



Security Analysis of BigBlueButton and eduMEET

Nico Heitmann^(✉), Hendrik Siewert, Sven Moog, and Juraj Somorovsky

Paderborn University, Paderborn, Germany
heitnico@mail.upb.de

Abstract. Video conferencing systems have become an indispensable part of our world. Using video conferencing systems implies the expectation that online meetings run as smoothly as in-person meetings. Thus, online meetings need to be just as secure and private as in-person meetings, which are secured against disruptive factors and unauthorized persons by physical access control mechanisms.

To show the security dangers of conferencing systems and raise general awareness when using these technologies, we analyze the security of two widely used research and education open-source video conferencing systems: BigBlueButton and eduMEET. Because both systems are very different, we analyzed their architectures, considering the respective components with their main tasks, features, and user roles. In the following systematic security analyses, we found 50 vulnerabilities. These include broken access control, NoSQL injection, and denial of service (DoS). The vulnerabilities have root causes of different natures. While BigBlueButton has a lot of complexity due to many components, eduMEET, which is relatively young, focuses more on features than security. The sheer amount of results and the lack of prior work indicate a research gap that needs to be closed since video conferencing systems continue to play a significant role in research, education, and everyday life.

1 Introduction

The COVID-19 pandemic forced millions of people worldwide to stay at home [19]. As a result, meetings, classrooms, and private events were held online, and the demand and interest for online video conferencing systems increased [42]. In April 2020, Zoom Video Communications reported that their number of users had significantly increased due to the pandemic. While Zoom had 10 million daily users in December 2019, it had 200 million daily users in March 2020. Besides Zoom, there are numerous other video conferencing services, such as Microsoft Teams, Webex by Cisco, Google Meet, Skype, Zoho Meeting, BlueJeans, LifeSize, Whereby, and many others [30].

Security of Video Conferencing Systems. With the rise of video conferencing systems, security and privacy concerns grew. In April 2020, Google, SpaceX, and others banned Zoom over privacy concerns regarding its end-to-end encryption (E2EE) [22, 38, 45]. To eliminate vulnerabilities and increase security, several

video conferencing providers, such as Zoom Video Communications, Inc. (Zoom) and 8x8, Inc. (Jitsi Meet), started bug bounty programs [1, 48]. This measure bears fruit; Zoom received 401 reports and awarded \$1.8 million in 2021 [12].

The demand for security and privacy in conferencing technologies also led to open-source video conferencing systems gaining popularity. Open-source software allows one to analyze the source code and self-host conferencing servers, which requires know-how but has the advantage that data remains on known servers. This is especially important in deployments that need to comply with European regulations for the protection of personal data. For example, in 2022, Germany's federal state Rhineland-Palatinate forbade the usage of Microsoft Teams in schools because it is not compliant with the General Data Protection Regulation (GDPR) [16].

Towards Systematic Security Analysis of Video Conferencing Systems. Despite the importance of security and privacy in video conferencing systems, there is, to our knowledge, no systematic research on the security of video conferencing systems yet. To gain insights into the attack surface associated with video conferencing systems, we selected two open-source systems widely used in research and education: BigBlueButton and eduMEET.

BigBlueButton has been developed since 2007 with the goal of giving teachers and researchers the ability for a new style of hybrid teaching where BigBlueButton should serve as an online classroom [27]. Similarly to other video conferencing systems, BigBlueButton gained popularity during the COVID-19 pandemic. It has, for example, become the primary mode of communication and learning in schools in France [7]. The recommendation was issued by the French Ministry of Education, which is responsible for 65,000 schools serving 12 million students. Even after the pandemic, BigBlueButton remained the recommended education tool in France [5] and several German federal states [16, 46].

eduMEET was released in April 2020 by GÉANT, a European research network [15]. According to GÉANT, the release was rapidly accelerated due to lockdown measures and the need for an alternative and trustworthy video conferencing solution [26]. GÉANT's main arguments for having their video conferencing system were that it is from their community, self-hosted, and therefore the traffic stays within their network. Thus, they consider the tool trustworthy and cost-efficient compared to commercial alternatives [15].

To analyze the security of the chosen video conferencing systems, we must first understand how both systems are composed and which features they provide. This leads us to our research questions:

- RQ1 *Which architecture concepts do BigBlueButton and eduMEET follow?*
- RQ2 *What are the common features and user roles, and how are permissions assigned to individual features?*
- RQ3 *What types of attacks result from the given architecture, features, and user roles?*

Our Approach. To perform a systematic analysis of a video conferencing system, we need to know how it is structured. That includes its components, as well as their responsibilities and tasks. Therefore, we break down the complex structures

of each system to form shared components with their main tasks. Furthermore, we examine the connection between features, permissions, and user roles that are common to video conferencing systems. Using this information, we define our attacker model and use it to perform a source code analysis, for which we follow the data flow within and between components. Thus, we can check whether the components adhere to their responsibilities and correctly enforce user permissions as assigned by the user roles.

Results. Besides both being web video conferencing systems with similar user roles, the architectures of BigBlueButton and eduMEET differ drastically. BigBlueButton has many features with a very complex structure, while eduMEET is more minimal in comparison. Our architecture analyses laid the groundwork for the systematic security analyses of both conferencing systems; we found 7 vulnerabilities and 7 bugs. Among them are classic security flaws like broken access control, NoSQL injection, and DoS, but also vulnerabilities that are feature-specific and could be detected due to our in-depth architecture analyses.

Contributions. In summary, we make the following contributions:

- We provide a structured security analysis of two modern open-source video conferencing systems: BigBlueButton and eduMEET.
- We present a common structure of both systems and introduce their main components, features, and user roles.
- With our security analyses, we were able to identify 57 vulnerabilities and bugs. These range from attacks targeting confidential meeting chats, participant lists, and streams to impersonation and DoS attacks.

Responsible Disclosure. We responsibly disclosed all vulnerabilities and bugs to the developers of BigBlueButton and eduMEET.

2 Background

In this section, we cover WebRTC, which both analyzed video conferencing systems use as their method for real-time audio and video transfer.

2.1 WebRTC

WebRTC [8] is a suite of protocols for real-time communication (RTC) over the Internet. For web applications, it defines a JavaScript API to access media devices and to manage WebRTC connections. WebRTC supports media streams and message-based transfer of arbitrary data. It supports peer-to-peer (P2P) connections, where two users exchange data directly, without the data flowing over a server.

Before two peers can establish a direct WebRTC connection, they need to exchange information using a signaling server. They negotiate the initial media streams, with configuration such as codecs and bitrate, in the form of a Session

Description Protocol (SDP) offer and answer [6]. These also contain the information needed for opening the direct connection, including NAT traversal (Interactive Connectivity Establishment (ICE) candidates). Once a direct connection has been established, the peers can transfer the negotiated streams. WebRTC currently supports only one media transport protocol, DTLS-SRTP [25].

2.2 WebRTC Architectures in Conferencing Systems

In a typical conferencing setting, a group of users exchanges media data, for example, audio and camera streams. Using a P2P architecture for broadcasting media streams minimizes latency and avoids server bandwidth overhead. However, this approach does not scale well due to the limited bandwidth of end users. Therefore, using a P2P architecture is often infeasible for conferences.

Instead of using a P2P architecture, conferencing systems implement servers that can receive and redistribute the media streams for each user. There are two types of architectures a WebRTC server can follow. In the Selective Forwarding Unit (SFU) architecture, the server distributes incoming streams unmodified. If the server processes and combines incoming media streams, the architecture is called Multipoint Control Unit (MCU). This lowers the bandwidth requirements for the clients in exchange for processing on the server.

3 Analysis Method

Due to the lack of systematic analyses of video conferencing systems, there is no methodology for us to use and adapt. Thus, we started developing an approach that was further refined during our analysis. The structure of our paper reflects the steps of our analysis.

In the first part of this section, we outline the procedure for analyzing the architecture and user roles of the chosen systems. The second part of this section deals with the structured source code analysis. We assume attackers can reach a conferencing server over the network, with the server operator being a trustworthy party. The source code analysis requires a detailed attacker model based on the architecture, so we defer the detailed attacker model to Sect. 6.

3.1 High-Level Analysis

In the primary analysis, we use the respective documentation and the publicly available source code to get a broad overview of the respective system. Getting an overview helps to assess the complexity of the system, as well as understanding the functionality and use case for the video conferencing system (e.g., education). We divide the primary analysis into the following steps.

Architecture and Components (RQ1). The first step contains the architecture of the respective system (see Sect. 4). Next to the main components of the architectures, such as web client and server, we are especially interested in the WebRTC components and the messaging between components since these aspects facilitate the understanding of the systems the most.

Conferencing Features (RQ2). Then, we look at the features that each system offers (see Sect. 5.1). The features are needed for our analysis because each feature interacts with a meeting in a certain way (e.g., removing users from a meeting). These interactions are mostly limited to certain groups of users, such as moderators, and therefore require access control.

User Roles (RQ2). Finally, we check the user roles and permissions (see Sect. 5.2). We map these to the features that we gathered in the previous step. This allows us to get an overview and understanding of the system, which facilitates a more detailed source code analysis.

3.2 Source Code Supported Security Analysis

In our source code analysis, we chose a manual approach since automation does not work in our case (see Sect. 8.3). As the first step, we perform a detailed analysis of the implementation. This shares commonalities with the primary architecture analysis, but we now focus on the internal implementation of each component. We manually validate that the implementation matches the documentation and our understanding of the features. This step also results in the identification of internal assumptions, for example, which parts of internal messages the components treat as trustworthy. All components have the responsibility to satisfy these often implicit internal assumptions.

Because almost all server logic gets triggered by user actions, we perform a data flow analysis on each possible user action. We confirm the overall behavior in practice, for example, via the browser's developer tools. During this data flow analysis, we consider the responsibilities of each component (e.g., access control on the conferencing server). Whenever it is not certain that an aspect of the conferencing system correctly adheres to the responsibilities, we need to investigate further.

When investigating a potential vulnerability, we may move directly to building a proof of concept. Otherwise, we may also re-evaluate whether it is handled elsewhere than expected. In either case, we conclude when we have either demonstrated an exploit or have complete reasoning for the behavior to be correct.

4 Architectures of the Analyzed Open-Source Conferencing Systems (RQ1)

We answer **RQ1** by analyzing the architectures of BigBlueButton and eduMEET. From both systems, we first derive a shared architecture that gives a high-level overview of common components and outlines their main tasks, which not only helps understanding the analyzed video conferencing systems, but also might facilitate future work. Then, we describe the implementation specifics of BigBlueButton and eduMEET. Finally, we compare the feature sets they offer to users.

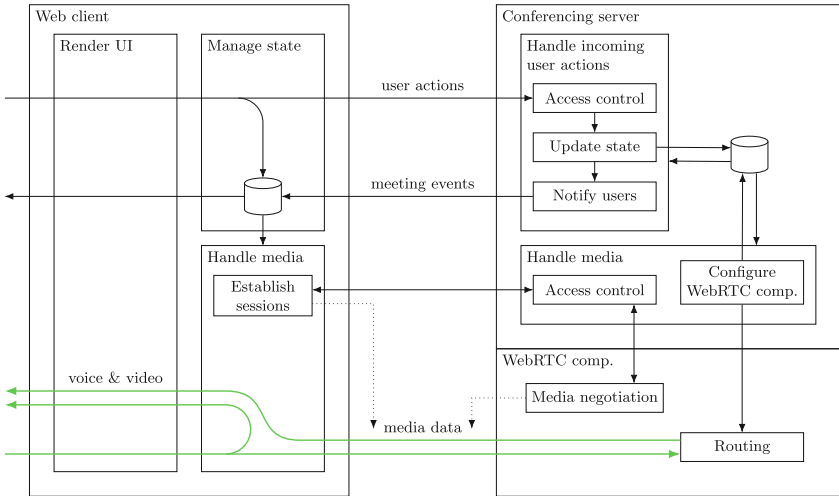


Fig. 1. Shared architecture of the analyzed video conferencing systems, showing the three components “web client”, “conferencing server”, and the “WebRTC component” with their main tasks. Arrows represent communication, and media streams are marked in green. The dotted arrows mark the creation of the WebRTC connection. The cylinders represent data storage.

4.1 Shared Architecture

We first focus on commonalities of the analyzed video conferencing systems by deliberately abstracting from specific features of BigBlueButton and eduMEET. This results in the architecture of a video conferencing system with minimal features. Figure 1 shows a summary of the components of such a video conferencing system. In the following, we describe the main components of the shared architecture. Then, we describe how each analyzed system implements each component with its uniqueness.

Web Client. The web client is responsible for three main tasks. The first main task, closest to the user, is rendering the user interface (UI). The UI allows users to interact with the meeting. The web client updates the UI in response to interactions triggered both by the local user and actions by other users in the meeting. Such actions include a user enabling their camera or sending a chat message. The web client is also responsible for displaying the conferencing system’s features, such as video chat. The features depend on the conferencing system (see Sect. 5.1).

The second main task of the web client is handling media streams. This includes establishing a WebRTC connection (see Sect. 2.1), where one peer is the web client and the other peer is the WebRTC component. Once a streaming session has been established, the peers can start sending and receiving media

data. On the client side, incoming media streams are connected to the UI, where the videos are displayed. The client also displays its outgoing video streams.

The third main task is processing and sending meeting state updates. When a user performs actions in the UI, the client sends *user actions*, i. e., intended changes to the meeting state, to the server. If the user's intended change is valid, the server notifies the web clients of the changes to the meeting state, which we refer to as an *event*. When a web client receives an event, it processes it and updates the local state in near real-time. Possible events include receiving new chat messages, changes to permissions, muting audio, or starting and stopping a video. The possible events depend on the features of the conferencing system.

Server-Side Components. The analyzed conferencing systems consist of two server-side components: the conferencing server and a WebRTC component.

Conferencing Server. The first task of the conferencing server is processing incoming user actions. This involves three main steps. First, the server performs access control by checking whether the user may perform the requested user action. Second, if the action is valid, the conferencing server executes it. This may involve additional processing by the server and results in changes to the meeting state. Finally, the server publishes events to the clients.

The second task of the conferencing server is managing streaming sessions. For this purpose, it controls the WebRTC component, which may be an external media server or embedded in the conferencing system as a library. The conferencing server participates in establishing streaming sessions by creating them in the WebRTC component and providing communication between the client and WebRTC component for the initial negotiation. The conferencing server is responsible for access control by mediating the initial communication. After the negotiation, the WebRTC component and client establish a direct communication channel, and the conferencing server can no longer mediate. If the permissions of a user get revoked, the conferencing server is responsible for closing streaming sessions via the WebRTC component's management interface.

WebRTC Component. Finally, there is a WebRTC component with loose coupling to the conferencing server. The WebRTC component relies on commands by the conferencing server for management and has the task to establish streaming sessions. The second task of the WebRTC component is to route media streams. The conferencing systems covered here use the SFU architecture for all video streams, so the server redistributes media streams unmodified (see Sect. 2.2).

4.2 Implementation of BigBlueButton

In this section we show how BigBlueButton's components implement their tasks.

Web Client. BigBlueButton uses the frontend framework React¹ for its UI. React does not provide any communication between the server and client. The server and client of BigBlueButton use the web framework Meteor.js to facilitate communication, which provides remote procedure call (RPC) and publish/subscribe capabilities. Internally, if possible, Meteor.js uses a WebSocket for communication. Using the publish/subscribe capabilities of Meteor.js, the client mirrors the meeting state of the server and receives state changes triggered by user actions. Therefore, the web client of BigBlueButton only needs to perform limited state management.

Server-Side Components. The server side of BigBlueButton is split into a conferencing server and two standalone servers for WebRTC.

Conferencing Server. The conferencing server of BigBlueButton is internally split into several individual components. It receives user actions on the WebSocket connection provided by Meteor.js and routes them within the conferencing server. At the destination, a handler performs access control checks and updates the meeting state. These updates to the meeting state are propagated internally. The conferencing server keeps a copy of the meeting state in a MongoDB database and uses the publish/subscribe mechanism of Meteor.js to pass change events to the clients, with access control in the publishing step.

For managing media streams, BigBlueButton interacts with its two media servers. The web client has the initiative to open media streams for its outgoing and incoming streams. For the audio conference, BigBlueButton does not mediate signaling between the client and the WebRTC component but instead relies on the client's knowledge of a five-digit voice conference number for access control. For video streams, the server performs a permission check when clients want to open a stream. When clients get removed from the meeting, the server component reacts to the event by closing their video streams.

WebRTC Component. BigBlueButton 2.3.3, the version analyzed here, uses two media servers: FreeSWITCH, and Kurento Media Server (Kurento).² The voice conference of meetings is handled by FreeSWITCH, with clients directly connecting to FreeSWITCH to perform media negotiation. The video streams are handled by Kurento, with the conferencing server mediating media negotiation. BigBlueButton also uses Kurento to relay the voice conference to participants who only listen. The conferencing server communicates with both media servers for access control and necessary configuration, for example, media routing.

Extensions to the Shared Architecture. BigBlueButton does not provide user management but instead relies on external software to integrate BigBlue-

¹ <https://reactjs.org/>.

² BigBlueButton 2.4 introduces the media server mediasoup, which replaces Kurento as the default as of 2.5, with a modified media topology.

Button’s meeting functionality, for example, Greenlight³ or Moodle.⁴ The 3rd-party application performs authentication and access control for joining meetings. For this purpose, BigBlueButton provides a custom HTTP management API. A shared secret between BigBlueButton and 3rd-party server applications controls access to the API.

For processing uploaded presentation slides, BigBlueButton uses several external programs, depending on the file type. The resulting files are made available to other clients from disk. BigBlueButton allows users to record the meetings. If a meeting is recorded, it stores audio and video recordings and the internal messages of the entire meeting as files. BigBlueButton embeds an instance of the collaborative text editor Etherpad⁵ to implement its shared notes.

The media negotiation between the client and FreeSWITCH extends our general model as it is not mediated by the conferencing server. This allows server operators to connect FreeSWITCH to an external telephony provider via Session Initiation Protocol (SIP), allowing users to join the conference by telephone.

4.3 Implementation of eduMEET

Web Client. For its web client, eduMEET uses the frontend framework React. For maintaining the meeting state on the client, eduMEET uses Redux,⁶ a JavaScript library for state management in web applications. It uses a store that holds the application state. The application state gets updated when user actions or events are dispatched. User-triggered actions can either modify the meeting room (e. g., locking the room) or change the user settings. The client may pass these user actions to the server using a WebSocket. To establish and manage a WebRTC streaming session, the client uses the mediasoup client library.

Server-Side Components. The server side of eduMEET is split into a conferencing server, which consists of an Express⁷ web server with WebSocket support, and the Node.js library mediasoup for WebRTC.

Conferencing Server. The conferencing server handles all incoming connections and user authorization. Because of the WebSocket support in the Express web server, WebSocket handlers and HTTP request handlers share access to a session object for all requests from the same client. The WebSocket are attached to a *peer* object representing a user. The peer object contains relevant information such as user roles, a unique peer ID, a room ID, and a socket. Any modification to a peer object is done via the peer ID, which references the peer object in a dictionary.

³ <https://github.com/bigbluebutton/greenlight>.

⁴ <https://moodle.org/>.

⁵ <https://etherpad.org/>.

⁶ <https://redux.js.org/>.

⁷ <https://expressjs.com/>.

WebRTC Component. For media handling on the server, eduMEET uses the Node.js library mediasoup, a layer of JavaScript that communicates with a set of C/C++ subprocesses. The internal architecture of mediasoup has its own terminology, which contains workers, routers, transports, producers, and consumers [11]. When a new user joins, the client and the conferencing server create a producer instance. The conferencing server then notifies the other peers and creates a consumer instance for each. The notified peers create local consumer instances for themselves.

Extensions to the Shared Architecture. The first additional component in eduMEET is a torrent tracker for its file sharing feature (see Sect. 5.1). It keeps track of users participating in upload and download, helping users to connect to each other. In the web client, eduMEET uses the WebTorrent⁸ library, which uses WebRTC for peer-to-peer communication. Furthermore, eduMEET uses the Passport⁹ module for external authentication strategies. Depending on the authentication strategy, new components might arise, for example, an Identity Provider for OpenID Connect (OIDC) [29].

5 Features and User Roles (RQ2)

In this section, we first compare features the analyzed conferencing systems offer. We then present user roles shared by both analyzed conferencing systems. Finally, we go into detail on how each of the analyzed video conferencing systems handles user roles, permissions, and the mapping to features.

5.1 Comparison of Features

Table 1 shows an overview of the features of both systems. While there is some overlap, there are also several features specific to BigBlueButton or eduMEET. Features specific to BigBlueButton are, for example, polls or shared notes. On the other hand, eduMEET offers file sharing, which is not implemented in BigBlueButton.

Some features require additional libraries or application logic. Other features require extending the conferencing system with new components, which are either controlled by the server operator or an external entity. Components can be additional servers or important libraries that play a vital role in the video conferencing system (e. g., mediasoup in eduMEET). A component controlled by the server operator is, for example, a WebRTC media server. A torrent tracking server for file sharing would be an example of a component that is controlled by an external entity (see Sect. 4.3). Importantly, additional features and components introduce a new level of complexity and a broader attack surface.

⁸ <https://webtorrent.io/>.

⁹ <https://www.passportjs.org/>.

Table 1. Conferencing features supported by BigBlueButton and eduMEET, with their required roles. Several features are present both in BigBlueButton and eduMEET, while others are only supported by one. The table lists which role a user needs to actively use a feature, where the role “everyone” includes users without access to the meeting. Note that some features are accessible by multiple user groups.

Feature	BigBlueButton		eduMEET	
	support	roles	support	roles
Voice conference	●	\mathcal{V}	●	\mathcal{V}
Video conference	●	\mathcal{V}	●	\mathcal{V}
Screen sharing	●	\mathcal{P}	●	\mathcal{V}
Text chat	●	\mathcal{V}	●	\mathcal{V}
Role management	●	\mathcal{M}	●	\mathcal{M}^*
External authentication	◐	Ω	◐	Ω
Waiting room	●	\mathcal{M}	●	$\mathcal{V}^*, \mathcal{M}$
Breakout rooms	●	\mathcal{M}	○	–
Hand raise	●	\mathcal{V}	●	\mathcal{V}
Shared notes	●	\mathcal{V}	○	–
Polls	●	\mathcal{P}	○	–
Text captions	●	\mathcal{M}	○	–
File sharing	○	–	●	\mathcal{V}
Slide upload	●	\mathcal{P}	○	–
Whiteboard	●	$\mathcal{V}^*, \mathcal{P}$	○	–
Recordings	●	\mathcal{M}	●	\mathcal{V}
Telephone dial-in	◐	Ω^*	○	–

○ not supported ● supported ◐ requires additional configuration
 Ω everyone * additional requirements \mathcal{V} viewers \mathcal{M} moderators \mathcal{P} presenters

5.2 User Roles

Common User Roles. BigBlueButton and eduMEET use user roles combined with permissions for their access control; users who participate in meetings have different roles, which give them permission to access or use certain features. Such permissions allow users to share their audio or video, or give users access to moderation features.

The analyzed conferencing systems have two main user roles in common: “viewer” and “moderator”. The viewer role, also referred to as “normal” in eduMEET, gives users basic permissions and allows them, for example, to send and receive media streams. The moderator role allows for managing the meeting room, the users, and access to other features. Furthermore, depending on the features of the respective conferencing system, we can differentiate between users in a waiting room (or lobby) and users in a meeting. Oftentimes, restrictions like this are not implemented by creating new user roles, but rather using properties or flags that are part of the user objects. Thus, two viewers might have different permissions or access to different features. For example, one user with the viewer role might be in a waiting room and cannot receive audio and video streams from other users, while other users with the same role do not have these restrictions because they are already in the meeting.

Because user roles and the associated permissions are heavily influenced by features and the current meeting state, access control is a complex topic. The following sections explain the details of each analyzed conferencing system. Table 1

gives an overview of the requirements to access individual features. Some features have additional requirements besides the user role; for example, regular users in BigBlueButton may only draw on the whiteboard if the presenter has given them permission.

BigBlueButton-Specific User Roles. In addition to the viewer and moderator role, every meeting has at most one presenter, who gets permissions related to a presentation area in the meeting. These additional permissions are limited to the presenter; other users, including moderators, cannot affect the presentation area.

Permissions may depend on context. Within breakout rooms, there is no distinction between moderators and viewers, and all users can interact with the meeting as viewers. BigBlueButton has a guest waiting room, allowing moderators to limit access to the meeting until they approve new users. Users calling in via telephone do not have access to the web interface of BigBlueButton and thus only have access to a very limited set of user actions.

BigBlueButton also allows moderators to “lock” viewers and presenters, to take away specific permissions. One may use this to aid in the moderation of large meetings or for specific use cases, like online exams, where participants should not see each other.

eduMEET-Specific User Roles. In eduMEET, one can use a configuration file to define new roles and to assign specific permissions. This also permits changing existing assignments of roles and permissions. The default configuration contains the roles “normal” (here referred to as “viewer”), “moderator”, and “admin”. A user can have multiple roles.

A moderator can kick users, disable audio, video, and screen sharing for users (which the user can activate again), take down raised hands, clear the chat, and end the meeting. Furthermore, a moderator can use the role manager to give and remove roles during a meeting. Each role has a “promotable” flag, which determines whether moderators can give and take the role. In addition, each role has a configurable level. The level of a moderator must be at least equal to the role of the target user to modify the target user role. The admin role, which has the highest level, allows users to enter a full room or a locked room, which normally sends users to the room’s lobby to wait for approval. As long as no moderator is in the meeting, viewers can also lock and unlock the room. The permission is revoked as soon as a user with the moderator role joins.

6 Attacker Model

After performing the primary analysis, as mentioned in Sect. 3, we developed an attacker model that fits the setting of a video conferencing system.

We assume an attacker may send arbitrary network requests. They do not have access to any private information regarding the server or the users. The

attacker cannot read or interfere with the network traffic of other users. The server operator is assumed to be entirely trustworthy. We do not impose conditions on the surrounding situation because conferencing systems are used to host various types of events. The attacker may be a viewer, presenter, or even moderator in a meeting. The attacker may also be a non-participant with no roles at all or be in the waiting room. The attacker may create their own meetings. During a meeting, users' roles may change, so we also consider cases where a moderator revokes an attacker's permissions.

We consider an attack successful if the attacker breaks any of the aspects of the CIA triad. The attacker breaks confidentiality guarantees if, for example, they join a locked meeting and retrieve sensitive streams or public chat content. The integrity of the meeting state is broken when an attacker oversteps the permissions of their role, by performing any action that modifies the meeting state in a way that they are not allowed to. This includes an attacker joining a meeting without permission. For availability, we consider an attack successful if the attacker performs DoS against any feature in any meeting, affecting any user other than the attacker themselves. We exclude DoS by resource exhaustion and only consider cases of DoS in the application logic, for example, an attacker blocking seats in a meeting.

We limit the scope of our analysis to the first-party code of BigBlueButton and eduMEET, respectively. External components and libraries are out of scope and thus deemed to be safe for the purpose of our evaluation. They are expected to conform to their documentation with configuration files as distributed with the conferencing systems. With our analysis, we target the server-side code because it implements the main application logic. For eduMEET, however, all clients take an active role in maintaining the meeting state, so we consider both the server and client side of eduMEET. BigBlueButton relies on the external framework Meteor.js to maintain the client state, which is out of the scope of this analysis. Since the server operator is fully trustworthy, we assume that additional configurations made by the server operator are secure.¹⁰

7 Evaluation (RQ3)

We performed the evaluation on BigBlueButton 2.3.3 and eduMEET 3.5.0-beta.1, the most recent versions at the time of analysis.¹¹ Because our attacker model is relatively broad, we identified not only high-impact vulnerabilities but also several smaller vulnerabilities without significant impact on the meeting. To not overestimate their impact, we explicitly classify such vulnerabilities as “bugs”. Hereafter, we refer to vulnerabilities and bugs as “findings”.

Our evaluation resulted in 45 findings in BigBlueButton (38 vulnerabilities and 7 bugs) and 12 findings in eduMEET (12 vulnerabilities). Table 2 gives a


¹⁰ This mainly applies to eduMEET, as it allows flexible configuration of the user roles and permissions.

¹¹ We analyzed commit `bb46e2d` of <https://github.com/edumeet/edumeet> (branch “develop”), which was later merged into eduMEET 3.5.0-beta.1.

Table 2. Summary of all findings in BigBlueButton (BXX) and in eduMEET (EXX). The final two columns denote which role a legitimate user needs to access the feature, and the role an attacker needs to perform the attack. The role “everyone” includes users without access to the meeting.

Description	Type (CIA)	Feature (see Table 1)	Roles	
			Legit.	Attacker
B1 Read other meetings' public chat	C		-	Ω^*
B2 Read arbitrary private chats	C		-	Ω^*
B3 Write into chat when lock setting changes	I		\mathcal{V}^*	\mathcal{V}
B4 Retain access to shared notes after leaving	CIA		-	Ω^*
B5 Bypass read-only shared notes	I		\mathcal{M}	\mathcal{V}
B6 Impersonate users in shared notes	I		-	\mathcal{V}
B7 Write captions (mechanism 1)	I		\mathcal{M}	\mathcal{V}
B8 Write captions (mechanism 2)	I		\mathcal{M}	\mathcal{V}
B9 View individual poll responses	C		\mathcal{P}	\mathcal{V}
B10 View free-form answers before poll ends	C		\mathcal{P}	\mathcal{V}
B11 Bug: Submit multiple free-form poll answers	I		-	\mathcal{V}
B12 View names of all waiting guests	C		\mathcal{M}	\mathcal{V}
B13 Get confidential meeting info as waiting guest	C		-	Ω^*
B14 Block seats in the meeting	A		-	\mathcal{V}
B15 Bypass participant limit when joining	CI		\mathcal{M}	\mathcal{V}^*
B16 View user list while locked	C		\mathcal{V}^*	\mathcal{V}
B17 Read information about previous users	C		Ω^*	\mathcal{V}
B18 Bypass getting banned	CIA		-	\mathcal{V}
B19 Receive meeting state updates after leaving	C		-	Ω^*
B20 Receive list of all users on the server	C		-	Ω^*
B21 Bug: Add emoji status to other users	I		-	\mathcal{M}
B22 Impersonate users in breakout rooms	I		-	\mathcal{V}^*
B23 Access all breakout rooms	CI		\mathcal{M}	\mathcal{V}^*
B24 Run moderator actions in breakout rooms	IA		-	\mathcal{V}^*
B25 Bug: Access breakout room data after demotion	C		-	\mathcal{V}^*
B26 View unshown slides in current meeting	C		-	\mathcal{V}
B27 View slides in other meetings	C		-	Ω^*
B28 Give others access to the whiteboard	I		\mathcal{P}	\mathcal{V}^*
B29 Run whiteboard actions after losing access	IA		-	\mathcal{V}^*
B30 Add pencil annotations to whiteboard	I		$\mathcal{P}, \mathcal{V}^*$	\mathcal{V}
B31 Impersonate users towards FreeSWITCH	CIA		-	Ω^*
B32 Impersonate users towards bbb-webrtc-sfu	CIA		-	Ω^*
B33 Access meeting voice conferences	CIA		\mathcal{V}	Ω^*
B34 Receive audio and screen share after leaving	C		-	Ω^*
B35 Bypass locked webcam viewing (mechanism 1)	C		\mathcal{M}	\mathcal{V}
B36 Bypass locked webcam viewing (mechanism 2)	CI		\mathcal{M}	\mathcal{V}
B37 Bypass locked webcam sharing	IA		\mathcal{M}	\mathcal{V}
B38 Manipulate shared camera	CI		-	\mathcal{V}^*
B38.1 Activate participants' webcams	CI		-	\mathcal{V}^*
B38.2 View webcams in other meetings	C		-	Ω^*
B39 Replace others' webcam streams	I		-	Ω^*
B40 Abort running screen shares	A		\mathcal{P}, \mathcal{M}	Ω^*
B41 Bug: Receive metadata on shared videos	C		\mathcal{V}	Ω^*
B42 Access recordings by meeting ID	C		Ω^*	Ω^*
B43 Bug: Exclude own camera from recording	I		-	\mathcal{V}
B44 Bug: See limited camera streams (via recordings)	C		\mathcal{M}	Ω^*
B45 Bug: See contents of shared notes (via recordings)	C		\mathcal{V}, Ω^*	Ω^*
E1 Forge malicious chat objects	IA		-	\mathcal{V}
E1.1 Write chat message as a different user	I		-	\mathcal{V}
E1.2 Send chat message in the past or future	I		-	\mathcal{V}
E1.3 DoS meeting via chat message	A		-	\mathcal{V}
E1.4 Manipulate link display via markup	I		-	\mathcal{V}
E2 Rejoin after kick, bypassing locked room	CIA		\mathcal{V}	\mathcal{V}
E3 Overwrite peer reference	CIA		-	\mathcal{V}
E3.1 Participate in meeting invisibly	CI		-	\mathcal{V}
E3.2 Being untargetable by moderator actions	CI		-	\mathcal{V}
E3.3 Block new media streams for victim	IA		-	\mathcal{V}
E4 DoS meeting with overlong display names	A		-	\mathcal{V}
E5 Hijack WebSockets cross-site	CI		-	Ω
E6 DoS meeting by manipulating the peer ID	A		-	\mathcal{V}
E7 Inject arbitrary data in logs	I		-	\mathcal{V}
E8 Impersonate via display name and picture	I		-	\mathcal{V}
E9 Bypass permission checks after role change	I		-	Ω^*
E10 Prevent getting muted	IA		\mathcal{V}	\mathcal{V}
E11 Client DoS via inconsistent media config	A		-	\mathcal{V}
E12 Read shared files	C		\mathcal{V}	Ω^*

C: confidentiality I: integrity A: availability vulnerability bug
 - not a feature Ω everyone * additional requirements \mathcal{V} viewer \mathcal{M} moderators \mathcal{P} presenters

short description of each finding, provides the type of violation of the CIA triad (see Sect. 6), and assigns to each finding the features it affects (see Table 1). A finding may affect multiple features because some features share parts of their implementation, for example, the voice and video conference. If a finding is not related to any specific feature but the core implementation for meeting state and communication of the respective conferencing system, we use . The last two columns of Table 2 show the user roles needed to perform the actions associated with the findings. Some of these actions are available as features to specific user roles, shown in the column *legitimate roles*, while others are not intended to be accessible.

As can be seen in Table 2, most of the findings in BigBlueButton are in the core implementation and in the video conferencing feature. The rest of the findings are distributed across the other features. In eduMEET, most of the findings are also in the core implementation and in the text chat feature.

7.1 BigBlueButton

In this section, we present five representative findings out of the 45 findings in BigBlueButton.

B1: Read Other Meetings' Public Chat. This finding allows an attacker to access sensitive data from other meetings hosted on the same server.

To transfer chat messages from the conferencing server to the web client, BigBlueButton uses the publish/subscribe mechanism of Meteor.js (see Sect. 4.2). In particular, the client subscribes to a publisher called *group-chat-msg*, which always publishes public chat messages in their meeting and messages in private chats. The client establishes the subscription with a WebSocket message to the server. Listing 1.1 shows how the server restricts its responses in the publisher. In this query, the server inserts the `meetingId` of the meeting. The first branch only matches messages where the `chatId` value is set to "MAIN-PUBLIC-GROUP-CHAT", which means that the client is subscribed to the public messages in their particular meeting. The second branch matches all messages with a chat ID in the `chatsIds` array, which is a parameter sent by the client. However, missing validation of `chatsIds` resulted in the fact that the server can leak public chats of *every* meeting hosted on the server.

For the attack description, we assume an attacker who participates in any meeting hosted on a BigBlueButton server. The attacker has access to the public chat in their particular meeting. Using a modified client or browser developer tools, the attacker can modify the parameters their client sends to the server for the subscription. If the attacker adds "MAIN-PUBLIC-GROUP-CHAT" into the `chatsIds` list, intended for private chats (see Listing 1.2), their clients' subscription applies to the public chat of every meeting hosted by the server; the publisher on the server provides the messages from the public chats of all meetings. The attacker thus gains access to all messages from the public chat of every meeting hosted on the particular server.

```

[
  { "meetingId": meetingId,
    "chatId": "MAIN-PUBLIC-GROUP-CHAT"
  },
  { "chatId":
    { "$in": chatsIds }
  }
]

```

```

[
  { "meetingId": meetingId,
    "chatId": "MAIN-PUBLIC-GROUP-CHAT"
  },
  { "chatId":
    { "$in": [ "MAIN-PUBLIC-GROUP-CHAT" ] }
  }
]

```

Listing 1.1. The publisher’s server-side MongoDB query (simplified). When a chat message matches either one of the two branches, the server publishes it to the client.

Listing 1.2. By adding the string "MAIN-PUBLIC-GROUP-CHAT" to the list intended for private chats, an attacker subscribes to public chat messages of all meetings on the server.

B2: Read Arbitrary Private Chats. This finding interacts with B1, increasing the impact of this finding. The publisher *group-chat-msg* is also vulnerable to NoSQL injection. The parameter `chatsIds` can contain arbitrary values supported by EJSON, an extension of JSON used by Meteor.js. The server does not check the value’s type. An attacker can modify the parameters their client sends to the server like in the previous attack. In particular, the attacker can set the publisher’s parameter `chatsIds` to [`./.*`], causing it to provide all messages from all public and private chats in all meetings on the server.¹²

B4: Retain Full Access to Shared Notes After Leaving. This finding affects shared notes. BigBlueButton relies on the external server component Etherpad for shared notes. Thus, the conferencing server needs to ensure access control, including revoking access when a user loses access to the meeting. For this, the conferencing server includes checks when users make HTTP requests to Etherpad which reject all users without a BigBlueButton session and check whether the Etherpad pad belongs to the meeting that the user is in.

However, there is an issue with this process as Etherpad uses a long-running WebSocket connection for communication between the server and client. When a user leaves or gets kicked from a meeting, the conferencing server cannot close the WebSocket to Etherpad; an attacker can continue reading and editing the shared notes. In addition, the session used for the server-side check stays valid after leaving the meeting, so the server also allows new WebSocket connections to Etherpad.

B26: View Unshown Presentation Slides in Current Meeting. BigBlueButton relies on a client’s knowledge of a presentation ID for the client to download presentation slides for each uploaded presentation. However, the server leaks the presentation IDs.

¹² This is an array, containing a native JavaScript regular expression object for `.*`. In EJSON, it is serialized as [{ `"$regex": ".*", "$flags": ""` }].

The server sends the presentation IDs to clients so they can display the slides, but inadvertently reveals them for all presentations in the meeting due to incomplete filtering. This allows an attacker in the meeting to view all slides that have been uploaded, including future slides in the currently chosen presentation and the slides of presentations that were uploaded but never shown to the viewers.

B34: Receive Audio and Screen Share After Leaving. The final finding described here affects the voice and video conference and the screen share feature of BigBlueButton. It allows the attacker to listen to audio and watch screen shares secretly, even after they leave the meeting.

We assume the attacker is a viewer in a meeting and leaves or gets kicked. While still in the meeting, the attacker can open multiple viewing sessions for each media stream with a modified client that sends additional requests. For the screen share and listen-only audio, the conferencing server only closes one of the sessions when the attacker leaves the meeting. The remaining sessions stay valid in the WebRTC component and only get closed when the screen share or meeting ends, respectively.

7.2 eduMEET

We explain three representative findings in eduMEET. In Sect. A.1, we cover an additional interesting yet more complex finding in depth (E3).

E1: Forge Malicious Chat Objects. This finding points to one of the root causes of several findings in eduMEET and allows a multitude of attacks. A client can send a chat message to the server as a chat message object in its WebSocket connection to the server. The server forwards this object to all other participants in the same room as long as the sender has the `SEND_CHAT` permission. An attacker can manipulate fields in messages they send to perform several attacks. In the following, we describe three possible attacks. First, the attacker can manipulate the `name` field, which is used to display the name of the sender. The attacker can abuse it to impersonate other users by changing the content. Second, the attacker can also manipulate the `time` field, allowing them to manipulate the chat conversation and send messages in the past or future. Third, the attacker can also set the `name` field to `null` or other invalid objects. This leads to a DoS attack against the receiving clients because the client does not expect other data types, and the errors are not handled, which leads to a crash in the application. Interestingly, when users affected by such a DoS attack try to rejoin the meeting, they are usually redirected to the index page instead of joining the meeting. This happens because joining users receive the chat and file history, which automatically repeats the attack. The attacker can stay in the room to prevent the room from resetting, effectively blocking the room indefinitely.

E2: Rejoin After Kick, Bypassing Locked Room. This finding allows an attacker to bypass the room lock, which can be used as a security mechanism to prevent other users from joining the room without approval. In this attack scenario, a moderator kicks the attacker from the room. Afterward, the moderator locks the room, which prevents users from joining the room without approval. The attacker is now not able to rejoin the meeting without further actions because the client generates a new peer ID and the server prevents new users from joining a locked room. However, the attacker can set their client's peer ID to any value, for example, by overwriting the client-generated value with the browser developer tools. If the attacker sets their peer ID to their old peer ID when they were in the meeting, the server treats the attacker as a returning user, which allows bypassing a locked or even a full room. Therefore, the attacker can rejoin the locked room after getting kicked by changing the peer ID to the old peer ID.

E10: Prevent Getting Muted. This finding allows an attacker in a meeting to disrupt it without others being able to mute them. Moderators can mute participants for everyone (global mute). The affected user can still unmute themselves, so this is not a security mechanism. Participants can also decide to mute another participant for themselves (local mute). However, an attacker can circumvent getting muted by sending a request to create a second microphone producer and muting their first microphone producer. Other participants cannot globally or locally mute the attacker's second microphone producer.

7.3 Responsible Disclosure

We reproduced all findings on unmodified instances of BigBlueButton 2.3.3 and eduMEET 3.5.0-beta.1. We worked in local environments to not affect real video conferencing deployments with their users. We reported the findings to the developers of BigBlueButton and eduMEET between July 2021 and May 2022. The developers of eduMEET thanked us for the findings but have not released fixed versions as of December 2023. The developers of BigBlueButton acknowledged the findings and started publishing fixes with BigBlueButton 2.3.9. As of December 2023, the developers have fully addressed 37 of the 45 findings and assigned CVEs to 14 of them (see Table 3). The remaining issues are still to be fixed.

8 Discussion

We discuss our findings from Sect. 7 by considering the potential root causes in the respective conferencing system. For this, we identify commonalities between the findings. Finally, we discuss the limitations of our evaluation.

8.1 BigBlueButton

BigBlueButton offers a lot of features, making it the more complex of the two conferencing systems. Because of this breadth of features, the attack surface is

naturally larger when compared to eduMEET. In addition, the interplay between features makes correct implementation more difficult. We observed that our findings in BigBlueButton have two major types of root causes, both of which relate to the complexity of the software.

Several vulnerabilities came up as a result of subtle logic bugs in the internal server logic. We can see this in the situation arising when an attacker opens multiple media streams B35, but also in several other findings: B1, B3, B10, and B18, among others. These can, to some extent, be traced back to the internal logical complexity of BigBlueButton, which results from a large set of features and evolution over time.

For several other vulnerabilities, one can see a commonality of incomplete or missing security considerations in the design. For example, in B33, the ability of an attacker to join voice conferences without legitimate access can be traced back to reliance on a 5-digit voice conference ID for access control. When users leave, the server cannot revoke this ID to revoke access, as it is identical for all participants. In this case, there is a mitigation in place, but it is not sufficient to prevent attacks. There are also some more subtle cases, for example, in B27, where guessable secrets allow an attacker to gain read access to uploaded slides.

8.2 eduMEET

The root causes in most of our findings for eduMEET, are of a different nature. Oftentimes, the server trusts the client and forwards its messages without properly checking the input, for example, in E1, E4, and E6.

While the technical details of the other findings differ, they may stem from a similar root cause. For example, E2, which results from an implementation error, can also be seen as a missing feature because the moderator cannot effectively ban the attacker from the meeting. The same applies to E10 where the moderator cannot force an attacker to stop sharing audio. Here, it would be helpful to have a more fine-grained permission system, like the “lock settings” feature in BigBlueButton. This feature could allow the moderator to withdraw permissions of viewers, for example, to share audio.

In summary, most findings in eduMEET are either because there is too much trust in the client or because of missing moderation features. Both factors result in a lack of security and measures to eliminate disruptive factors within a meeting. Consequently, these findings show that filtering client messages and moderation features are critical measures to ensure secure meetings.

8.3 Limitations

Scope. To understand the architecture and behavior of conferencing systems, we analyzed the functionality and interaction of both conferencing systems. We examined server-side and client-side components in eduMEET. In BigBlueButton, we concentrated on the server-side components since these implement most

of the logic and functionality. BigBlueButton’s client delegates state management to the third-party framework Meteor.js, which is out of scope for our analysis. For this reason, we did not examine the BigBlueButton client, which could bring new findings regarding web security.

Automation. For comparison with our manual approach, we used SonarCloud¹³ to scan for bugs automatically. While it found code snippets that could be improved, it did not find any vulnerabilities. This result is expected because most of the bugs can be classified as logical flaws and require user interactions and a certain meeting state. Such conditions cannot be automatically applied by a static code analysis tool. BigBlueButton has publicly used SonarCloud as part of their quality control since June 2021.¹⁴

Architecture of Conferencing Systems. Comparing two architectures as different as those of eduMEET and BigBlueButton was not a trivial task. Thus, we agreed on a shared architecture by breaking down the architecture of the respective conference systems. Certainly, the shared architecture can be used for future work. However, depending on the conferencing system and architecture, it may be necessary to extend the model. Our model uses the SFU WebRTC architecture, while other systems may use P2P or other WebRTC architectures, which allow for direct communication between the clients. Furthermore, other conferencing systems may communicate differently, for example, by using Extensible Messaging and Presence Protocol (XMPP).

Analysis of Further Conferencing Systems. We limited the scope of our analysis to allow us to cover the chosen conferencing systems and their architectures in detail. Further analyses of open-source conferencing systems may be performed using a similar process, applied to their respective architectures. Our analysis process is not directly applicable to closed-source software. Nevertheless, the detected logical flaws can provide inspiration for new vulnerabilities in other closed-source conferencing systems supporting the affected features.

9 Related Work

Although various vulnerabilities have been found in web conferencing systems in the past, there is little exhaustive scientific research in the general area of video conferencing systems. Thus, we consider previous research, vulnerability reports, and talks regarding conferencing systems to get a grasp of the attack surface.

Most of the vulnerabilities found in web conferencing systems are related to classic web security vulnerabilities such as cross-site scripting (XSS) [43, 44], server-side request forgery (SSRF) [9], SQL injection via custom URI

¹³ <https://www.sonarsource.com/products/sonarcloud>.

¹⁴ <https://github.com/bigbluebutton/bigbluebutton/pull/10737#issuecomment-860211455>.

scheme [18], and different types of misconfigurations [40,41]. Also common are vulnerabilities resulting from missing checks [40,41], flawed role management [4,40,47], missing security considerations [2,47], and image or document conversions leading to vulnerabilities [2,10]. While all these vulnerabilities are interesting, we wanted to focus more on factors that extend our attack surface.

Among the previously mentioned reports, some stand out in particular because the described vulnerabilities are located in the client, but the client differs from our architecture. In our architecture, the client is a web browser. In some reports [3,20,28,43], the client is an Electron¹⁵ app. These applications are made with web technologies and use Chromium and Node.js. Vulnerabilities in these applications are critical since they can lead to client-side remote code execution (RCE) [3,20,28,43]. Other kinds of conferencing clients are classic executables on Windows, Mac, or Linux, which extend the attack surface as well, for example, due to memory-related issues [31,37]. Thus, different types of clients introduce different types of attacks, and the more types of clients the conferencing system offers, the larger the attack surface. The same applies to other components, such as Zoom’s Multimedia Router (MMR), which is responsible for transmitting audio and video between Zoom clients; this component was affected by a buffer overflow found by Google Project Zero [37].

Another interesting component used in conferencing systems is the login mechanism. Sudhodanan and Paverd found an attack related to Single Sign-On (SSO), where an attacker creates a Zoom account with the victim’s email (before the victim creates an account) [39]. When the victim now uses an identity provider with the same email to create a Zoom account, Zoom merges the accounts, which allows the attacker to log in to the victim’s account with the attacker’s password.

Natalie Silvanovich from Google Project Zero released articles in 2018, where she analyzed and fuzzed the WebRTC implementation in Chrome and closed-source video conferencing applications such as FaceTime and WhatsApp [32–36]. Four years later, she found one memory-related vulnerability in Zoom’s client and another one in Zoom’s MMR [37]. In the end, she pointed out that the closed-source software comes with a lot of challenges for researchers, which prevents further progress in verifying security properties [37]. She recommended making closed-source software available to security researchers [37]. In the same year, Ivan Fratric from Google Project Zero presented at Black Hat USA a 0-click RCE vulnerability in Zoom [14]. Fratric found out that different components use different XML parsers, which allowed him to smuggle XMPP messages (stanza smuggling) [14].

In the last years, cryptographic vulnerabilities in Matrix clients and libraries became public [13,24]. In 2021, Kasak et al. drew attention to two vulnerabilities where vulnerable clients may be tricked into disclosing encryption keys [13]. In 2022, Albrecht et al. presented six attacks that affected the Matrix standard and its flagship client Element [24]. These attacks break authentication and confidentiality but require the cooperation of the homeserver, which is responsible

¹⁵ <https://www.electronjs.org/>.

for storing communication history and account information and relaying messages [23,24].

While we mentioned vulnerabilities in conferencing systems from a technical point of view, Ling et al. focused on the attacker as a person who is responsible for disruptions in a meeting, i. e., Zoombombing [21]. Their results indicate that such attackers often have help from an insider within the meeting. Therefore, password protections and meeting IDs are a rather ineffective mechanism to prevent Zoombombing; they argue that unique join links would be an effective security mechanism.

In summary, there are lots of reports and findings in different fields regarding video conferencing systems and their components. However, there is a gap in scientific approaches, especially regarding open-source video conferencing systems. Our work is a first step to approach this problem.

10 Conclusions and Future Work

In our work, we systematically analyzed two open-source conferencing systems and detected 57 vulnerabilities and bugs. While the root cause for vulnerabilities in BigBlueButton mostly lies in the complexity of the system and the interplay between the features, in eduMEET, they mainly resulted from missing strict authorization checks and excessive trust in client messages. We want to highlight that our findings do not imply that BigBlueButton and eduMEET are less secure than commercial closed-source alternatives. The high number of findings was largely enabled by the open-source implementations, which facilitated our in-depth evaluations. On the negative side, it needs to be mentioned that both systems lack a swift vulnerability patching process. In the case of eduMEET, none of the reported vulnerabilities have been fixed. This is not acceptable for systems processing security-critical data.

The high number of findings shows that there is indeed a research gap in the security of video conferencing systems. With our systematic security analyses, we want to draw attention to this topic and want to stress that video conferencing systems offer a large attack surface due to their large number of components and used technologies. This is also confirmed by many related vulnerabilities, mostly found in non-systematic analyses by bug bounty hunters in recent years.

Our work can be extended in different directions. XML parsers within XMPP implementations are underexplored and are an interesting attack vector since XMPP is often used in video conferencing systems [14]. Other than that, the systematic approach that we applied to BigBlueButton and eduMEET could be applied to other open-source conferencing systems. Closed-source software is often more difficult to analyze if it is not freely and openly available [37]. Commercial providers should consider facilitating further security research and we hope there will be more future work that helps to improve the security of video conferencing systems.

Acknowledgements. We thank our anonymous reviewers for their insightful comments and detailed suggestions. We are also grateful to Sven Hebrok for helpful discussions and contributions in the early stage of this research.

This research was funded by the PRISMA Elite Program of the Department of Computer Science of Paderborn University, and by the research project “North Rhine-Westphalian Experts in Research on Digitalization (NERD II)”, sponsored by the state of North Rhine-Westphalia – NERD II 005-2201-0014.

A Appendix

A.1 eduMEET

E3: Overwrite Peer Reference. This finding leads to multiple high-impact issues in eduMEET. When a user connects to a room, the server creates a JSON Web Token (JWT) [17], bound to a peer ID generated by the client. The JWT is stored on the server and referenced by a cookie-based session. When a user connects to a room, eduMEET performs two checks. First, the server checks if there is a JWT for the session and if the peer ID is already used by a connected user. The server rejects the connecting peer if the peer ID is already used, but no JWT exists. Otherwise, the server verifies the JWT, i. e., the server checks if the peer ID matches the JWT. In case of a valid JWT, the server treats the user as returning and closes the old connection.

The vulnerability stems from the fact that the server creates a new peer object regardless of whether the JWT matches the peer ID. The peer object (see Sect. 4.3) represents a meeting participant and contains, among other things, a unique peer ID, a room ID, a list of user roles, and a socket for communication. Despite the importance of the JWT validity, in the first step, the server only checks if a peer ID and a JWT exist. Therefore, the peer ID does not have to match the JWT. When an attacker connects to a room with an existing peer ID and a valid session referencing a JWT that does not match the peer ID, the server still creates a new peer object. The server keeps a list of peer objects referenced by their peer IDs. Since the peer ID exists, the server overwrites the existing peer reference. However, the old connection stays open, and the old peer can still participate in the meeting.

To perform an attack, the attacker first joins any meeting. The attacker has a valid peer ID generated by the client and a valid session from the server. The attacker now joins the targeted meeting. Instead of having their client generate a new random peer ID for this connection, the attacker chooses an existing peer ID (see below). To achieve this, they may use the browser developer tools to overwrite the JavaScript variable holding the peer ID their client generated and then join the meeting. The new peer ID does not match the attacker’s JWT anymore. The WebSocket connection of the old peer with the same peer ID is not closed but not referenced anymore. That means the old peer cannot be targeted by moderator actions (E3.2). If the new peer with the same peer ID now leaves the meeting, the UI of all participants changes, and peers with the target peer ID are not visible anymore, but the connection of the old peer is still

open. The old peer is invisible in the meeting but can still participate (E3.1), for example, listen to existing streams and read the chat. To exploit the first two scenarios (E3.1 and E3.2), the attacker needs to be the old and new peer at the same time, which they can achieve using two browser instances. In the last scenario (E3.3), the attacker uses the victim’s peer ID to prevent the victim from receiving new media streams. The peer IDs of all participants are known because the server broadcasts them when joining.

A.2 Status of Fixes in BigBlueButton

Table 3. All findings in BigBlueButton (BXX) with their status as of December 2023. The latest version at this time was BigBlueButton 2.7.3. In total, 37 of the 45 findings were fully fixed and the impact of one additional finding was significantly reduced.

Description	Fixed?	CVE
B1 Read other meetings’ public chat	●	CVE-2022-29232
B2 Read arbitrary private chats	●	(none)
B3 Write into chat when lock setting changes	●	CVE-2022-29234
B4 Retain access to shared notes after leaving	●	(none)
B5 Bypass read-only shared notes	●	(none)
B6 Impersonate users in shared notes	●	(none)
B7 Write captions (mechanism 1)	●	(none)
B8 Write captions (mechanism 2)	●	CVE-2022-29231
B9 View individual poll responses	●	CVE-2022-23490
B10 View free-form answers before poll ends	●	(none)
B11 Bug: Submit multiple free-form poll answers	●	(none)
B12 View names of all waiting guests	●	(none)
B13 Get confidential meeting info as waiting guest	●	(none)
B14 Block seats in the meeting	●	(none)
B15 Bypass participant limit when joining	●	(none)
B16 View user list while locked	○	(none)
B17 Read information about previous users	●	(none)
B18 Bypass getting banned	●	CVE-2022-41961
B19 Receive meeting state updates after leaving	●	CVE-2022-41959
B20 Receive list of all users on the server	●	(none)
B21 Bug: Add emoji status to other users	●	CVE-2022-41962
B22 Impersonate users in breakout rooms	●	(none)
B23 Access all breakout rooms	●	CVE-2022-29233
B24 Run moderator actions in breakout rooms	●	(none)
B25 Bug: Access breakout room data after demotion	●	(none)
B26 View unshown slides in current meeting	○	(none)
B27 View slides in other meetings	○	(none)
B28 Give others access to the whiteboard	●	(none)
B29 Run whiteboard actions after losing access	●	CVE-2022-41963
B30 Add pencil annotations to whiteboard	●	CVE-2022-29236
B31 Impersonate users towards FreeSWITCH	○	(none)
B32 Impersonate users towards bbb-webrtc-sfu	●	(none)
B33 Access meeting voice conferences	○	(none)
B34 Receive audio and screen share after leaving	○	(none)
B35 Bypass locked webcam viewing (mechanism 1)	●	CVE-2022-23488
B36 Bypass locked webcam viewing (mechanism 2)	●	(none)
B37 Bypass locked webcam sharing	●	(none)
B38 Manipulate shared camera	●	CVE-2022-23489
B38.1 Activate participants’ webcams		
B38.2 View webcams in other meetings		
B39 Replace others’ webcam streams	●	CVE-2022-23489
B40 Abort running screen shares	●	(none)
B41 Bug: Receive metadata on shared videos	●	CVE-2022-29235
B42 Access recordings by meeting ID	○	(none)
B43 Bug: Exclude own camera from recording	○	(none)
B44 Bug: See limited camera streams (via recordings)	○	(none)
B45 Bug: See contents of shared notes (via recordings)	●	(none)

○ unfixed ● partially fixed ● fixed

References

1. 8x8, Inc., Vulnerability Disclosure Program Policy (2023). <https://hackerone.com/8x8>
2. Ahmed, M.: Hacking Zoom: Uncovering Tales of Security Vulnerabilities in Zoom (2020). <https://mazinahmed.net/blog/hacking-zoom/>
3. Altpeter, B.: RCE in Jitsi Meet Electron prior to 2.3.0 due to insecure use of shell.openExternal() (CVE-2020-25019) (2020). <https://benjamin-altpeter.de/jitsi-meet-electron-rce-shell-openexternal/>
4. Anthony, T.: Zoom Security Exploit - Cracking private meeting passwords (2020). <https://www.tomanthony.co.uk/blog/zoom-security-exploit-crack-private-meeting-passwords/>
5. Thévenet, A.: France digital strategy for education supports the use of digital commons (2023). <https://joinup.ec.europa.eu/collection/open-source-observatory-osor/news/france-digital-strategy-education-2>
6. Begen, A.C., Kyzivat, P., Perkins, C., Handley, M.J.: SDP: Session Description Protocol. RFC 8866 (Proposed Standard) (2021). <https://www.rfc-editor.org/rfc/rfc8866.txt>
7. BigBlueButton. French Ministry of Education chooses BigBlueButton (2023). <https://bigbluebutton.org/2023/01/11/french-ministry-of-education-chooses-bigbluebutton/>
8. Boström, H., Jennings, C., Castelli, F., Bruaroey, J-I.: WebRTC: Real-time communication in browsers. W3C recommendation, W3C (2023). <https://www.w3.org/TR/2023/REC-webrtc-20230306/>
9. Bräunlein, F.: MS Teams: 1 feature, 4 vulnerabilities (2021). <https://positive.security/blog/ms-teams-1-feature-4-vulns>
10. Böck, H.: File Exfiltration via Libreoffice in BigBlueButton and JODConverter (2020). <https://blog.hboeck.de/archives/902-File-Exfiltration-via-Libreoffice-in-BigBlueButton-and-JODConverter.html>
11. Castillo, I.B.: mediasoup v3 Design (2020). <https://mediasoup.org/documentation/v3/mediasoup/design/>
12. Davis, R.: Zoom's Bug Bounty Program: 2021 in Review (2022). <https://blog.zoom.us/zoom-bug-bounty-program-2021/>
13. Kasak, D., Callahan, D., Hodgson, M.: Practically-exploitable Cryptographic Vulnerabilities in Matrix (2022). <https://matrix.org/blog/2021/09/13/vulnerability-disclosure-key-sharing>
14. Fratric, I.: XMPP Stanza Smuggling or How I Hacked Zoom (2022). <https://i.blackhat.com/USA-22/Thursday/US-22-Fratric-XMPP-Stanza-Smuggling.pdf>
15. GÉANT. Build Your Own eduMEET Service (2020). <https://web.archive.org/web/20200416162612/https://edumeeet.org/build/>
16. heise online. Rheinland-Pfalz: Schulen dürfen Microsoft-Software Teams nicht mehr nutzen [Rhineland-Palatinate: Schools no longer allowed to use Microsoft Teams] (2022). <https://www.heise.de/news/Rheinland-Pfalz-Schulen-duerfen-Microsoft-Software-Teams-nicht-mehr-nutzen-7154309.html>
17. Jones, M.B., Bradley, J., Sakimura, N.: JSON Web Token (JWT). RFC 7519 (Proposed Standard) (2015). <https://www.rfc-editor.org/rfc/rfc7519.txt>. Updated by RFCs 7797, 8725
18. Keegan, R.: Patched Zoom Exploit: Altering Camera Settings via Remote SQL Injection (2020). <https://medium.com/@keegan.ryan/patched-zoom-exploit-altering-camera-settings-via-remote-sql-injection-4fdf3de8a0d>

19. Kelly, S.M.: Zoom’s massive ‘overnight success’ actually took nine years. CNN (2020). <https://edition.cnn.com/2020/03/27/tech/zoom-app-coronavirus/index.html>
20. Kinugawa, M.: Discord Desktop app RCE (2020). <https://mksben.l0.cm/2020/10/discord-desktop-rce.html>
21. Ling, C., Balci, U., Blackburn, J., Stringhini, G.: A first look at Zoom bombing. In: 2021 IEEE Symposium on Security and Privacy (SP), pp. 1452–1467 (2021). <https://ieeexplore.ieee.org/document/9638984>
22. Marczak, B., Scott-Railton, J.: Move fast and roll your own crypto - a quick look at the confidentiality of zoom meetings (2020). <https://citizenlab.ca/2020/04/move-fast-roll-your-own-crypto-a-quick-look-at-the-confidentiality-of-zoom-meetings/>
23. Martin, T., Radzio, M., Sharma, K.: Matrix concepts (2023). <https://matrix.org/docs/matrix-concepts>
24. Albrecht, M.R., Celi, S., Dowling, B., Jones, D.: Practically-exploitable Cryptographic Vulnerabilities in Matrix (2022). <https://nebuchadnezzar-megolm.github.io/static/paper.pdf>
25. McGrew, D., Rescorla, E.: Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP). RFC 5764 (Proposed Standard) (2010). <https://www.rfc-editor.org/rfc/rfc5764.txt>. Updated by RFCs 7983, 9443
26. Meyer, K.: GÉANT eduMEET service launched early to support communication needs during the COVID-19 lockdown (2020). <https://connect.geant.org/2020/06/16/geant-edumeet-service-launched-early-to-support-communication-needs-during-the-covid-19-lockdown>
27. Nettleton, R.: BigBlueButton (2010). <https://web.archive.org/web/20100814003302/https://edc.carleton.ca/blog/index.php/2010/06/04/bigbluebutton/>
28. s1r1us and TheGrandPew. Remote Code Execution on Element Desktop Application using Node Integration in Sub Frames Bypass - CVE-2022-23597 (2022). <https://blog.electrovolt.io/posts/element-rce/>
29. Sakimura, N., Bradley, J., Jones, M.B., de Medeiros, B., Mortimore, C.: OpenID Connect Core 1.0. OpenID Foundation (2014). <https://openid.net/specs/openid-connect-core-1.0-final.html>
30. Schreiber, P., Hoffman-Andrews, J., Grauer, Y.: Videoconferencing Guide (2020). <https://videoconferencing.guide/>
31. Sector7. Zoom RCE from Pwn2Own 2021 (2021). <https://sector7.computest.nl/post/2021-08-zoom/>
32. Silvanovich, N.: Adventures in Video Conferencing Part 1: The Wild World of WebRTC (2018). <https://googleprojectzero.blogspot.com/2018/12/adventures-in-video-conferencing-part-1.html>
33. Silvanovich, N.: Adventures in Video Conferencing Part 2: Fun with FaceTime (2018). <https://googleprojectzero.blogspot.com/2018/12/adventures-in-video-conferencing-part-2.html>
34. Silvanovich, N.: Adventures in Video Conferencing Part 3: The Even Wilder World of WhatsApp (2018). <https://googleprojectzero.blogspot.com/2018/12/adventures-in-video-conferencing-part-3.html>
35. Silvanovich, N.: Adventures in Video Conferencing Part 4: What Didn’t Work Out with WhatsApp (2018). <https://googleprojectzero.blogspot.com/2018/12/adventures-in-video-conferencing-part-4.html>

36. Silvanovich, N.: Adventures in Video Conferencing Part 5: Where Do We Go from Here? (2018). <https://googleprojectzero.blogspot.com/2018/12/adventures-in-video-conferencing-part-5.html>
37. Silvanovich, N.: Zooming in on Zero-click Exploits (2022). <https://googleprojectzero.blogspot.com//2022/01/zooming-in-on-zero-click-exploits.html>
38. Reuters Staff. Google bans Zoom software from employee laptops. REUTERS (2020). <https://www.reuters.com/article/us-google-zoom-idUSKCN21Q32V>
39. Sudhodanan, A., Paverd, A.: Pre-hijacked accounts: an empirical study of security failures in user account creation on the web. In: Proceedings of the 31st USENIX Security Symposium (USENIX Security 2022), pp. 1795–1812, Boston, MA (2022). USENIX Association. <https://www.usenix.org/conference/usenixsecurity22/presentation/sudhodanan>
40. Thodupunoori, R.: Part-1 Dive into Zoom Applications (2021). <https://rakesh-thodupunoori.medium.com/part-1-dive-into-zoom-applications-d70f3de53ec5>
41. Thodupunoori, R.: Part 2: Dive into Zoom Applications (2021). <https://rakesh-thodupunoori.medium.com/part-2-dive-into-zoom-applications-1b01091345c1>
42. Tudor, C.: The Impact of the COVID-19 pandemic on the global web and video conferencing SaaS market. Electronics **11**, 2633 (2022)
43. Vegeris, O.: “Important, Spoofing” - zero-click, wormable, cross-platform remote code execution in Microsoft Teams (2020). <https://github.com/oskarsve/ms-teams-rce>
44. Vela, E.: Zoom: XSS in Zoom.us Signup Flow (2020). <https://github.com/google/security-research/security/advisories/GHSA-fpgp-vmv-v8f2/>
45. Vengattil, M., Roulette, J.: Elon Musk’s SpaceX bans Zoom over privacy concerns -memo. REUTERS (2020). <https://www.reuters.com/article/us-spacex-zoom-video-commn-idUSKBN21J71H>
46. Website of the conference of ministers of education (Kultusministerkonferenz). Digitale Lernangebote [Digital Learning Tools] (2023). <https://www.kmk.org/themen/bildung-in-der-digitalen-welt/distanzlernen.html>
47. Wittmann, L.: Visavid - Datensicherheit im Warteraum [Visavid - Data Security in the Waiting Room]. Medium (2021). <https://lilithwittmann.medium.com/visavid-datensicherheit-im-warteraum-77c184c1d58a>
48. Zoom Video Communications, Inc., Vulnerability Disclosure Policy (2021). <https://www.zoomgov.com/docs/en-us/vulnerability-disclosure-policy.html>