An Evaluation of Single Sign-On in a Zero Trust Environment

Harris W. McCall

Columbia Basin College

CSIA-450: Cybersecurity Capstone

Matthew Boehnke

November 22nd, 2024

Abstract

This research evaluates the Single Sign-on (SSO) process and determines whether or not it has a place in a Zero-Trust Environment. Single Sign-on's purpose is to make the lives of users easier by lowering the number of accounts the user needs to know the authentication details for, and by limiting the number of times the user is asked to sign in. This is typically done by utilizing one of two protocols to achieve authentication; Security Assertion Markup Language (SAML) or OpenID Connect (OIDC). SAML v2.0 has been an open standard since 2005 and has been primarily used in enterprise SSO scenarios as it provides a standardized way to handle authentication and authorization across different platforms and technologies. OIDC achieves the same goals as SAML through different mechanisms and is typically used outside enterprise environments. Both of these protocols contain vulnerabilities that could result in the compromise of not just one, but multiple services and platforms. Zero Trust Architecture (ZTA) is currently the most robust security model to date, operating under the assumption that no single user or system should be given authorization until fully authenticated in accordance with stringent security policies. Users in ZTA are asked to sign in via a username and password, along with a second form of authentication (MFA), and are still only granted access if the system they are using meets specific security policy requirements.

Introduction

Single Sign-On (SSO) systems have become a prevalent solution for managing user authentication across multiple platforms. SSO uses protocols such as OpenID Connect and SAML to improve user experience by allowing access to various service providers (SP) through a single identity provider (IdP). However, this convenience also introduces potential vulnerabilities and single points of entry that can be exploited by threat actors to gain unauthorized access to multiple user accounts without the need for user credentials. While SSO does improve the quality of life for users, its core functionality operates in contrary to Zero Trust Architecture's more stringent methods of granularly making access decisions per every request.

Zero Trust Architecture (ZTA) operates on the principle that no user or system should be automatically trusted. Instead, access is granted only after stringent authorization that complies with strictly defined security policy, with users receiving only the minimum privileges necessary to perform their tasks. The growing importance of ZTA can be shown via a recent directive issued by President Joseph R. Biden in 2021 (Executive Order no. 14028). This order mandates federal agencies to adopt Zero Trust principles as defined by the Cybersecurity and Infrastructure Security Agency (CISA, 2023). If it is important for our most secure systems, like those used for federal government purposes, to adopt a security model such as Zero Trust, then that should also be the goal of every sector, both public and private, as well. Through analysis of existing research on Single Sign-on vulnerabilities, protocol documentation, enumerated common vulnerabilities, and federal governance this research will determine if SSO has a place in Zero Trust Environments.

Problem Statement

Single Sign-on protocols utilize elements that allow a user to be authorized through authentication elsewhere and therefore implicitly trusted, this appears, at first glance, to directly conflict with the core principle of a Zero-Trust model.

Single Sign-on (SSO)

If you have ever signed up on a website you have never visited before via the “Login with Google”, “Login with X”, or “Login with Facebook” buttons, you have used Single Sign-on. You effectively used an account you had with one service (the Identity Provider, or IdP) to access the resources of a different service (the Service Provider, or SP). You didn’t have to go through the process of registering with that service using your email, and creating a password to go along with it. (Dineshchandgr, 2022)

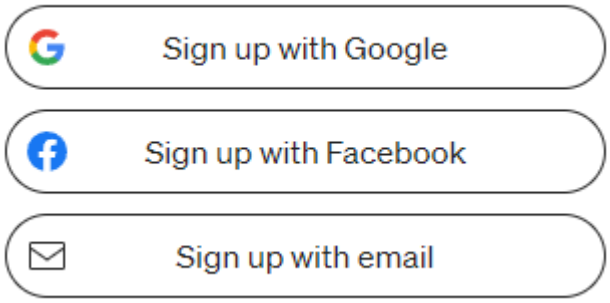

Figure 1: SSO Example

This is an example of SSO in action, which it accomplishes via authentication and authorization protocols like Security Assertion Markup Language or OpenID Connect. These protocols allow the SP to obtain proof of authentication via the IdP. The methods in which they do this can vary, and sometimes multiple protocols are used in conjunction with each other to validate a user’s identity (Somorovsky et al., 2012). These protocols will have varying levels of security to them depending on how they are implemented, but any time users’ lives are made easier by something like SSO it can create potential attack vectors for malicious actors.

IdP user accounts become a prime target for threat actors because gaining access to that many different interconnected services could prove lucrative. Especially if there are methods of

obtaining account access without the need for the user account credentials. Which is why SSO protocols should be carefully implemented, if they are used at all.

Security Assertion Markup Language (SAML)

SAML is an XML-based protocol that enables secure communication between IdPs and SPs. When a user attempts to access a service, the SP redirects the user to the IdP for authentication. The user logs in, and the IdP generates a SAML assertion, which contains information about the user's identity and attributes. This assertion is sent back to the SP who then verifies the assertion and grants access to the user based on the information provided. This process allows users to access multiple services with a single login, enhancing convenience (OASIS, 2005). This approach also means to improve security by reducing password fatigue and through better management of user credentials (OWASP, 2024).

SAML is used in many sectors that benefit from a zero-trust approach to security. For instance, SAML is frequently used in enterprise and cloud applications such as Salesforce, Google Workspace, and Microsoft 365 to centralize user management. It is also used in government and healthcare sectors that require strict compliance with security and privacy regulations (OASIS, 2005).

Given the sensitive nature of many of these industries, SAML must be appropriately configured to maintain a strong security posture and limit attack vectors. An early example of how a SAML misconfiguration can lead to vulnerabilities was found in 2008 by Armando et al. These researchers found that Google had omitted the ID and SP fields in the authentication assertion sent by SAML. This allowed malicious SPs to reuse the authentication assertion to impersonate a legitimate user at another, legitimate SP. This vulnerability in Google’s SSO

implementation has since been fixed, but it serves as an important example of why SAML should be fully and appropriately configured. (Armando et al., 2008)

The vulnerability found by Armando et al. (2008) can be described simply as a misconfiguration vulnerability, due to, essentially, laziness on Google's part in generating SAML assertions. However, in 2012 Somorovsky et al. discovered vulnerabilities that allowed for the manipulation of the XML used for making SAML assertions. This type of attack is called an XML Signature Wrapping (XSW) Attack. The researchers determined that there could be two types of adversaries; *Advacc*, a person who registers as a user with an IdP, receiving valid signed SAML assertions about themselves who then modifies those assertions to impersonate other users; and *Advintc*, an attacker who finds SAML assertions on the internet, likely from unprotected networks or caches, and uses those to create malicious assertions. This ability to modify SAML assertions via XSW Attacks is a much more serious vulnerability, that can still exist if SAML is not implemented properly. (Somorovsky et al., 2012)

Having been designed with security in mind, SAML can be made very secure. It is just up to the programmers who are implementing it to be sure it is compliant with best practices and security policy. Some examples of this include using TLS 1.2 for guaranteeing message confidentiality and integrity at the transport layer, digitally signing messages with a certified key to guarantee message integrity and authentication, and encrypting assertions via XMLEnc to prevent disclosure of sensitive attributes post transportation. These measures help with mitigating Man-in-the-middle, eavesdropping, forged assertions, message modifications, and theft of user authentication information among other things. By validating message confidentiality and integrity, protocol usage, signatures, protocol processing rules, binding implementation, and security countermeasures SAML is incredibly secure. (OWASP, 2024)

OpenID Connect (OIDC)

The next most used protocol for enabling SSO is called OpenID Connect (OIDC). OpenID Connect is an identity layer built on top of OAuth 2.0, allowing clients to verify the identity of an end-user based on the authentication performed by an authorization server. It also transmits profile information about end-users in an interoperable manner, allowing for integration with a wide variety of services. OIDC is designed to be simple to implement by providing a framework for identity interactions on the web. (Urueña et al., 2014)

Where SAML sends assertions via XML which contain identifying information about a user, OIDC uses tokens for a similar purpose. These tokens are distributed after the user authenticates with the desired IdP (OpenID Foundation, n.d.). Once the user is authenticated, the IdP issues carefully crafted tokens to the user designed for the specific resources the user is accessing, these are the Access Token and the ID Token (Chiarelli, 2021). The access token provided by OIDC is based on OAuth 2.0, and as such authorizes only the least amount of access for the resource being used. While the access token provides authorization to access certain resources on behalf of the user, the ID Token is used to prove that the user has authenticated with the IdP.

Authorization of access to resources within an OIDC implementation is handled by the OAuth 2.0 protocol, which, unless it is properly configured, can lead to attackers gaining unauthorized access to resources. To mitigate these vulnerabilities, some precautions should be made. For instance, HTTP 303 redirects should be used instead of HTTP 307 redirects, this is because HTTP 307 redirects can contain user credentials that can be intercepted. Utilizing HTTPS is also always recommended, as it provides transport-layer encryption. Also, the IdP should be identified in the redirect URI in such a way that it cannot be tampered with (the “iss”

parameter) (Fett et al., 2016). Some other ways of securing OAuth include using: PKCE for Public Clients, CSRF Protection, Sender-Constrained Tokens, and Client Authentication, although, this list is not exhaustive (OWASP, 2024).

Evidently, OIDC runs into a similar issue as SAML when it comes to properly implementing recommended security measures and following the best practices for this protocol. Even in 2017, many implementations of OIDC still did not adhere to the security guidelines outlined in the protocol specifications. In addition to the prevalence of misconfigured implementations of OIDC, 75% of the OIDC libraries were susceptible to at least one Single-Phase Attack, and all of them were vulnerable to Cross-Phase Attacks. (Mainka et al., 2017)

Some of the Single-Phase Attacks that were found to be effective by Mainka et al. were attacks such as Identity Spoofing, Replay Attacks, Signature Bypass, and Wrong Recipient. These types of attacks are much easier to take advantage of than the Cross-Phase attacks like IdP Confusion or a Malicious Endpoints Attack (Mainka et al., 2017). Cross-Phase Attacks are more complicated to perform due to multiple steps and interactions between different entities like the End-User, SP, and IdP. The Malicious Endpoints Attack was discovered in 2016, and takes advantage of the Discovery and Dynamic Registration extensions used by OIDC. When an End-User initiates authentication, they unknowingly interact with a malicious Discovery service that provides manipulated URLs which allow for capturing authentication tokens, or worse things like Server-Side Request Forgery or Code Injection (Mladenov et al.).

Mladenov et al. and Mainka et al. have reported their findings to the OpenID Connect and OAuth specification authors and improvements were made to the protocol itself, however it is still not invulnerable to misconfiguration issues. In 2024 alone, there have been 21 new

enumerated common vulnerabilities across various different applications and plugins, most of which are due to an improperly configured implementation of OIDC protocol (CVE, 2024).

As previously discussed, there are attacks that can be performed on OIDC-dependent platforms due to misconfigurations. However, security evaluation tools like PrOfESSOS are being developed and distributed for the purpose of testing OIDC protocol against known attacks before publishing systems that make use of OIDC for its SSO functionality (Mainka et al., 2017). Tools like PrOfESSOS paired with following all the recommendations in the protocol specifications can effectively secure OIDC against every currently known attack.

Zero Trust Architecture (ZTA)

Zero Trust Architecture (ZTA) is a cybersecurity framework which operates in a sort of "guilty, until proven innocent" way. In essence, no single user or device is ever trusted by default. Every device, whether within or outside the network perimeter, must be authenticated before being granted granular access to only necessary resources. (CISA, 2023)

In light of Executive Order 14028 (2021), the Cybersecurity and Infrastructure Security Agency (CISA) developed a Zero Trust Maturity Model, and released an updated version 2.0 to the public in 2023. This document offers guidance to organizations in navigating through 4 different stages of maturity. In the first stage of maturity, traditional network security methods are still in place. This is your average SOHO perimeter-based security, relying on implicit trust within the network, that use static access controls with minimal visibility into user and device behavior. (CISA, 2023)

When organizations move into the next stage, they are usually beginning to implement automation and start to assess their security posture. They may begin to automate Identity and Access Management, and some cross-pillar solutions. These cross-pillar solutions include

mechanisms such as Device Management, Network Segmentation, and Application Security wherein the organization will be trying to maintain some level of compliance with security policies among most of the devices on the network. (CISA, 2023)

In the third stage of ZTA implementation, more comprehensive automation is being integrated, creating stronger security measures across different pillars (Identity, Devices, Networks, Applications, and Data). The organization will, at this point, also be starting to leverage real-time risk assessments and dynamic policy enforcement. The enhanced visibility and analytics in this stage allow for better monitoring and response to security incidents. (CISA, 2023)

By the end of the fourth and final stage of ZTA implementation, an organization will have reached a fully automated and integrated Zero Trust Environment. In this environment, security policies are dynamically enforced based on real-time data and contextual information which ensures that access is granted on a per-session basis, inside and outside the network. Even though the security of a network is very strong at this point, continuous monitoring and improvement of security measures to maintain compliance with governance as it becomes relevant. (CISA, 2023)

While CISA provides a good maturity model that generally guides organizations through what the different stages of ZTA look like, they do not go into specific detail about the logical components that make maintaining a Zero Trust Environment possible. However, the National Institute of Standards and Technology (NIST) does. In NIST Special Publication 800-207 (2020), the authors include a whole section dedicated to describing and discussing the different technical components of ZTA. (Rose et al., 2020)

There are three core components of a ZTA. These are the Policy Engine (PE), Policy Administrator (PA), and the Policy Enforcement Point (PEP). The PE makes the ultimate decision for granting access to resources by using enterprise policy and input from other various systems like continuous diagnostics and mitigation (CDM), industry compliance, ID management, and security information and event management (SIEM) systems. This decision is made by taking all of the information from those systems and policies and inputting them into a trust algorithm that decides whether to grant, deny, or revoke access. After the decision is made, the PE logs the decision and the PA executes the decision, allowing or denying access to the user. PEPs are effectively gatekeepers at the resource level, directly allowing or preventing access based on the decided trustworthiness of an entity. (Rose et al., 2020)

Conclusion

Zero Trust Architecture requires an ID management system to keep track of users, their roles, and their permissions. SSO protocols offer the ability to securely create, manage, and store user IDs along with pertinent user information from a centralized IdP. Organizations that are moving towards optimal ZTA should consider making use of SSO protocols like SAML and OIDC to fill this requirement. SSO can be effectively leveraged by ZTA, but SSO protocols are not always necessarily secure enough for use in a Zero Trust Environment.

As continuous improvement is a directive of ZTA, the way that SSO is implemented may change over time, but in general, these protocols should be implemented following the recommendations made in each of their corresponding documentations. As new attacks and vulnerabilities in these protocols are discovered, the affected protocol and its documentation will be edited to reflect effective countermeasures. This means that as long as all security best practices (including threat awareness) and documentation are followed, along with potentially

making use of more than one of these accredited protocols, a high level of security can be maintained for user accounts while keeping user management centralized, automated, and easily monitored.

References

Armando, A., Carbone, R., Compagna, L., Cuellar, J., & Tobarra, L. (2008). Formal analysis of SAML 2.0 web browser single sign-on. 1-10. https://doi.org/10.1145/1456396.1456397

Chiarelli, A. (2021, October 28). *ID Token and Access Token: What's the Difference?* Retrieved from auth0.com: https://auth0.com/blog/id-token-access-token-what-is-the-difference/

CISA. (2023, April). *Zero Trust Maturity Model V2.0.* Retrieved from cisa.gov: https://www.cisa.gov/sites/default/files/2023-04/CISA_Zero_Trust_Maturity_Model_Version_2_508c.pdf

CVE. (2024). *Search Results*. Retrieved from cve.mitre.org: https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=OpenID+Connect

Dineshchandgr. (2022, July 8). *Single Sign-On (SSO): SAML, OAuth2, OIDC simplified*. Retrieved from medium.com: https://medium.com/javarevisited/single-sign-on-sso-saml-oauth2-oidc-simplified-cf54b749ef39

Executive Order no. 14028. (2021, May 12). Executive Order on Improving the Nation's Cybersecurity. Retrieved from https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

Fett, D., Küsters, R., & Schmitz, G. (2016). A Comprehensive Formal Security Analysis of OAuth 2.0. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. https://doi.org/10.48550

Mainka, C., Mladenov, V., Schwenk, J., & Wich, T. (2017). SoK: Single Sign-On Security — An Evaluation of OpenID Connect. *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, 251-266. https://doi.org/10.1109/EuroSP.2017.32

Mladenov, V., Mainka, C., & Schwenk, J. (2016). On the security of modern Single Sign-On Protocols: Second-Order Vulnerabilities in OpenID Connect. *arXiv: Cryptography and Security*. https://doi.org/10.48550/arXiv.1508.04324

OASIS. (2005, March 15). *Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0.* Retrieved from oasis-open.org: https://docs.oasis-open.org/security/saml/v2.0/saml-sec-consider-2.0-os.pdf

OpenID Foundation. (n.d.). *How OpenID Connect Works*. Retrieved from openid.net: https://openid.net/developers/how-connect-works/

OWASP. (2024). *OAuth 2.0 Protocol Cheatsheet*. Retrieved from owasp.org: https://cheatsheetseries.owasp.org/cheatsheets/SAML_Security_Cheat_Sheet.html

OWASP. (2024). *SAML Security Cheat Sheet.* Retrieved from owasp.org: https://cheatsheetseries.owasp.org/cheatsheets/SAML_Security_Cheat_Sheet.html

Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020, August). *NIST SP 800-207 Zero Trust Architecture.* https://doi.org/10.6028/NIST.SP.800-207

Somorovsky , J., Mayer , A., Schwenk , J., Kampmann , M., & Jensen, M. (2012). On Breaking {SAML}: Be Whoever You Want to Be. *21st USENIX Security Symposium (USENIX Security 12)*, 397--412. Retrieved from https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/somorovsky

Urueña, M., Muñoz, A., & Larrabeiti, D. (2014). Analysis of privacy vulnerabilities in single sign-on mechanisms for multimedia websites. *Multimedia Tools and Applications,* 159-176. https://doi.org/10.1007/s11042-012-1155-4