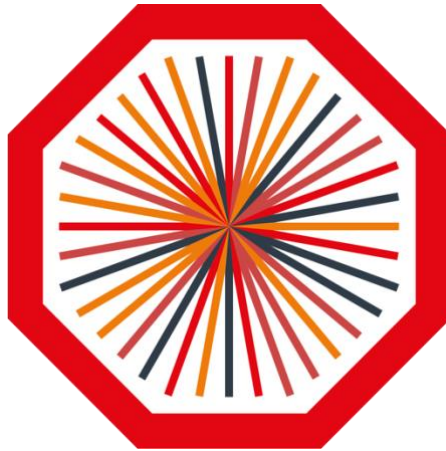


GATEkeeping: Governing Access To Endpoints of Bookkeeping

Emily Slade^{1*}

¹CERN, Geneva, Switzerland

Abstract. This report describes the implementation of access control for the ALICE Experiment at CERN's Large Hadron Collider, based on industry standards. Extensive role-based access control was implemented in the frontend and backend of the Bookkeeping application, and an improved developer UI was added for testing. Fully-configurable rate limiting middleware was added to the HTTP server, to be used in all of ALICE's GUIs. The existing application was upgraded with new features based on user feedback and bug testing.



* Corresponding author: emily_slade@outlook.com

Table of Contents

1 Overview	3
2 Introduction	3
3 ALICE	3
4 Project outline	4
4.1 <i>Creating the application</i>	4
4.2 <i>Deploying the application</i>	4
5 Gatekeeping	5
5.1 <i>Rate limiter</i>	5
5.1.1 Existing security	5
5.1.2 The need for rate limiting	5
5.1.3 Implementing a rate limiter	5
5.2 <i>Access control</i>	6
5.2.1 Frontend role-based access control	6
5.2.2 Specifying user scenarios	7
5.2.3 Reusable button component	7
5.2.4 Updating the local development user interface	8
5.3 <i>Backend role-based access control</i>	9
5.3.1 The need for backend access control	9
5.3.2 Middleware implementation	9
5.3.3 Mocha tests	9
5.3.4 ALICE members in charge of specific detectors	9
6 Feature Improvement	10
6.1 <i>Linking environments to logs</i>	10
6.2 <i>Displaying all detectors</i>	10
6.3 <i>Autofilling EoS reports</i>	10
6.4 <i>Updating Run Type filter</i>	11
6.5 <i>Combining EPN columns</i>	11
6.6 <i>Removing active filters header bar</i>	11
7 Bug fixes	11
7.1 <i>Initialise Collapse All button correctly</i>	11
7.2 <i>Remove bar overlap</i>	12
7.3 <i>Display ON for runs with no EPN number</i>	12
7.4 <i>Change colour of runs with no detector quality</i>	12
8 Acknowledgements	12
Bibliography	13

ALICE is investigating heavy-ion physics at the LHC. This experiment aims to recreate the conditions of the universe immediately after the Big Bang [9]. Quarks – fundamental particles

– and gluons – force carriers – are usually tightly bound together, but at the beginning of the universe they were mixed in an extremely hot, dense soup where they were only weakly bound with each other and were free to move around, in what is known as a quark-gluon plasma.

The LHC runs different types of particle collisions. One of these is a ‘heavy-ion run’ where it collides heavy ions, such as lead ions with protons (Pb-p) or lead ions with lead ions (Pb-Pb). ALICE has several detectors that encircle a section of the LHC ‘tube’. During the heavy-ion runs[†], the collisions are positioned (using magnets) to occur inside the ALICE detectors. The ALICE researchers then take measurements of the resulting collisions.

The collisions occur as part of a beam, or LHC fill, which is a period of several hours of successive particle collisions every 25ms. The researchers store information about the fills, such as the type of collision, the duration of the fill, and the efficiency of the fill.

During an LHC fill, the researchers at ALICE activate some of the detectors to take measurements, and this forms a run. There may be several runs during one LHC fill. The researchers also store information about the runs, such as the LHC fill it was active in, the quality of the run, which detectors were active, and the reason the run was ended.

The researchers work in one of three shifts: Morning (7am to 3pm), Afternoon (3pm to 11pm), or Night (11pm to 7am). During this time, researchers create log entries about other information during their shift, such as detector alarms that occurred, changes in detectors configuration that were made, and minutes of meetings that took place. They can use tags to group similar log entries together, such as ‘GLOBAL’, ‘EoS’, or ‘TEST’.

There are five different types of shift role: shift leader (SL), shift leader in matters of safety (SLIMOS), experiment control system (ECS), detector control system (DCS), and quality control/physics data processing (QC/PDP). Each shifter is responsible for producing an End of Shift (EoS) log entry as a handover document for the next shifter.

All of this information (log entries, end of shift reports, LHC fills, and runs) and more is stored in a single application: Bookkeeping.

4 Project outline

The work I completed consisted of: first, creating and integrating a role-based access control system into Bookkeeping as a self-directed open-ended project, and second, building and improving new and existing features for Bookkeeping as part of a team-directed and user-feedback-led project, with code reviews and pair programming.

4.1 Creating the application

- **JavaScript + HTML + CSS:** frontend
- **Express:** server
- **npm:** manage the JavaScript packages
- **NodeJS:** backend runtime
- **eslint:** format and style the code
- **Mocha + Puppeteer + Chai:** test the application
- **MariaDB:** store and query the database
- **Sequelize:** abstract database queries
- **Mithril:** build fast single-page applications

4.2 Deploying the application

- **Jira:** assign tasks
- **GitHub:** collaborate on the code
- **GitLab:** deploying continuous integration and continuous delivery (CI/CD)
- **Docker:** build and test the application
- **VS Code:** write and edit code

[†] The detectors also take measurements during proton-proton runs, to compare with heavy-ion runs

5 Gatekeeping

Governing Access To Endpoints of Bookkeeping (GATEkeeping) is a self-directed project about implementing role-based access control for Bookkeeping.

5.1 Rate limiter

5.1.1 Existing security

There is currently one Internet-facing HTTPS server implementation that is shared across all of ALICE's graphical user interfaces (GUIs) [13]. This is an express server. It limits the size of all request bodies to 100kb, but does not limit the number of requests made by a single endpoint. The Bookkeeping application is expected to be the most used application of ALICE's GUIs, with an estimated 100,000 requests every 2-3 seconds expected next year, so it is crucial that the server can meet this need for speed from legitimate endpoints.

CERN IT provides user authentication and authorisation through OpenID Connect, using the OAuth2.0 flow [14].

5.1.2 The need for rate limiting

The CIA triad – confidentiality, integrity, availability – constitutes the three essential properties of a cyber-secure application. Attacks from compromised users (as well as well-intentioned repeated requests from non-malicious users) pose a threat to the availability of an application. This project aims to mitigate the risks from DDoS attacks to ensure that the Bookkeeping application remains available.

The current implementation of the server leaves it open to distributed denial of service (DDoS) attacks, where an attacker remotely uses a network of endpoints to repeatedly send requests to an internet-facing server in order to overload it [15]. This prevents the server from responding to legitimate requests and renders the application unavailable.

5.1.3 Implementing a rate limiter

My aim was to implement a rate limiter on the server which blocks repeated requests from the same IP address in a given time window.

This project builds on CERN's authentication services, and aims to provide an extra layer of security for ALICE GUIs. Given the high volume of legitimate traffic, the rate limiter must be configurable on a per-application basis, where system administrators (who are aware of the expected traffic flow) can set an appropriate rate limit. To achieve this, I researched different rate limiters that could be used in conjunction with the Express server.

There are several rate limiters available as npm packages. The required features for the rate limiter are:

- **Lightweight and low overhead:** the package should be simple to use, and doesn't need complex features
- **High number of downloads:** to act as a proxy for quality
- **Able to rate limit on a per-IP basis:** to ensure that other users' functionality isn't impacted
- **Configurable:** so system administrators can choose the size and time period of the rate limit
- **Recently updated:** as an indicator of it being maintained and patched in the future

The top five results for rate limiting are p-limit, express-rate-limit, bottleneck, limiter, and express-brute. This table shows a summary of the different features of the five rate limiters:

Table 1 npm rate limiter packages and the number of dependents, number of weekly downloads, and the time since the publish

Package	Number of dependents	Weekly downloads	Last publish [‡]
p-limit	2445	98m	2 years ago
express-rate-limit	778	3m	13 hours ago
bottleneck	674	2m	4 years ago
limiter	329	5m	2 years ago
express-brute	86	17k	7 years ago

p-limit is used for limiting concurrency with promises, and it doesn't have an inbuilt option to rate limit on a per-IP address basis. Bottleneck, limiter, and express-brute were all last updated 2 or more years ago. Therefore express-rate-limit is the best option: this package has per-IP address rate limiting, is configurable, and is lightweight. I then implemented this in the server.

```
const { limit } = require('express-rate-limit')
const limiter = limit({
  windowMs: this.rate_limit_window,
  max: this.rate_limit,
  message: 'Too many requests. Please try again later',
});
this.app.use('/', limiter);
```

Code sample 1 adding rate limiting middleware to the server using the express-rate-limit package

The rate limit window (`this.rate_limit_window`) and the rate limit itself (`this.rate_limit`) are configurable in a separate configuration file, making it customisable for each application, as desired.

I added mocha tests for the limiter, and also built a graphical test page to test the rate limiter with. As expected, all requests from the same IP address reach the server correctly, and are responded to with '200 OK', until the limit is hit, at which point the server refuses any more requests from that endpoint. After the rate limit window has elapsed, the counter resets and the page can once more make successful requests to the server.

5.2 Access control

5.2.1 Frontend role-based access control

Access control is a method of mediating access to resources based on the user's identity [16]. There are four types of access control: role-based access control (RBAC) – which is based on the roles a user plays in an organization, attribute-based access control (ABAC) – which is based on contextual factors such as location and current organizational threat level – mandatory access control (MAC) – which is based on the sensitivity of the information in the resource – and discretionary access control (DAC) – where the owners of a resource set the permissions for it.

[‡] As of 12 September 2023

Access control is a key part of meeting information security management systems standards, such as ISO/IEC 27001 Annex A.9.4: System and application access control [17], which states that information systems should be protected from unwanted access.

5.2.2 Specifying user scenarios

ALICE has a set of roles, and users of the ALICE applications can hold one, none, or several of these roles. These include administrators, run coordinators, members of ALICE, and members of CERN. These roles do not form a strict hierarchy, and one role may intersect with (yet be neither a strict superset of or a subset of) another.

Bookkeeping uses CERN’s JWTs to verify the user’s identity. These tokens encode the user’s information (including their name, username, and access roles) in Base64, and create a keyed hash of this using HMAC SHA256 (HS256): a symmetric keyed hashing algorithm. JWTs are an open standard (RFC 7519) that provide information transmission with integrity (and can be encrypted to provide confidentiality) [18].

ALICE applications are not currently open to the public or the Internet, and this access control is the first step towards being able to safely make Bookkeeping available on the Internet.

I used Jira tickets to order, prioritise and keep track of my work. I first created user scenarios to describe how different roles should be able to interact with the application.

Table 2 an example selection of ALICE roles, with my proposed access rights

Role	What content is the user permitted to edit	Can the user view the controls for editing
Administrators	All content	Yes
ALICE members (with all permissions)	All content	Yes
ALICE members (in charge of specific detectors)	Content related to their detector	Yes
ALICE members (other)	No content	Yes
CERN members	No content	Yes
The general public	No content	No

5.2.3 Reusable button component

I created a reusable component that could display any button or link in one of three states: visible, disabled, and hidden. As outlined in the user scenarios, a user may be able to view, but not click, some of the editing buttons. I used the ALICE-themed CSS building blocks from the ALICE O² WebUI Framework [19] to style the buttons, to ensure the application look remained sleek and cohesive. This component is displayed in **Figure 2**.



Figure 2 From left to right: the button is hidden (not displayed, no special cursor), the button is visible but disabled (displayed in grey, inactive cursor, no href or onclick property), the button is visible and active (displayed in site theme colours, normal cursor, works and redirects as expected)

5.2.4 Updating the local development user interface

Another task was to update the user interface and allow developers to easily select multiple roles in order to be able to test these while developing. This new user interface would only be available in the local development environment. I updated the navigation bar at the top to split the existing settings menu into two: one for access roles and the other for additional frontend settings. I changed the icons to match their new role.

I added the list of currently active roles to the title in the top left of the screen so that developers can easily see which roles they currently have. I changed the access menu from a slider between ‘user’ and ‘admin’ to a dropdown checkbox list, to allow developers to select multiple roles. I added a toggle button for selecting ‘all’ and ‘none’ of the checkboxes, and an option to turn the ‘active roles’ title off. This interface is displayed in **Figure 3**.

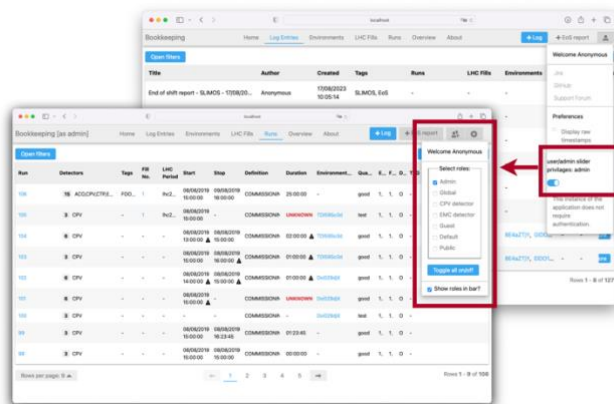


Figure 3 Updated developer interface for easily testing role-based access control

I converted all the buttons to this configurable component, meaning that future buttons can be added with minimal overhead, by separating the access from the component creation.

The final task was to ensure that users with detector-specific roles can only edit their own detectors when editing a run. If a user is permitted to edit all roles, this should override any detector-specific rights. If the user has only detector-specific rights, they should be able to *view* all detectors in a run, but only *edit* their own detectors. If a user has multiple detector-specific roles, the associated rights should be unioned together. See **Figure 4** for an example.

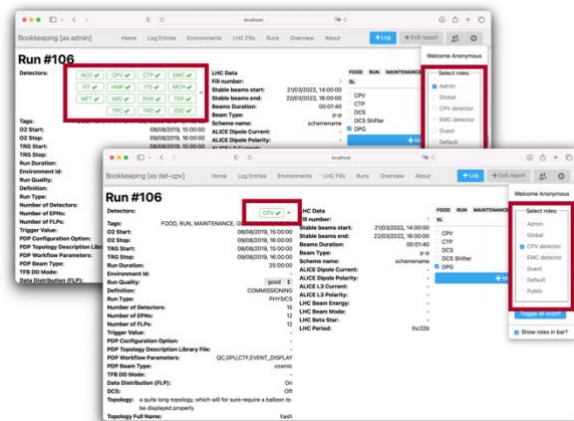


Figure 4 (Top left) users with admin access can edit all roles. (Bottom right) users with CPV detector access can only edit their own detectors

5.3 Backend role-based access control

5.3.1 The need for backend access control

Frontend access control is useful for visually indicating to a user what actions they can and can't perform, but it does not provide complete security. Even if links to a page are removed from the application, the user could still manually enter the URL and attempt to navigate to the hidden page. Even if a 'submit' button from a form is removed, the user can still attempt to send a maliciously-crafted packet to the server with a custom payload.

The best solution to this is to implement role-based access control within the server, so that even if an attacker can bypass the frontend interface, they are still prevented from interacting with the backend. This will also provide a failsafe against potential future misconfigurations of the application, if a developer accidentally sets a button to 'visible'.

5.3.2 Middleware implementation

I implemented middleware in the router, so that every HTTP request is parsed by access control middleware, and the requested action can be checked against the user's roles. The middleware takes as input a list of allowed roles, and the list of the user's roles (that are validated separately using the tokens and authentication described above), and checks whether (at least) one of the user's roles is part of the 'allowed roles' list.

5.3.3 Mocha tests

I tested the middleware with mocha tests.

```
describe('GET requests on public routes', () => {
  it('should return 200 with admin rights', async () => {
    const response = await request(server)
      .get('/api/tags')
      .send({
        access: ['admin'],
      });
    expect(response.status).to.equal(200);
  });
});
```

Code sample 2 mocha test to check that the middleware correctly passes the admin request to the server, which then responds with status 200: OK

5.3.4 ALICE members in charge of specific detectors

I needed to create specialized middleware to be able to handle requests to edit run detectors. Other routes depend only on the user's roles, but this route depended on both the user's roles and on which detectors the user was attempting to edit. I created separate middleware for the run-detector-edit route, which uses the same logic as the frontend run detector component, and checks whether the user has admin-like rights (and passes the request on to the server as normal), and then checks whether the user has a detector-specific role and is only attempting to edit their own detectors (again passing the request on to the server as normal), and if the user does not have the correct roles, or if they are attempting to edit any detector they are not allowed to, the middleware returns an error 403: Forbidden.

6 Feature Improvement

6.1 Linking environments to logs

Log entries currently consist of a title, a body, and optionally any associated run numbers, file attachments, and tags. My task was to provide functionality for linking environments to log entries as well. Environments are configurations of the detectors.

Logs are stored as one table in the database, and environments are stored as another. I created an associative table, LogEnvironments, to map environments to logs. I created an API route to fetch all environments associated with a given log ID, and updated the API route to allow filtering and sorting and to allow different Boolean operators ('and', 'or') in the environments request.

I then added the frontend components to allow the shifters to add environments to log entries, to view the environments in the Log Overview page, and to sort and filter by environments.

6.2 Displaying all detectors

Most runs only use a subset of the available ALICE detectors. Currently, the Run Details page only displays the detectors that were active during the run, colour-coded according to whether their quality was Good or Bad. The ALICE shifters provided feedback to our team that they wanted to have all the detectors displayed as well as the active ones, with inactive detectors displayed as greyed-out. I implemented this according to the users' specifications.

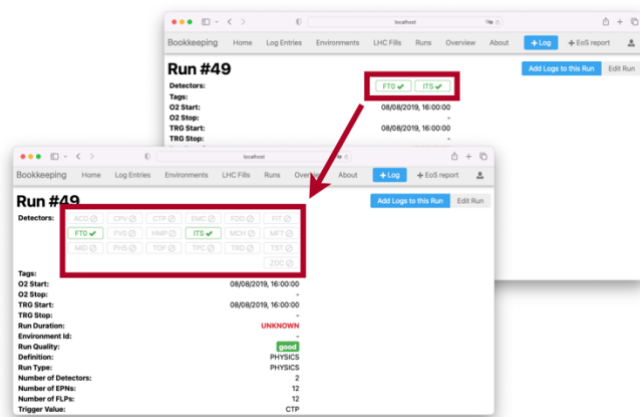


Figure 5 Run Details Page: detectors that were not active during the run are now displayed

6.3 Autofilling EoS reports

When a shifter finishes their shift, they complete an End of Shift (EoS) report. This is a summary of any issues that happened during their shift, as well as the current state of the experiments. The report contains, among others, two fields: 'From previous shifter' and 'For next shifter'. This contains the information that needs to be transferred to the next shift. When an EoS report is created, it is added as a log entry, and tagged with 'EoS'. EoS reports are specific to each shifter type (e.g. ECS, SL etc.).

Currently, the shifters search through the log entries to find the previous EoS report of their shift type, and then copy and paste this information into the new EoS report. My task

was to automate this by autofilling the 'From previous shifter' section in new EoS reports with the 'For next shifter' information from the previous EoS report. This involved:

- Create the expected title in the format 'End of shift report - <shift type> - <date> <shift period>'
- Search through log entries to find all logs with that exact title
- If there are no results or multiple results, return an error
- Extract the 'For next shifter' section using regular expressions
- Display the information or the error message on the EoS report creation page

6.4 Updating Run Type filter

Runs can be filtered by different properties, such as the run number, the detectors used in the run, and the type of run. This task involved simplifying the filter panel interface by converting the inline list of run types to a collapsible dropdown list of run types.

6.5 Combining EPN columns

Currently there are two columns related to the ALICE Event Processing Nodes (EPN): one column displays whether the EPNs are ON or OFF, and a second column displays the number of EPNs that were used. However, if the EPNs were OFF, the default value 10 is displayed in the 'number of EPNs' column, which can be confusing for the shifters. My task was to merge the two columns, and display either 'OFF' or the number of EPNs used.

6.6 Removing active filters header bar

The Log Overview page and the Run Overview page contain lists of items that can be filtered on various properties. The user can toggle the visibility of the filtering panel. To summarise the active filters when the filtering panel has been minimised, there is a header bar that appears that contains the names of the properties that are currently being filtered on, e.g. 'Active Filters: Title, Author, Created from'.

However, feedback from the shifters suggested that this information was not useful, and it would be easier to see the actual property values that are being filtered on. As this is a complex task (to summarise this information while not cluttering the interface) to implement in the future, my task was to remove this header bar.

7 Bug fixes

I discovered and raised several bugs while working on features of the application. I also received feedback that some of the code I had written didn't handle all edge cases (such as not accounting for database entries that don't exist), and I fixed and pushed the relevant bug fixes promptly.

7.1 Initialise Collapse All button correctly

The log entries are stored as trees, with a root and children. When the Log Details page is loaded, it displays all entries in the tree, but collapses the entries to save space. A button is available to toggle between 'Show all' log entries and 'Collapse all' log entries, as well as buttons to toggle each entry individually. However, the application automatically un-collapses single-log trees, so the button incorrectly reads 'Show all' on the initial load. I raised and fixed this bug.

```
const isSingleTree = this._logTreeLeafs.isSuccess()
  ? this._logTreeLeafs.payload.length === 1
  : false;
this._showAllPosts = isSingleTree;
```

Code sample 3 check if the log tree was loaded correctly and is a single tree, and display the corresponding button label

7.2 Remove bar overlap

When the active filters bar is displayed, it overlaps with the dropdown menu under ‘Overview’. I raised and fixed this bug.

7.3 Display ON for runs with no EPN number

Runs used to only contain information about whether EPNs were ON or OFF, and did not have an associated variable containing the number of EPNs. This caused incorrect values to be displayed when viewing these older runs. My task was to ensure the application checks if the EPNs were set to ON but have no ‘number of EPNs’ variable associated with them, and to display either ‘ON’ or the number.

```
nEpsns: {
  name: 'Number of EPNs',
  visible: true,
  format: (nEpsns, { epn }) => epn
    ? (typeof nEpsns === 'number' ? nEpsns : 'ON')
    : 'OFF',
}
```

Code sample 4 check if the nEpsns variable is set. Display the value if it is, and ‘ON’ if it is not

7.4 Change colour of runs with no detector quality

Runs used to be allowed to contain detectors that did not have a detector quality set. This means that when all detectors are displayed on the Run Details page, inactive detectors are displayed as the same greyed-out colour as detectors which were part of the run but did not have the quality set, which is misleading for the shifters. My task was to differentiate these two detectors with different shades of grey, to indicate that the detector was used but has no colour-coded quality associated with it.

8 Acknowledgements

I would like to express my gratitude to my supervisors, George Raduta and Martin Boulais, without whom I would not have been able to complete this project. I would also like to thank all the ALICE members and the wider CERN community for making me feel so welcome.

Bibliography

1. *Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC*. The ATLAS Collaboration. 2012, Physics Letters B. <https://doi.org/10.1016/j.physletb.2012.08.020>.
2. Berners-Lee, Tim. *Information Management: A Proposal*. 1989. <https://www.w3.org/History/1989/proposal.html>.
3. CERN. *The Large Hadron Collider*. <https://www.home.cern/science/accelerators/large-hadron-collider>.
4. CERN. *The Super Proton Synchrotron*. CERN. <https://home.cern/science/accelerators/super-proton-synchrotron>.
5. CERN. *The Antiproton Decelerator*. CERN. <https://www.home.cern/science/accelerators/antiproton-decelerator>.
6. Lopienska, Ewa. *The CERN accelerator complex, layout in 2022*. Complexe des accélérateurs du CERN en janvier 2022. 2022. <https://cds.cern.ch/record/2800984>.
7. CERN. *AWAKE*. CERN. <https://home.cern/science/accelerators/awake>.
8. CERN. *ALPHA*. CERN. <https://home.cern/science/experiments/alpha>.
9. CERN. *ALICE*. CERN. <https://www.home.cern/science/experiments/alice>.
10. CERN. *ATLAS*. CERN. <https://www.home.cern/science/experiments/atlas>.
11. CERN. *CMS*. CERN. <https://www.home.cern/science/experiments/cms>.
12. CERN. *LHCb*. CERN. <https://www.home.cern/science/experiments/lhcb>.
13. *From design to production: State-of-the-art web user interfaces to operate ALICE offline and online system*. Raduta, George. s.l. : IOP Publishing, February 2023, Journal of Physics: Conference Series. <https://dx.doi.org/10.1088/1742-6596/2438/1/012041>.
14. *CERN's Identity and Access Management: A journey to Open Source*. Corman, Asier, et al. November 2020, EPJ Web of Conferences. <https://doi.org/10.1051/epjconf/202024503012>.
15. CompTIA. *What Is a DDoS Attack and How Does It Work?* <https://www.comptia.org/content/guides/what-is-a-ddos-attack-how-it-works>.
16. Access Control. OWASP. https://owasp.org/www-community/Access_Control.
17. ISO/IEC 27001. ISO. November 2022. <https://www.iso.org/standard/27001>.
18. *Introduction to JSON Web Tokens*. JWT. <https://jwt.io/introduction>.
19. O2 WebUI. <https://github.com/AliceO2Group/WebUi/tree/dev/Framework>.