LemonLDAP::NG - A Full AAA Free Open Source WebSSO Solution

Christophe Maudoux, Selma Boumerdassi, Selma Boumerdassi

LemonLDAP::NG
A Full AAA Free Open Source WebSSO Solution

Christophe Maudoux
CNAM/Cedric
Paris, France

Selma Boumerdassi
CNAM/Cedric
Paris, France

***Abstract*—Nowadays, security is becoming a major issue and concern. More and more organizations like hospitals, metropolis or banks are under cyberattacks and have to improve their network infrastructure security. The first prerequisites are to authenticate users, to provide identity and to grant just the needed and useful accesses. These requirements can be solved by implementing a Single Sign-On (SSO) solution. It is an authentication scheme that permits a user to log in with a single identity to any of several related, yet independent, systems. It allows users to log in once and to access services without authenticating again. SSO solutions are classified depending on *Authentication*, *Authorization*, and *Accounting* features. The 'AAA' acronym defines a framework for intelligently controlling access to resources, enforcing security policies, auditing usage, and providing the information necessary to bill for services. These combined processes are considered *important* for effective *network management* and *cybersecurity*. `LemonLDAP::NG` (LL::NG) is a full AAA WebSSO solution. It implements all standard authentication and identity federation (IdF) protocols. The main LL::NG's advantages compared to other products are its plug-in engine and its advanced handler-based protection mechanism that can be employed to protect Server2Server exchanges or to offer the SSO as a Service, a solution to implement a full DevOps architecture. LL::NG is a community and professional project mainly employed by the French government to secure Police, Finance or Justice Ministries and a French mobile operator IT infrastructures since 2010. But for several years, contributions come from all around the world and LL::NG is becoming more and more popular.**

***Index Terms*—WebSSO, Security, AAA, OIDC, SAML, CAS, MFA, Handler, Server2Server exchanges, DevOps, SSO as a Service, Authentication, Authorization, Accounting**

I. Introduction

SSO is a centralized session and user authentication service in which one set of login credentials can be used to access multiple applications. Main advantage is in its simplicity; the service authenticates you on one designated platform or portal, enabling you to then use a variety of services without having to log in and out each time. Deploy an SSO architecture avoids multiplication of passwords and increase the overall IT security. Full SSO solutions provide the three essential services that are users authentication with different methods, access control with authorizations, and accounting by providing logs.

`LemonLDAP::NG` [1] works with PSGI, FastCGI or uwsgi [2] standards/protocols compliant web servers like `Nginx`, `Apache`, `Starman`. It offers different ways for *authenticating* users, checks *authorizations* before accessing resources, and provides logs for *accounting*. It is composed of four main components as depicted by Fig. 1.

(i) The *Portal* implements standard protocols. When a user tries to access an application protected by LL::NG and connected by using authentication or IdF protocols, he is redirected to the *Portal*. It displays authentication screen and then, redirects user to the requested application or it displays the applications menu. *(ii)* The *Handler* (depicted by a lock) can be employed to protect applications that do not implement standard authentication or IdF protocols but working with HTTP headers. Handler code can be embedded by applications or Reverse Proxies as explained in Section IV-D. When an unauthenticated user tries to access an application protected by a LL::NG handler, he is redirected to the Portal by the Handler. Then, the Portal displays authentication screen, builds a SSO session in the sessions back-end, provides a SSO cookie with the Session ID (SID) and redirects the authenticated user to the requested application. With the SID, the Handler can retrieve sessions data and send session attributes to the protected application by using HTTP headers. *(iii)* LL::NG relies on different *back-ends* that can be files, LDAP, Active Directory, SQL or noSQL (for SSO sessions) databases to store configuration, SSO sessions and persistent sessions (with history, second factors, consents), and users data. *(iv)* The *Manager* is the WebSSO administration interface used by administrator to set access rules, HTTP headers sent to protected applications, define the applications menu, configure identities and services providers, enable authentication methods.

Rest of this paper is organized as follow. Section II is an overall presentation of the LL::NG project. In Section III, we expose main authentication methods. Standard authentication or IdF protocols and applications protection are described in Section IV. LL::NG implements a plug-ins engine and some community supported plug-ins are presented in Section V.

II. Background & Related Work

SSO platforms can be implemented for different use cases. To increase security, we can not rely on heterogeneous products. So, it is important to provide complete solutions like LemonLDAP::NG.

A. History & Development

The first stable LL::NG version has been released in 2010. It was a fork of `LemonLDAP`, an OpenLDAP-based authentication project that has been modified by the French 'Gendarmerie Nationale' (GN) to meet its specific needs.

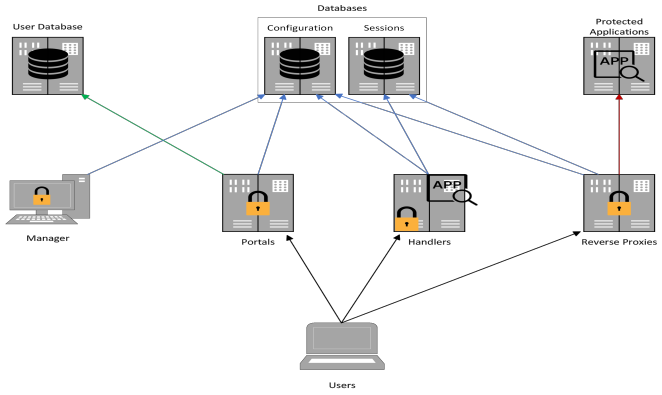

Fig. 1. Main components

Since 2004, GN is part of the LL::NG core team and a major contributor to this open source project. In 2016, the 1.9 release provides a new ANGULARJS Manager and supports the OpenId Connect protocol. The fully re-coded 2.0 version, released in 2018, provides a new plug-ins engine and supports Multi Factor Authentication (MFA) [3]. LL::NG 2.0.15-2, the last stable version that offers several new features, can be downloaded since September, 2022 from the OW2 official website [4].

LL::NG is mainly coded in `Perl` and `JavaScript` by the core development team members: XAVIER GUIMARD (GN/STSISI), CHRISTOPHE MAUDOUX (GN/STSISI & Cnam), CLÉMENT OUDOT (Worteks) and MAXIME BESSON (Worteks). 'Worteks' [5] is an expertise and publishing company in free and open source software. The 'STSISI' department's primary mission is to manage the technological modernization of the French 'Police Nationale' and 'Gendarmerie Nationale'.

Figure 2 describes the LL::NG development process that is composed of two different cycles. *(i)* The community cycle, hosted by the OW2 GitLab forge [6], is used for opening issues, Continuous Integration & Development (CI/CD) pipelines and milestones. *(ii)* Then, when a new version is released by the community every three months, this one is tested, validated, and deployed during the STSISI 1-year cycle.

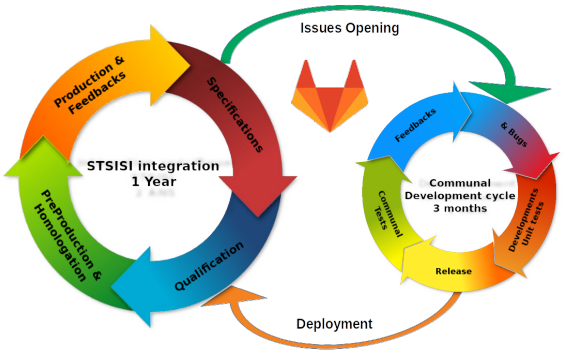

Fig. 2. Development process

B. Related Work

Some SSO solutions have been already implemented to protect data collection and server to server exchanges. Study in [7] describes how `Spring Security` and the `Keycloak` open-access platform can be employed to secure a microservices-based architecture Application Programming Interfaces (APIs). Embedded protection uses the `OAuth2` method, the underlying concept behind the OIDC protocol that is presented in Section IV-C. Problem here is that the `OAuth2` solution only consists in authenticating users. It can not be used for providing user's identity attributes to services unlike the OIDC protocol.

Some works have been conducted in [8] to deploy a Zero Trust Architecture (ZTA) using `Kubernetes` and a SSO product. As explained by the authors, a ZTA keeps checking the user's authenticity, monitors the user's devices, and checks for any location change initiated by the user device. Moreover, it also regularly checks for any discrepancies in the application that the user would be using. Main drawback here compared to *ServiceToken* handler explained in Section VI-A is that none authenticated logs are generated to track users activities.

III. USERS AUTHENTICATION

Many protocols or methods can be employed or combined to authenticate users. This section provides an overview of the main ones supported by LemonLDAP::NG.

A. Methods & Protocols

LL::NG provides several authentication methods and implements standard protocols. Users can authenticate by using network protocols like *Gnu Private Guard* (GPG) [9] that consists in asking user to sign a challenge and to post the result. *Remote Authentication Dial-In User Service* (Radius) is a client-server networking protocol that runs in the application layer [10]. LL::NG handles and forwards Radius authentication requests to the Radius server. With *Kerberos* [11], users are authenticated based on their desktop session. LL::NG validates the Kerberos ticket against a *local keytab*. LL::NG can also rely on an *LDAP* directory [12] to authenticate users and to get user attributes. LL::NG is compliant with LDAP v2 or v3 server, including *Active Directory* (AD) [13], and is compatible with LDAP password policy to check password strength, to block brute-force attacks or can force password change on first connection. LL::NG proposes *Pluggable Authentication Module* (PAM) as a simple authentication back-end. It is a mechanism to integrate multiple low-level authentication schemes into a high-level Application Programming Interface (API). PAM permits software that rely on authentication to be written independently of the underlying authentication scheme.

Connectors based on *OAuth2* protocol [14] are available like *Google*, *GitHub*, *Facebook*, *LinkedIn* or *Twitter* that allows applications to reuse its own authentication process [15].

Main authentication and standard SSO protocols like *CAS* or *SAMLv2* and *OIDC* described in Section IV are also implemented for authentication, authentication delegation or identities federation [16].

Furthermore, "combo" back-ends like *Authentication choice* and *Combination* can be employed to propose different authentication methods or to combine sequences.

B. Multi Factor Authentication

MFA is supported since the LL::NG 2.0 release. This feature can be activated for confirming a user's claimed identity by using a combination of two different factors [17] between:

- something you *know* (login/password, ...)
- something you *have* (U2F Key, TOTP, mail, ...)
- something you *are* (biometrics like fingerprints, ...)

LL::NG provides some second factor plug-ins that can be enabled for completing authentication module with MFA:

TOTP Time based One-Time Password is an algorithm that computes a one-time password from a shared secret key and current time [18]. Then, users might register a device as smartphone by using a dedicated application like FREEOTP or GOOGLE AUTHENTICATOR

U2F Universal 2nd Factor is an open authentication standard that enforces and simplifies two-factor authentication using specialized USB or NFC devices [19]. LL::NG can propose to users to register their key(s). Then, 2F registered users can not login without using their key(s)

Yubikey It is a hardware token manufactured by 'Yubico'. It sends an OTP, which is validated via Yubico server [20]

Mail After logging in via an authentication module, a one-time code is generated by the Portal and sent to the user's e-mail address. Then, user is prompted for this code in order to continue and to achieve the login process

External This method can be used to append a second factor authentication device like SMS, OTP, and so on. It calls external commands to send or to validate a second factor

REST/Radius Those methods rely on the corresponding protocol to submit and to check the second factor

IV. APPLICATIONS PROTECTION

Once users are authenticated, identity claims might be sent or provided to protected applications. Depending on applications and their capabilities different solutions can be used.

A. Central Authentication Service

CAS is an enterprise SSO solution and identity provider for web applications [21], [22]. It is an open and well-documented authentication protocol. When a client try to access a "CAS-sified" application which is an application requiring CAS authentication, the application redirects it to the CAS server. CAS validates the client's authenticity, usually by checking a username and password against a database.

Unlike protocols below, CAS requires to modify the application code to authenticate and retrieve the user data.

B. Security Assertion Markup Language version 2

SAMLv2/Shibboleth is an open standard used for authentication [23], [24]. Based upon the Extensible Markup Language (XML) format, web applications rely on SAML to transfer authentication data by using assertions (authentication request and response) between two parties – the Identity Provider (IdP) and an application known as the Service Provider (SP). SAMLv2 protocol requires an approbation link between the IdP and the SP. In fact, IdP and SP might exchange their respective metadata that permits to provide configuration parameters like end points and public keys.

SAMLv2 protocol does not require a direct link between IdP and SP. All network exchanges go through the browser that is not the case with OIDC.

C. OpenID Connect

OIDC is an evolution of SAMLv2 [25]. It employs REST protocol instead of SOAP to exchange assertions and JSON replaces XML to format messages. OIDC is easier to implement with mobile applications by using the refresh token mechanism.

OIDC standard defines 3 authentication flows. The *Hybrid* and *Implicit* flows are *deprecated* and should not be employed. The *Authorization Code* (AzC) flow is the most secured because user and applications are authenticated. The AzC is returned by the OP after user authentication. This code is provided to the application. This flow is used to prevent that the password is stolen by a malicious application. Then, it is provided with the *Client Id* and *Secret* to authenticate the application, and to retrieve *Id*, *Access* and *Refresh* tokens.

LL::NG implements all these standard SSO protocols. It can act as IdP/SP (SAML) or OP/RP (OIDC) and could be used as a "federation proxy" between all these protocols [26].

D. Handlers

Protection by handlers is a mechanism that can be employed to protect applications working with HTTP headers. Each request sent to the protected application is caught by the handler. It looks for a specific "token" depending on the used handler type. LL::NG proposes the following types:

Main SID is retrieved from the SSO cookie value
AuthBasic Performs Basic authentication [27]
SecureToken Decrypts SID from a ciphered SSO cookie
ServiceToken Retrieves SID from *X-LLNG-TOKEN* header
DevOps Fetches configuration from 'rules.json' file
OAuth2 Gets SID from an OAuth2 Access token
Zimbra Performs Zimbra pre-authentication [28]
CDA Used for Cross Domain Authentication [22], [29]

With the SID, handler collects user data from SSO sessions back-end, computes and checks access rules and sends HTTP headers with session attributes to the protected application.

Handler code can be embedded by the protected applications but it requires to have access to the application code and to modify it. To avoid this, applications can be hidden behind Reverse Proxies that embed the handler code. So, protected applications are not exposed to users and directly requested. Reverse Proxies-based architecture is described by Fig. 1.

Figure 3 details the Main handler kinematic: 1) User tries to access a protected application. His request is caught by Handler 2) SSO cookie is not detected, so Handler redirects user to Portal 3) User authenticates on Portal 4) Portal checks authentication 5) If authentication succeeds, Portal collects

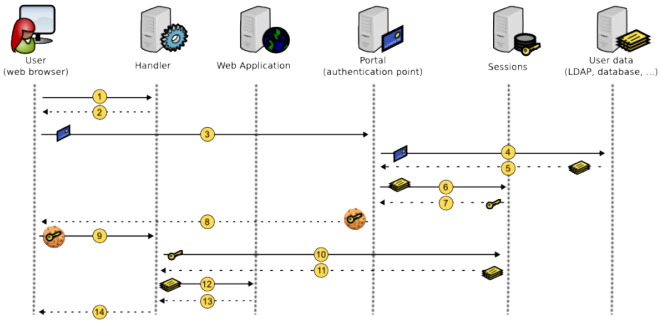

Fig. 3. Main Handler kinematic

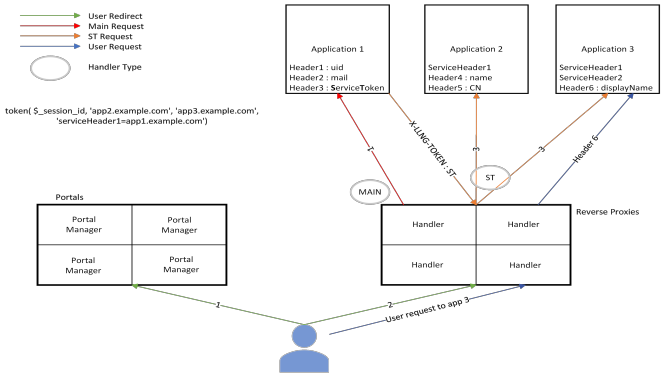

Fig. 4. ServiceToken handler kinematic

user data 6) Portal creates a session to store user data 7) Portal gets the SID 8) Portal creates a SSO cookie only valid on the SSO domain with the SID as value 9) User is redirected on the protected application with a SSO cookie 10) Handler gets SID from the cookie and retrieves user session data 11) Handler stores user data in its cache 12) Handler checks access rules and sends HTTP headers to the protected application 13) Protected application sends response to Handler 14) Handler forwards the response to user.

V. Extended Features

LemonLDAP::NG offers possibility to extend its features by hooking internal processes or appending functionalities.

A. Plug-ins Engine & Entry Points

Since 2.0 version, LL::NG provides a plug-ins engine. It allows to append specific or custom features. Plug-ins can handle authenticated or unauthenticated routes. They can also be launched by using particular entry points (EP) during login or logout process. This mechanism is a very interesting feature to extend LL::NG and to meet specific needs.

B. Specific Plug-ins

`AdaptativeAuthLevel` A user reaches an authentication level depending on the used authentication module, and eventually MFA as explained in Section III-B. This plug-in adapts the authentication level depending on other conditions, like network or IP address, device, and so on

`BruteForceProtection` To prevent brute force attack by blocking account after several login failures

`CheckDevOps` To validate 'rules.json' files (Section V)

`CheckUser` To check session attributes, access rights and transmitted headers for a particular user and URL

`ContextSwitching` Specific users like 'admins' can switch context other user for debugging purpose. Beginning and end of context switching process are logged

`CrowdSec` A free and open-source security automation tool leveraging local IP behaviour detection and a community-powered IP reputation system [30]

`Impersonation` Users can assume identity of another user for training or teaching purpose

`Notification` To notify messages to users when log in

`REST/SOAP` To provide the corresponding services

VI. Advanced Usages

LemonLDAP::NG implements all the previous features. But some specific needs can only be met by using advanced functionalities or mechanisms exposed below.

A. Server2Server Exchanges Protection

The *ServiceToken* handler is a mechanism to protect *Server2Server exchanges*. As described by Fig. 4, a web application (`WebApp1`) may need to request some other web applications (`WebAppN`) on behalf of the authenticated user. You just have to provide a header containing the *ServiceToken* (ST) to `WebApp1` and to protect `WebAppN` with the corresponding handler. Then `WebApp1` can request `WebAppN` by appending the 'X-LLNG-TOKEN' HTTP header with the ST as value that will be caught by the handler.

The ST is built by encrypting the SID and an authorized VHosts list to restricted its scope. The ST lifetime is limited in case of it is stolen. The ST can also be used for sending service headers to `WebAppN` like the source host by example.

B. SSO as a Service

The *DevOps* handler is a mechanism to implement the *SSO as a Service* (SSOaaS). '*aaS' means that application can drive underlying layer like IaaS for Infrastructure or PaaS for Platform. SSOaaS means provide the ability for an application to manage authorizations and choose user attributes to set. But authentication can not be really *aaS. This step must be performed by a well known, identified and managed service. It can not be delegated except with IdF protocols as described and explained in Section IV. Applications might just use it but not manage it. LL::NG offers some features that can be used for providing SSOaaS and implementing DevOps architectures.

DevOps concept means "development" and "operations". It is the combination of practices and tools designed to increase an organization's ability to deliver applications and services faster than traditional software development processes. This speed enables organizations agility to better serve their customers and compete more effectively in the market. DevOps breaks barriers between traditionally isolated development and operations teams. Under a DevOps model, development and operations

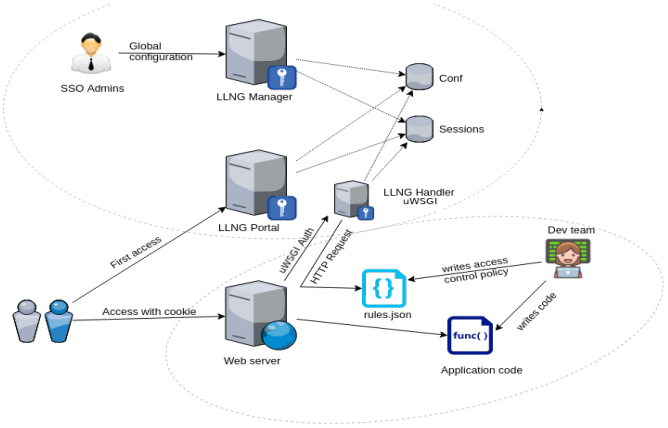

Fig. 5. DevOps handler kinematic

teams work together across the entire software application life cycle, from development and test through deployment to operations. It means in a SSO architecture that a web application can manage its own rules and headers regardless the SSO administration team.

The DevOps handler is an on-premises component (the LL::NG handler) designed to retrieve users data or SSO configuration from back-ends and VHost configuration (access rules and sent HTTP headers) not from LL:NG configuration but from the web application itself that can be hosted in a cloud environment. Rules and headers are set in a 'rules.json' file stored at the website root directory as described by Fig. 5. Communications between DevOps handler and web applications are based on a Common Gateway Interface (CGI) protocol like FastCGI or uwsgi. Rules.json files syntax and format can be checked by the DevOps teams with the `CheckDevOps` plug-in described in Section V.

VII. Conclusion

LemonLDAP::NG [31] is a free and open source full AAA SSO project that supports standard SSO protocols (SAMLv2, CAS, OIDC, ...), provides several features (MFA, different handlers or authentication methods) and services. It is a complete, versatile and customizable SSO solution. Its main specific and original features compared to the few other existing products are SSOaaS and Server2Server protection mechanisms. Thus and standard IdF protocols can be deployed and implemented in a cloud infrastructure to easily authenticate users, protect web applications and provide logs. It can also be employed as a proxy to interconnect different cloud platforms or between heterogeneous identity federation protocols.

Next release will offer a new Manager based on `ReactJS` framework, implement the `WebAuthn` protocol and a new MFA engine.

References

[1] X. Guimard, C. Oudot, C. Maudoux, and M. Besson, "LemonLDAP::NG," Dec. 2010. [Online]. Available: https://hal.inria.fr/hal-03776592

[2] "The uWSGI project — uWSGI 2.0 documentation." [Online]. Available: https://uwsgi-docs.readthedocs.io/en/latest/

[3] C. Maudoux, "Implémentation de l'authentification à double facteur dans la solution de SSO AAA LemonLDAP::NG 2.0," p. 170.

[4] "OW2 Projects - LemonLDAP::NG (lemonldap-ng.WebHome)." [Online]. Available: https://projects.ow2.org/view/lemonldap-ng/

[5] Worteks, "Worteks - Expertise Open Source." [Online]. Available: https://www.worteks.com/

[6] "GitLab OW2 LemonLDAP::NG." [Online]. Available: https://gitlab.ow2.org/lemonldap-ng

[7] A. Chatterjee and A. Prinz, "Applying Spring Security Framework with KeyCloak-Based OAuth2 to Protect Microservice Architecture APIs: A Case Study," *Sensors*, vol. 22, no. 5, p. 1703, Feb. 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/5/1703

[8] D. D'Silva and D. D. Ambawade, "Building A Zero Trust Architecture Using Kubernetes," in *2021 6th International Conference for Convergence in Technology (I2CT)*, Apr. 2021, pp. 1–8.

[9] "Index - GnuPG wiki." [Online]. Available: https://wiki.gnupg.org/

[10] "How Does RADIUS Work?" [Online]. Available: https://www.cisco.com/c/en/us/support/docs/security-vpn/remote-authentication-dial-user-service-radius/12433-32.html

[11] "Kerberos: The Network Authentication Protocol." [Online]. Available: https://web.mit.edu/kerberos/

[12] "OpenLDAP, Main Page." [Online]. Available: https://www.openldap.org/

[13] "Active Directory." [Online]. Available: https://www.cyberark.com/what-is/active-directory/

[14] "OAuth 2.0 — OAuth." [Online]. Available: https://oauth.net/2/

[15] K. Dodanduwa and I. Kaluthanthri, "Role of Trust in OAuth 2.0 and OpenID Connect," in *2018 IEEE International Conference on Information and Automation for Sustainability (ICIAfS)*, Dec. 2018, pp. 1–4.

[16] I. S. Sette and C. A. Ferraz, "Integrating Cloud Platforms to Identity Federations," in *2014 Brazilian Symposium on Computer Networks and Distributed Systems*, May 2014, pp. 310–318.

[17] E. Huseynov and J.-M. Seigneur, "Chapter 50 - Context-Aware Multifactor Authentication Survey," in *Computer and Information Security Handbook (Third Edition)*, J. R. Vacca, Ed. Boston: Morgan Kaufmann, Jan. 2017, pp. 715–726. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128038437000508

[18] "What are Time Based One Time Passwords (TOTP)? — Security Encyclopedia." [Online]. Available: https://www.hypr.com/security-encyclopedia/time-based-time-password-totp-otp

[19] "What is FIDO U2F?" [Online]. Available: https://www.yubico.com/authentication-standards/fido-u2f/

[20] "Let's get started with your YubiKey." [Online]. Available: https://www.yubico.com/gb/setup/

[21] "CAS - Home." [Online]. Available: https://apereo.github.io/cas/6.5.x

[22] H. Hu and Z. Guo, "The application of cross-domain single sign-on in municipal portal," in *2013 IEEE International Conference of IEEE Region 10 (TENCON 2013)*, Oct. 2013, pp. 1–4.

[23] "SAML Explained in Plain English — OneLogin." [Online]. Available: https://www.onelogin.com/learn/saml

[24] T. S. Sobh, "Identity management using SAML for mobile clients and Internet of Things," *Journal of High Speed Networks*, vol. 25, no. 1, pp. 101–126, Feb. 2019. [Online]. Available: https://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/JHS-190606

[25] "OpenID Connect explained." [Online]. Available: https://connect2id.com/learn/openid-connect

[26] "LL::NG as federation protocol proxy — LemonLDAP::NG 2.0 documentation." [Online]. Available: https://lemonldap-ng.org/documentation/latest/federationproxy.html?highlight=proxy

[27] "IBM Documentation," Mar. 2021. [Online]. Available: https://prod.ibmdocs-production-dal-6099123ce774e592a519d7c33db8265e-0000.us-south.containers.appdomain.cloud/docs/en/cics-ts/5.1?topic=concepts-http-basic-authentication

[28] "Preauth - Zimbra :: Tech Center." [Online]. Available: https://wiki.zimbra.com/wiki/Preauth

[29] "What Are Cross Domain Cookies and How Do They Work," Nov. 2018. [Online]. Available: http://limevpn.com/what-are-cross-domain-cookies-and-how-do-they-work

[30] "Firewall Bouncer — CrowdSec." [Online]. Available: https://docs.crowdsec.net/docs/bouncers/firewall/

[31] "LemonLDAP::NG - Web SSO and Access Management Free Software." [Online]. Available: https://lemonldap-ng.org/