin can share the attributes of users as well as their pseudonymous IDs. Such user attributes as email addresses or employment/university affiliations are examples of personal information. With uApproveJP [19], since user attributes can be shared with a user's consent in the IdF, sharing user attributes under user control is enforced in Gakunin.

OAuth2.0 [20], which is a protocol for authorization delegation, allows a third party (a Client) to access a resource server on behalf of the resource owner who provides authorization within a limited access scope. On a resource server, the resource owner deploys its resources and is protected by an authorization server, which asks the resource owner for an authorization decision for resource access requests from clients. When the resource owner approves, the authorization server issues an access token to the Client, which presents the access token to the resource server, which verifies it and determines the validity of the access. In this way, the resource owner can delegate authorization to the Client with OAuth2.0.

III. Zero Trust Federation

This section defines the Zero Trust Federation and proposes its design and implementation.

A. Problem

Assume an organization controls access to its systems based on the Zero Trust Network (ZTN) concept. The organization uses an Identity Provider (IdP) operated by another organization and ensures security by introducing various Software as a Service (SaaS), such as Mobile Device Management (MDM) and Endpoint Detection and Response (EDR), which are operated by outside organizations.

[5]https://pages.nist.gov/800-63-3/sp800-63b.html
[6]https://openid.net/wg/sse/

To perform zero-trust access control, the access control mechanism of an organization's system must use a sufficient amount of contexts when making authorization decisions. For example, when the EDR detects a malware infection on a device, the EDR service immediately notifies the system's access control mechanism, which recalculates the trust level of the access request from that device. When MDM detects unmanaged devices as access sources, the access control mechanism must restrict access based on its security policy. When another system, which is not directly related to this organization but under the same identity federation, detects suspicious behavior, the access control mechanism restricts access based on that information.

A federation that shares contexts among organizations is essential for zero-trust access control in the identity federation. Therefore, in this study, we consider a federation among systems operated by different organizations to achieve ZTN in their systems and propose Zero Trust Federation and its design.

B. Zero Trust Federation

We define Zero Trust Federation (ZTF) as a federation that allows each RP to perform zero-trust access control by federating with IdP and CAP (Fig. 1). Identity Provider (IdP) is an entity that identifies and authenticates users and provides authentication assertions to entities under the identity federation. Context Attribute Provider (CAP) collects and manages contexts about a user and provides context information to entities. A Relying Party (RP) controls access using assertions from IdP and contexts from CAP. RP is also where a user requests access.

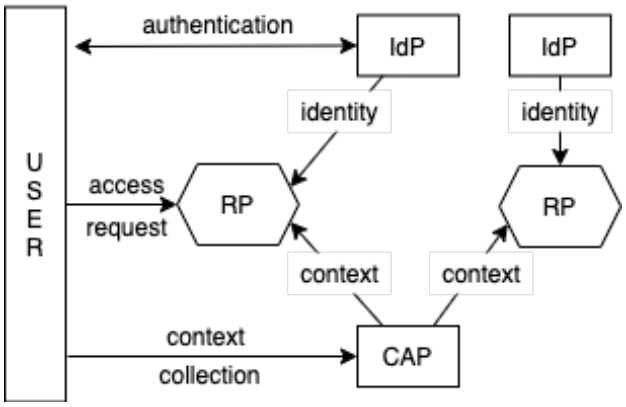

Fig. 1. RPs in ZTF

For example, an EDR is a CAP. The anomalies detected by EDR (e.g., malware detection) are reported to the RP as a context. The RP then recalculates the access request's trust level from the device where the anomaly was detected and denies or restricts access based on the new trust level and the policy.

ZTF allows multiple RPs to share contexts among multiple RPs (Fig. 1). Consider a CAP that logs the network to which a device is connected. This service can determine whether the

devices are in the same network as the one usually employed by the user. Suppose she regularly uses RP1 and infrequently uses RP2, for example, once a year. In ZTF, the RP2 can share with RP1 through CAP the context that includes whether the user accessed the network from devices connected to it that she usually uses. Without this federation, such information is unknown. Control access can be based on this context.

C. Context Attribute Provider

ZTF introduces a new entity called CAP, which 1) collects contexts from users and 2) provides contexts to RPs.

Contexts can be collected in two ways: by installing an agent on devices used by users and directly gathering them or indirectly doing so from RPs. The former collects more detailed information. For example, when a CAP is an EDR, the CAP has an agent installed on a device to audit the device's health. But it is not always possible to deploy agents to every device. In that case, the CAP collects as much context as the RPs because they communicate directly with users more frequently. Some contexts managed by the CAP, such as EDR, can only be collected by deploying agents to the user side.

A context is provided in the following way. An RP requests a CAP to provide contexts of a user who has requested access. Upon receiving the request, the CAP identifies the user and provides her contexts to the RP. When the CAP detects that user's contexts has been updated, it provides the updated context to the RP.

D. Access Flow in ZTF

Figure 2 shows the flow in the ZTF from the user's viewpoint from when an access request is made to RP1 until a decision is made. We assume that a user has previously accessed RP2 and that the context is indirectly collected from RPs.

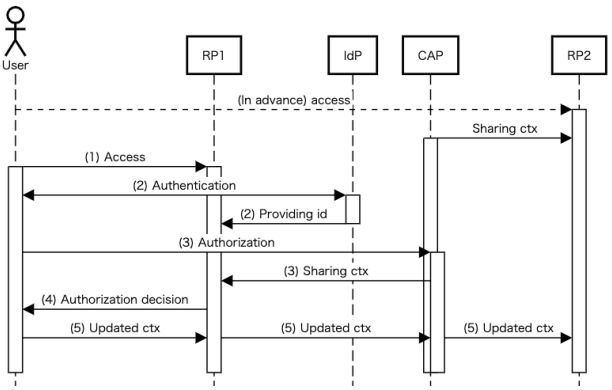

Fig. 2. Access flow in ZTF

1) The user tries to access RP1;
2) The user provides an identity (id) to RP1 using IdP;
3) The user provides a context (ctx) to RP1 using CAP;
4) The RP1 determines an authorization decision based on the given context and identity;
5) The RP1 monitors the behavior of the End-User, and if the context has changed, notifies the CAP, which shares the notified context with other RPs.

E. ZTF Implementation

Mechanisms for providing identities and sharing contexts are necessary to implement ZTF. In this study, we assume that OpenID Connect [18] is used for an identity federation and CAEP [14] is used for context sharing. However, since CAEP has not been standardized yet and has no specifications, we implemented it based on the assumption that specifications will eventually be developed based on the current designs being discussed.

1) Providing Identity: An identity representing users is provided for the CAP and RP from the IdP that authenticated them. A user does not necessarily provide the same identity to the CAP and RP using the same IdP. However, when the context is shared by RP and CAP, the user must be identifiable by sharing the same identifier using a single IdP to transmit and receive contexts without involving her.

Therefore, we need an IdP that provides at least one identifier shared by RP and CAP. In this implementation, we prepared different IdPs for RPs shared by RPs and CAP, and for CAP.

2) Sharing Context: Two communication channels are provided for sharing contexts: 1) an RP (or a CAP agent) provides contexts to CAP and 2) a CAP provides contexts to RPs. These channels are established with CAEP, and contexts are transmitted and received on channels.

A context is represented by an extended JWT[7] called SET[8], as proposed in CAEP.

CAEP is also based on an IETF draft called "Management API for SET Event Streams" [21]. In CAEP, contexts are transmitted over Event Streams. The context's receiver provides such configuration information as the target-user identifier by the Event Stream Management Endpoint and requests that a stream be established. The transmitter configures a stream based on the configuration information and transmits the context.

IV. Sharing Context Under User Control

This section describes a mechanism for sharing context under user control in ZTF.

A. Problem

In ZTF, RPs are provided with the context from the CAP and can perform zero-trust access control. CAPs can also collect contexts from RPs and manage them. However, since contexts are information that includes user privacy, contexts must be shared under user control. Therefore, we investigate a mechanism in which user authorization is mandatory to establish context sharing.

[7]https://tools.ietf.org/html/rfc7519
[8]https://tools.ietf.org/html/rfc8417

B. Design

1) What users should control: Contexts can be shared or provided in three ways:

- from a CAP to a RP;
- from a RP to a CAP;
- from a CAP agent to a CAP.

The purpose of this study is to share contexts across systems managed by different organizations under user control. Therefore, we consider providing contexts from a CAP to a RP and from a RP to a CAP. Since the only difference between these two flows is that the CAP and RP roles are reversed, we investigate a mechanism that provides contexts from a CAP to an RP under user control.

2) Control Flow: Sharing contexts under user control means that the CAP provides the RP with limited contexts authorized by the user when the latter requests a context from the former.

The flow of user authorization and context sharing from CAP to RP is shown in Fig 3.

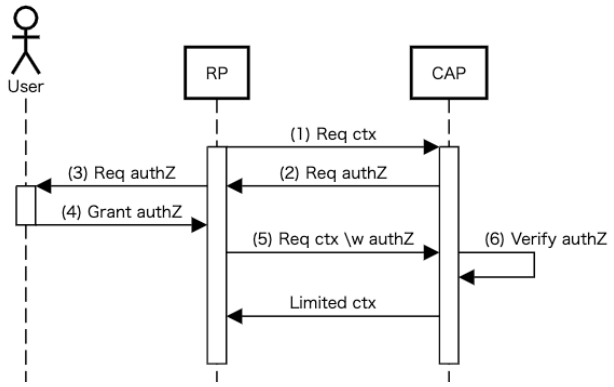

Fig. 3. Access flow in ZTF

1) An RP requests (req) a context sharing (ctx) from a CAP.
2) The CAP verifies that the RP is authorized by the user, and if not, responds (resp) that the RP must be granted authorization (authZ) by the user.
3) The RP requests the user to get authorization.
4) The user checks what contexts the RP wants and authorizes the RP to get them.
5) The RP requests ctx again.
6) The CAP verifies that the RP is authorized by the user and determines whether to establish a context sharing based on the authorization results.

C. Implementation

In this study, we used the User Managed Access protocol (UMA) [4] [5] that extends OAuth2.0.

1) User Managed Access: User Managed Access is a protocol based on OAuth2.0. It extends OAuth2.0 from the following two points. In OAuth2.0, the Client is operated by a resource owner who can make authorization decisions. But in UMA, the Client is operated by an entity called a Requesting Party that requests access to resources, which is not always a resource owner. In some cases no entity involved in the authorization delegation flow can make authorization decisions. For this reason, UMA extends the authorization server to make authorization decisions depending on the policy set by the resource owner in advance. These extensions allow a Requesting Party to access the resource server within the range defined by the resource owner in its policy.

The following are two advantages of using UMA in ZTF: 1) the authorization server can centrally manage which RPs are allowed to share contexts and to what extent; 2) the policies based on RP attributes can be written in the authorization server to grant authorization to the RPs without a consent action each time to establish a context sharing with a new RP.

2) Where the authorization server should be located: Since the UMA also proposed a protocol to loosely couple a resource server and an authorization server, the authorization server is not necessarily managed by the CAP. In this implementation, the authorization server is managed by an IdP, which provides the identity used for the RP and CAP to share contexts.

V. Use Cases and Considerations

A. Prototype

We prepared a prototype that satisfies the implementations shown in Sections 3 and 4.

The prototype was created using Golang[9], with Golang standard packages, JWT libraries[10] and HTTP utility libraries[11] as external libraries. The source code is available on GitHub[12].

B. Use Cases

Using the prototype, we confirmed that the context can be shared in the following use cases. We assumed a situation with three entities, CAP, RP1, and IdP, and a user. We also prepared a program that mimics CAP's agent that detects user context updates and notifies the CAP.

1) The user makes an access request to RP1.
2) The RP1 requests the authorization server (AuthZSrv) to get authorization (authZ).
3) Since the AuthZSrv has no RP1 policy, it asks the user to authorize access to her contexts from RP1.
4) The user sets a policy that allows partial contexts to be provided to RP1.
5) AuthZSrv issues an authZ token based on that policy to the RP1.
6) The RP1 requests the CAP to provide contexts with a token.
7) The CAP verifies the token and shares partial contexts granted by the policy.
8) The RP1 receives the contexts provided from the CAP under user control.
9) The agent detects context updates and notifies the CAP of them.

[9]https://golang.org/
[10]https://github.com/lestrrat-go/jwx
[11]https://github.com/gorilla/
[12]https://github.com/hatake5051/ztf-prototype/tree/percom

10) The RP1 shares updated contexts by CAP.

Using this scenario, we confirmed that RP1 can get contexts that would be unknown if it had not participated in the ZTF and that the contexts sent to the RP1 are within the allowable scope. A description of the use case is available on Github[13].

C. Considerations

As seen in the previous section, our proposal allows users to control their context sharing. Considering contexts as user attributes, we can compare our proposal with uApproveJP, which is an attribute information federation system with user consent. Although both uApproveJP and our proposal have the same mechanism for providing contexts (user attributes) with user authorization, there are differences in the meaning of the granted authorization and the frequency at which information is updated. For the authorization method, uApproveJP assumes that the user grants authorization by pushing a consent button. On the other hand, our proposal allows users to set a policy, freeing them from the burden of bestowing consent every time a new context sharing occurs. This is an example how our proposal improves the user experience. In our proposed ZTF, the context shared between RP and CAP is expected to be frequently changed. Since uApproveJP assumes that the values of the user attributes are not updated frequently, uApproveJP has a mechanism that requests consent again whenever the attribute values are changed. The user's convenience is reduced when uApproveJP is used in ZTF. In our proposal, the responsibility for revoking the granted authorization is left to the user.

VI. Conclusion

We proposed the concept of Zero Trust Federation (ZTF), which applies the concept of ZTN under the identity federation. We proposed a mechanism for sharing contexts among organizations under user control in ZTF and verified context sharing by implementing a ZTF prototype.

In a federation that shares contexts, two aspects are important: reactively sharing context updates and context sharing that is under user control. For the former, we designed a mechanism to transmit and receive contexts using a protocol called CAEP, which is currently being standardized by Shared Signals and Events WG of OpenID Foundation as of this paper's writing. For the latter, we designed a user control mechanism with a user-centric access control protocol called UMA and confirmed that our proposed method can be used in ZTF to share contexts among organizations under user control.

Future work will address the following problems: 1) standardizing the format and the semantics of contexts in ZTF; 2) operating the authorization server, e.g., how to write policies and where to deploy them; and 3) which identifier to use when sharing contexts.

[13]https://github.com/hatake5051/ztf-prototype/blob/percom/example/usecase.md

References

[1] S. Northcutt *et al.*, *Inside Network Perimeter Security (2nd Edition)*. USA: Sams, 2005.

[2] J. Kindervag, "Build Security Into Your Network's DNA: The Zero Trust Network Architecture," 2010.

[3] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero Trust Architecture," NIST SP 800-207, 9 2019.

[4] E. Maler *et al.*, "User-Managed Access (UMA) 2.0 Grant for OAuth 2.0 Authorization," Kantara Initiative, Jan 2018.

[5] M. Machulak and J. Richer, "Federated Authorization for User-Managed Access (UMA) 2.0," Kantara Initiative, Jan 2018.

[6] E. Gilman and D. Barth, *Zero Trust Networks Building Secure Systems in Untrusted Networks*. O'Reilly Media, 2017.

[7] R. Ward and B. Beyer, "BeyondCorp: A New Approach to Enterprise Security," *USENIX ;login:*, vol. Vol. 39, No. 6, pp. 6–11, 2014.

[8] B. Osborn *et al.*, "BeyondCorp: Design to Deployment at Google," *USENIX ;login:*, vol. 41, pp. 28–34, 2016.

[9] B. Spear *et al.*, "Beyond Corp: The Access Proxy," *USENIX ;login:*, 2016.

[10] N. N. Diep *et al.*, "Contextual Risk-Based Access Control," *Security and Management*, vol. 2007, pp. 406–412, 2007.

[11] I. Traore *et al.*, "Combining Mouse and Keystroke Dynamics Biometrics for Risk-Based Authentication in Web Environments," in *2012 Fourth International Conference on Digital Home*, Nov 2012, pp. 138–145.

[12] M. Frank *et al.*, "Touchalytics: On the Applicability of Touchscreen Input as a Behavioral Biometric for Continuous Authentication," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 1, pp. 136–148, Jan 2013.

[13] D. Preuveneers and W. Joosen, "SmartAuth: Dynamic Context Fingerprinting for Continuous User Authentication," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, 2015, p. 2185–2191.

[14] A. Tulshibagwale, "Re-thinking federated identity with the Continuous Access Evaluation Protocol," https://cloud.google.com/blog/products/identity-security/re-thinking-federated-identity-with-the-continuous-access-evaluation-protocol, 2 2019.

[15] K. Minami and D. Kotz, "Secure context-sensitive authorization," *Pervasive and Mobile Computing*, vol. 1, no. 1, pp. 123 – 156, 2005.

[16] M. J. Covington *et al.*, "Securing Context-Aware Applications Using Environment Roles," in *Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies*, 2001, p. 10–20.

[17] OASIS Security Services TC, "Security Assertion Markup Language (SAML) v2.0," http://docs.oasis-open.org/security/saml/v2.0/sstc-saml-approved-errata-2.0.pdf, 3 2005.

[18] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore, "Final: OpenID Connect Core 1.0 incorporating errata set 1," OpenID Foundation, 11 2014.

[19] T. Orawiwattanakul *et al.*, "User consent acquisition system for Japanese Shibboleth-based academic federation (GakuNin)," *International Journal of Grid and Utility Computing*, vol. 2, no. 4, pp. 284–294, 2011.

[20] D. Hardt, "The OAuth 2.0 Authorization Framework," RFC 6749, Oct. 2012.

[21] M. Scurtescu *et al.*, "Management API for SET Event Streams," Internet Engineering Task Force, Jun. 2017.