Private and Secure Videoconferencing

Kevin E. Wilson, PhD
Georgia Institute of Technology
Computer Science Department, Cybersecurity
kevinwilson@gatech.edu

***Abstract*—Videoconferencing solutions for individuals and small organizations are dominated by commercial providers using proprietary software with terms of service that require individuals to agree to allow collection of personal information as a condition of service. Predominantly one open source solution exists, *Jitsi-Meet,* however it assumes the user can host and secure their own web server. Additionally, it does not publish a threat model or security audit.**

This project provides two complete, open source solutions based on *Jitsi-Meet* to meet the needs of individuals or small organizations. One is intended for cloud deployment and one intended as a dedicated server running in a SOHO (Small Office/Home Office) environment. It is available under the open source Apache 2.0 license at https://github.com/fgamgee/Jitsi-Meet-Secure-Server

Keywords—videoconferencing, privacy, Jitsi, security

I. INTRODUCTION

Covid-19 has disrupted society profoundly. To prevent the spread of the disease, individuals have been subject to mandatory or "voluntary" requirements to avoid contact with other people. This has caused a huge surge in the use of videoconferencing services for work, education, medical visits, and social interaction. Many of the interactions now taking place via videoconferencing are of a confidential nature. The use of video conferencing in place of in-person interactions brings with it numerous security and privacy issues that must be considered when choosing a videoconferencing solution.

Consider first the privacy issues brought about by videoconferencing. Privacy was defined succinctly by US Supreme Court Justice Louis Brandeis and Samuel Warren as "The right to be let alone" [1] and described as one of "the most valued rights of civilized men." They wrote in 1890:

Recent inventions and business methods call attention to the next step which must be taken for the protection of the person, and for securing to the individual what Judge Cooley calls the right "to be let alone". Instantaneous photographs and newspaper enterprise have invaded the sacred precincts of private and domestic life; and numerous mechanical devices threaten to make good the prediction that "what is whispered in the closet shall be proclaimed from the house-tops." For years there has been a feeling that the law must afford some remedy for the unauthorized circulation of portraits of private persons… [1]

which is as relevant in this "internet age" as when it was first penned.

When we consider the privacy offered by "free" videoconferencing solutions for consumers, the picture is bleak. An analysis in April 2020 of the privacy policies of three major commercial video conferencing providers for consumers (Google Meet, Microsoft Teams and Webex) found that "…from a privacy point of view, none of these options are great. According to their privacy policies, all three companies can collect data while you're in a videoconference, combine it with information from data brokers and other sources to build consumer profiles, and potentially tap into the videos for purposes like training facial recognition systems." [2]

Zoom, the market share leader in videoconferencing had numerous well-publicized security issues [3] and an analysis of its privacy policy found "Zoom spies on its users for personal profit. It seems to have cleaned this up somewhat since everyone started paying attention, but it still does it. The company collects a laundry list of data about you, including user name, physical address, email address, phone number, job information, Facebook profile information, computer or phone specs, IP address, and any other information you create or upload. And it uses all of this surveillance data for profit, against your interests." [4]

Additionally, the commercial videoconferencing providers use proprietary software. This prevents independent analysis of the security of the system violating a well-accepted principle of secure computing. For Zoom, which has most of its developers based in China, this raises a concern. Chinese law requires cooperation with the government for surveillance which could require hidden backdoors [5] and the Chinese government has a well-established record of suppressing public opinion and minority groups.

II. BACKGROUND

A. *Jitsi – Open source videoconferencing*

Jitsi is an open source videoconferencing solution which is supported by the 8x8 Corporation. 8x8 provides a free *Jitsi* Instance (meet.jit.si) with a much better privacy policy [6] than most commercial providers, but which still uses website analytics. The meet.jit.si service also lacks host controls. Additionally, there is no published threat model or security audit of the meet.jit.si service, and its configuration is proprietary [7].

Jitsi provides hop-to-hop encryption using WebRTC. Therefore, besides the users being trusted, which is common to all videoconferencing solutions, the server needs to be trusted as well. (*Jitsi* is in beta testing of End-to-End encryption as of July 2020 [8], but the date this feature will be available and the details of its security guarantees are unclear). Because *Jitsi* uses

WebRTC, it works with all 4 major browsers: Chrome, Safari, Firefox, and Edge, and with the major operating systems: Windows 10, iOS, and Linux. Most consumers will possess a pairing of these already. This is an advantage over some videoconferencing software which requires downloads of proprietary applications or plug-ins, risking opening additional vulnerabilities on a user's computer.

For those who want host controls, or who do not want to trust the 8x8 corporation due to the lack of transparency on how their server is secured etc., the *Jitsi* developer community provides Install Instructions for hosting your own videoconferencing server based on *Jitsi-Meet*. However, the instructions assume a significant technical background and lack important information, such as how to secure your server, register a domain name, and configure *Jitsi-Meet* for privacy. They also lack a threat model and security evaluation of the system [7]. Additionally, the installation alone has 17 separate Linux commands; and to restrict to authorized users the ability to set up new meetings three configuration files need to be edited to add multiple lines.

B. Prior Research on Video Conferencing Security.

The security of WebRTC as a videoconferencing solution has been previously examined [9] [10] and [11]. While the WebRTC standard was designed with security built in, the authors of the references provided point out that WebRTC is dependent on the specific software implementation in the server and browser. It also necessitates trust of the entities who can exercise control of the server and browser devices.

For example, an adversary with access to the server can easily capture traffic and act as a man-in-the-middle. Thus, absolute trust is assumed for the server and its software. By using open source software, such as Jitsi-Meet, you can at least be assured that the server software is open for inspection by a wide community which makes it less likely to have vulnerabilities and backdoors than proprietary software. However, control of the server and the security of all the software on the server is crucial in a private and secure videoconferencing solution. This is an impediment in hosting your own video conferencing solution if you are not skilled in the art of server security. It is also an issue if you are not inclined to trust the security of the hosting provider, as there are many publicized security breaches of seemingly well-respected companies.

The other vulnerable endpoint is the browser. Browsers have had various security vulnerabilities which have been exposed (Common Vulnerabilities and Exposures aka CVE's) and patched related to their implementation of WebRTC. Common browsers are now typically based on open source software which allows for better analysis of their security; however, they are very complex systems which are feature-rich and have a large attack surface. WebRTC CVE's are still common. Even as recently as late May 2020, both Firefox and Chrome had Critical and High, respectively, CVE's published with respect to their WebRTC implementation.

There are interesting and creative exploits which are based on natural speech and the timing and size of packets that carry even encrypted voice [12]. White, *et al.* found that they could reconstruct approximate transcripts of VOIP. WebRTC typically uses an "audio level" header which can leak information about the level of individual audio packets. This could possibly be used infer information about the conversation, possibly with phoneme-level resolution [13].

C. Problem Statement.

No private and secure open source video conferencing solution is available for organizations or individuals who need meeting host controls and/or password protection of the meetings unless they already have, or know how to host, a secure and private public-facing webserver.

III. Methods

A. Project Scope

This project addresses this need by providing instructions and automation for setting up a private and secure dedicated *Jitsi-Meet* server for video conferencing, either in the cloud or with user provided hardware in a SOHO environment with broadband internet connectivity. It is appropriate for utilization by a small organization or an individual. The project is run in the open source *Jitsi* community [14] on GitHub [15] and invites comments and contributions from the open source community. Secondarily to privacy and security, ease of use was also a design goal, with the intended audience being an individual who is technically savvy but lacks any experience with the required individual components (setting up a server, firewalls, DNS domain registration, etc.)

Provided as part of the project is a complete default server installation guide with automation for a private and secure dedicated *Jitsi-Meet* server with software and hardware recommendations, firewall rules and deployment, how to obtain a hostname, TLS certificate, typical server configuration for meetings (requiring a password for use of the service and restricting certain privileges, like muting a participant, to the meeting host), and management policy. Security documents also include a threat model, security, and maintenance automation.

As noted, there are guides for both a cloud and a stand-alone solution. While the cloud solution requires some compromise of privacy (you must trust the cloud provider not to use root access of the server to compromise your privacy), it may be the most popular option as it has the lowest fixed cost and can easily scale to larger groups if required.

B. Target Audience

The target audience for this project was individuals or small organizations, such as a small private school, scout troop, or religious organization, which want host controls and privacy from mass corporate and criminal surveillance. Some general technical knowledge is assumed, but specialized knowledge specific to hosting a web server, Linux, registering a domain, assigning an IP Address via a DNS, etc. is not assumed.

C. Threat Model

This project is designed to protect against mass surveillance from both corporations and criminals. It is designed to protect the users of the videoconferencing from:

1) Their personal information being collected by a corporation, and thus subject to subsequent security breaches with leakage of their information to criminals and others.

2) The videoconference being hijacked by hackers who want to disrupt or spy on the meeting.

3) Someone other than the host muting participants or kicking them out of the meeting.

4) Tracking analytics being collected on the participants in the conference.

5) Requiring the users to agree to share their information with a third party, such as the videoconferencing provider.

6) Requiring the users to download software which may contain vulnerabilities or otherwise compromise their computer in order to participate in the conference.

The project is not appropriate for protection from targeted attacks by nation-states. It provides a level of security against organized crime, but again may not be sufficient for targeted attacks from a well-resourced attacker who can gain physical access to the server or compromise one of the participant's computers.

All participants of the meeting are assumed to be trusted, as well as, by extension, the devices they use to connect to the meeting or devices that are present with them (e.g. digital assistants or phones) during the meeting. There is no protection that would keep participants from surreptitiously recording the meeting using smart phones or other devices, nor is there significant protection from a participant actively disrupting the meeting - though host controls (allowing the muting of participants and kicking them out of the meeting) can provide modest protection, appropriate for disruptive behavior by students in a classroom.

The Threat model follows the STRIDE Methodology [16] and is limited in scope to the hosting of a *Jitsi-Meet* Server and the data transit between the server and the participants devices.

TABLE I. STRIDE THREAT MODEL

Threat	Mitigation
Spoofing	Password Authentication required to start a meeting. Unique URL and password can be required to join the meeting. Let's Encrypt certificate for domain name.
Tampering	CIS Benchmark Level 1 and 2 on server, with a few required exceptions. Firewall. SSH with private key (2048 bit) for server management. Security updates automatically configured. Exposed CVE's eliminated.
Repudiation	Can conflict with privacy goal in general case. However, meetings can require passwords which could be sent securely to participants.
Information Disclosure	Encryption using WebRTC. Configured to require TLSv1.2 with forward secure strong ciphers. No logs relevant to the videoconference kept on the server.
Denial of Service (DOS)	Protection is very rudimentary against DOS. Requiring authentication to start a meeting prevents adversary from starting multiple meetings. Restrictive firewall rules.
Elevation of Privilege	CIS Benchmark Level 1 and 2 on server which enables AppArmor and other defense in depth measures.

7) Outside of Scope:

a) DOS and DDOS (Distributed Denial of Service) attacks.

b) Man-in-the-middle-attacks. This is a difficult attack to perform, needs to be targeted and is typically executed by a strong adversary.

c) Adversary gains physical access to the server.

d) Adversary gains access to a participant's device.

e) Targeted attacks based on natural speech as previously described [12]. Again, this requires a very resourced attacker.

D. Project Design

1) Constraints

For the cloud solution, cost will be the primary constraint, but it is expected that cost will be very modest, approximately $50 for a year for 300 hours of videoconferencing for 10-20 participants, including Domain name registration and DNS use. This solution can be very easily scaled to more participants by purchasing a more powerful instance. The cost is billed at per hour of use. The approximate cost for 10-20 participants is $0.10 an hour (in 2020 on AWS t3.large [17]), and an instance that could comfortably handle twice as many participants (t3.xlarge) is about $0.20 per hour. There are issues in scaling past 70 participants (multiple servers handling different components of the videoconference are necessary), and this solution is not appropriate for gatherings larger than 70 participants.

For a stand-alone server in a SOHO environment, network bandwidth is likely to limit the number of participants. Cloud servers typically have 5 Gb/s network bandwidth, while a SOHO environment usually has much less. Cost will be much higher than the cloud solution if new hardware is required and/or network bandwidth must be purchased. If existing hardware can be repurposed and network bandwidth is freely available or already purchased the only costs are the domain name and DNS Server registration, which is less than $20 a year with no limitation on number of hours.

2) System Environment

The project was developed for an Ubuntu 18.04 server with CIS (Center for Internet Security) Level 1 and 2 Benchmarks Version 2.01 applied [18] (with a few required exceptions). CIS Level 1 benchmarks are consensus-based best practices for hardening a server. Level 1 benchmarks are intended to be "practical and prudent; provide a clear security benefit; and not inhibit the utility of the technology beyond acceptable means." Level 2 benchmarks are "intended for environments or use cases where security is paramount; acts as defense in depth measure; may negatively inhibit the utility or performance of the technology." [18] These benchmarks were applied using Ansible and the open source project of Florian Utz [19] under the MIT License.

The cloud solution was developed and tested with Amazon AWS, but none of the automation is specific to AWS. However, the specific installation instructions on how to navigate the AWS website would not be relevant to a different cloud, domain registrar, and DNS provider.

The stand-alone solution was developed and tested using an Intel NUC 10 i7 with a core i7 processor, 32 GB of Memory,

and a 256GB SSD drive (of which only 64 GB was needed) behind an Ubiquiti EdgeRouter on a Verizon FIOS network of 75 Mb/s (upload and download). This solution is bandwidth, not CPU or Memory, limited and can likely support only 8-10 participants (depending on browsers the participants use). This CPU/Memory/Disk would be more than sufficient for a 1 Gb/s network connection. The number of participants *Jitsi-Meet* can support with specific bandwidth and hardware is outside of the scope of this project but has already been well documented [20] [21].

3) Architectural Design

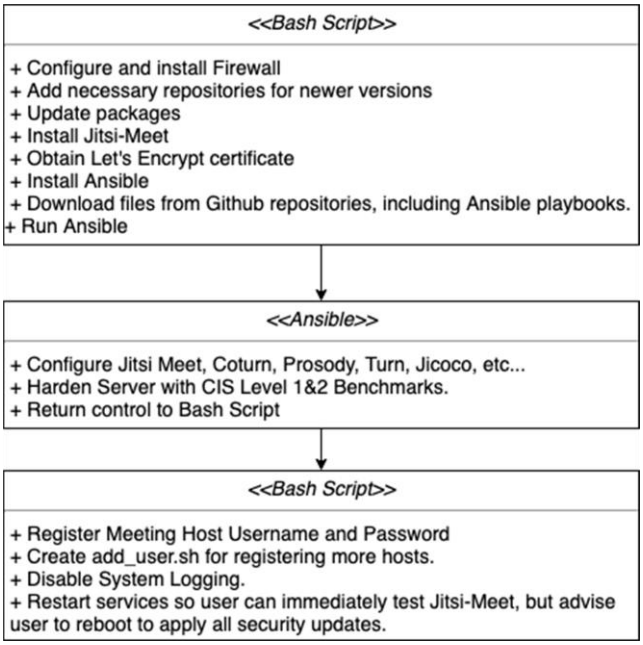

Figure 1: Architectural design diagram

4) Components

A bash script is downloaded from the GitHub repository [15] and utilized to run the installation process. The firewall is configured and activated. The script installs *Jitsi-Meet* [22], Let's Encrypt [23], and Ansible [24] with all needed package dependencies. Ansible is a popular open source package for the configuration and maintenance of servers. Ansible is used to harden the software with recommendations from the CIS Benchmark using Florian Utz's open source project [19], however not all of the CIS recommendations are used. In particular, Section 4, "Logging and Auditing," is disabled to increase privacy. Also, *Jitsi-Meet* uses coturn which has telnet as a hard package dependence, though the telnet client is not started and coturn is configured to disallow its use (a future enhancement would be to install a mock telnet package to satisfy the coturn package dependence). Ansible is used to configure *Jitsi-Meet* and related components, nginx, prosody, coturn, etc.

5) User Interface

Detailed instructions are provided for those steps that are not amendable to automation, such as obtaining a domain name, IP address, and similar.

Once the user is in the server, the user interface is the command line. The user needs only to enter three commands to execute the automated setup process. Instructions are provided on how to copy and paste these three commands from the browser into the server terminal for the cloud implementation. The user then needs to answer a few questions (e.g. domain name, e-mail address, create meeting host name and password).

IV. RESULTS

A. Security Evaluation

Three separate installations of *Jitsi-Meet* were evaluated and the results are in Table II.

1) The *Jitsi-Meet* install as provided by the *Jitsi* Community on a standard Ubuntu 18.04 Server installation: Version 1.0.4127 (Stable) [25], released May 26, 2020.

2) The installation provided by this project for a cloud instance using *Jitsi-Meet* Version 1.0.4127 (Stable), released May 26, 2020 and project tag V0.11 [26] released June 30, 2020.

3) The 8x8 meet.jit.si server that 8x8 provides for free, to promote the use of *Jitsi*: *Jitsi-Meet* Version 1.0.4203 (with 8x8 customization) tested July 18, 2020.

This project's installation of the standalone server was not evaluated separately as it is identical to the cloud instance except in disk partitioning and partition permissions, which were not tested by any of the tools.

The following automated tools were used for the evaluation:

a) Qualsys SSL Labs [27] - Automated "free online service that performs a deep analysis of the configuration of any SSL web server on the public Internet." Tests SSL and various vulnerabilities (DROWN, Heartbleed, Poodle, etc.). "SSL Labs is a non-commercial research effort." [27]

b) Immuniweb SSL [28] - Automated free service with paid premium services. Tests SSL and various vulnerabilities (DROWN, Heartbleed, Poodle, etc..).

c) tls.imirhil.fr [29] - Automatedly tests TLS implementation for insecure ciphers. Its tests are strongly focused on forward secrecy.

d) securityheaders.com [30] - Automated free website, built by Scott Helme, to assess quality of security headers. It tests Strict-Transport-Security, Content-Security-Policy (CSP, mitigates XXS attacks), X-Frame-Options (mitigates clickjacking), X-Content-Type-Options (mitigates MIME-sniff), and Referrer-Policy (Prevents browser from providing information on current site to the next site visited).

e) Mozilla SSH Observatory [31] - Automatedly tests for weak key exchange, ciphers and MACs used in SSH connection.

f) Greenbone Community Editions V3.1 (formerly OpenVAS) [32] - A semi-automated sophisticated security analysis/vulnerability assessment tool. Community edition is open source. Results are many and diverse, depending on configuration. It finds CVE's of packages, misconfigurations,

informational leaks that might be useful to an adversary, etc. Many of the results are informational only.

Important results of the above automated test tools are included included in the Table II.

TABLE II. SECURITY EVALUATION

Evaluation	Tested Server		
	Jitsi-Meet (stable)	***This Project***	***8x8 meet.jit.si***
Qualsys SSL Labs Grade	B	A	B
Immuniweb SSL Grade	A-	A	A
tls.imirhil.fr Grade	B	A	B
Securityheaders.com Grade	D	A	D
SSH Observatory Grade	C	A	N/A
CVE's discovered by Greenbone Community Editions, without internal access to the server (rating in parenthesis) Maximum CVE rating out of 10	CVE-2018-16843 (7.5), CVE-2018-16844 (7.5), CVE-2018-16845 (6.1) 7.5 High	None	None
Other CVE's in outward facing application components (rating in parenthesis) Maximum CVE rating out of 10	CVE-2018-4059 (9.8), CVE-2018-4058 (7.7) 9.8 Critical	None	None

The CVE's in public-facing applications existed even though Ubuntu Security updates were applied. These components, nginx, coturn and prosody, are in theory supported by the Ubuntu Repository, however the newer, patched versions of those packages were not available through the Ubuntu Repository even though the CVE's date from 2018. The nginx and prosody CVE's were fixed in this project by using ppa repositories for Ubuntu 18.04 which have newer versions of those packages, specifically ppa:nginx/stable and https://packages.prosody.im/debian bionic main. The coturn CVE's were mitigated using a configuration of coturn recommended by Talos that did not expose the vulnerability [33] [34].

Some additional hardening was applied on unrated issues reported by Greenbone Community Edition. Namely, the nginx version number in the http header was suppressed (to deny an adversary the information) and http methods were limited to GET, HEAD, POST and OPTIONS (all needed by Jitsi-Meet), denying use of HTML TRACE, TRACK, DELETE and PUT which an adversary might use to advantage.

Some CVE's of a less serious nature, since they do not exist in outwardly facing components, should be addressed in the future. The highest rated CVE (6.5 Medium, a denial of service vulnerability) is from the use of python 2.7.17, which is installed by the Jitsi-provided script which installs the Let's Encrypt certificate. Since this is only used from within the server, an adversary would first need user access within the server.

After the security evaluation was privately shared with the Jitsi developers at 8x8 corporation, they were concerned that the default community install had critical CVE's and asked that this security evaluation be kept private until they could issue a new release, which they promised to do "soon". The mitigations have made it into the nightly builds, but the stable version has not yet been updated as of July 18, 2020.

B. Privacy Evalution

As stated earlier, there is little privacy protection if a hosting server is compromised before or during a videoconference.

For the cloud solution, the server is under direct control of a third party, and the person or organization using the service must identify themselves to the provider for billing purposes and agree to the provider's terms and conditions. The person or organization does not have to identify for what purpose the server will be used for, though they may have to attest that it is a lawful purpose. Participants in the videoconference do not need to provide any information to the cloud provider.

Additionally, there are many more cloud providers to choose between (21 are listed at this reference [35]) than videoconferencing solutions, so there is a large amount of choice. There is fierce competition in the cloud services market, with an important competing point being privacy of the data and the physical security of the servers, as companies (the primary customers) are very concerned about the possibility that their data could be exposed. As such "Most companies adopt very strict protocols to ensure data security, confidentiality, and privacy. [35]" If a company violates it stated privacy policy, it can be prosecuted by the FTC (Fair Trade Commission); this represents a large business risk, as many users would switch to another provider if it were revealed that a cloud provider had been caught "snooping".

The one exception to the privacy guarantee provided by cloud providers is that they can be required by law to give access to the government. Fortunately for US citizens using a cloud within the US, there are strong protections from government snooping provided by the Fourth Amendment, which requires "probable cause" to obtain a search warrant, and additional protections from the Electronic Communications Privacy Act (ECPA). The US has some of the stronger privacy protections from the government within the US for its own citizens, but rather weak protection of the privacy of citizens from commercial snooping, particularly when users "consent" to their own surveillance by agreeing to the published privacy policy [36], as in commercial consumer videoconferencing solutions [2].

For the standalone server, the terms and conditions you must agree to are determined by the network provider, and typically there is little choice of what provider is available to you. However, the server cannot be accessed by the network provider who can only examine the metadata since the data is encrypted in transit.

Having the server directly under your physical control can be viewed as an advantage or disadvantage, depending on your

threat model. When the server is in part of a cloud provider, they typically provide physical security for the facility in terms of security guards, security cameras, access control, etc. When the server is under your direct control, you are responsible for the physical security of the server.

In both the standalone and cloud solutions, additional protection is provided because all relevant logging is automatically disabled, so there is no record ever written to disk. All the saved logs were manually checked to verify that no information is being saved about any videoconference. Some logs remain on the server that would allow an adversary to know when the system was turned on or off, when packages were installed, sudo was used, etc., but those were deemed to be more useful to an administrator monitoring the system than useful for an adversary in discovering information about the content of the videoconferences conducted.

C. User Evalutions

Six independent testers tried to follow the installation instructions for setting up a Jitsi-Meet host in the cloud (Amazon AWS). None of the testers had experience with registering a domain name, assigning an IP address to a domain name in a DNS service, or using a cloud provider. Four testers were public school teachers, and thus were considered within the target audience. None of the teachers had experience with Linux.

The time taken to complete setup of a cloud based Jitsi-Meet server ranged from 1 hour and 45 minutes with no help required to 7 1/2 hours - with assistance and ultimately giving up without accomplishing the setup (4th tester). The median time was 2 hours 15 minutes. As each tester finished, the project was updated to improve the areas in which the tester had had problems. Of the four who are public school teachers, two got through it in a little over 2 hours, and two gave up without being successful. No one in the target audience was able to complete the setup without a phone call for assistance - but the next to last one would have completed it without help if the instructions had not had a typographical error (since corrected).

Most of the user issues arise from the part of setup that is not amendable to automation: registering a domain name, obtaining a public IP address, assigning the IP to the domain name in a Domain Name Server, and obtaining and using a public/private key pair to ssh into the server. These tasks are all done through GUI's at Amazon AWS's or of the OS. Numerous screen shots were provided, but the concepts of what is being accomplished is quite foreign to the users. Without any real understanding, it is easy to go astray. A video might be the best way to help, and I prepared a small video which the 5th and 6th tester used. Usability is an area that needs future improvement.

Once the testers were logged into their server, all but one finished without issue. Given the automation they just had to copy and paste three commands, and then type in their domain name and e-mail address when prompted.

As evidence that the project does not impair the use of *Jitsi*, two videoconferences were done using the cloud solution. The first, with extended family, used as many devices as possible including Windows 10 desktop, Macbook Pro, Macbook Air, iPhone and Android Pixel, with browsers including Chrome, Firefox, and the iPhone and Android *Jitsi* apps. No issues were found, and the consensus was it worked similarly to Zoom.

The second videoconference was for a 90-minute BSA scout meeting. It had nine participants and there were no issues, with all participants chatting together, sharing screens as they went through checklists, adding ideas to the chat, etc. The consensus was it worked well, similar to what they had used previously, Zoom and Google Meet. They thought the audio was somewhat better, but video transition between speakers a fraction of a second longer than Zoom. The feature set was intuitive to use and had all the features they were accustomed to using (chat, screensharing, muting and unmuting, etc.). They plan to use this videoconferencing solution for their future meetings because of its better privacy.

For the scout meeting, with nine participants, the AWS t3.xlarge instance used on average 12% of the CPU and 2.5% of the available bandwidth (125 Mbs of 5 Gbs available). Therefore, more participants could have been accommodated.

V. DISCUSSION

This project, to this author's knowledge, is the first to provide a fully private, secure, open source videoconferencing solution aimed at a non-expert audience that does not have experience with securing a public-facing internet server, registering a domain name, and using a cloud provider. It has a number of strengths.

Firstly, it has a threat model that provides definition to the user on the type of threat which it is designed to protect against, as well as the type of threats that it does not protect against. Thus, a user can judge whether it is appropriate for his circumstances.

Secondly, it has a security evaluation. This evaluation indicates that the provided solution scores well in tests of its transport encryption (including forward secrecy) for its users. Mitigation is also provided to protect against common web attacks such as cross-site scripting and clickjacking. The server is hardened so that there were no existing significant CVE's or vulnerabilities in exposed application software packages. Defense in depth was provided by applying consensus-based security best practices for servers via the CIS Benchmarks Level 1 & 2. Security updates are enabled to keep the system secure against future threats.

Thirdly, the security analysis and the reaction of the Jitsi Developers to the vulnerabilities means that the default configuration provided by the Jitsi community will be more secure and not as permissive. Unused services (e.g. coturn telnet) will be turned off by default, reducing the attack surface, and encryption will be upgraded to newer, more secure, algorithms. This will benefit everyone who depends on Jitsi-Meet software, not just those who use the solution described in this paper.

Finally, privacy is provided not only by encryption, but by keeping only system logs which do not contain information about video conferencing. Additionally, the site is free of website analytics.

Nevertheless, the project also has some weaknesses. The goal of making it accessible to a typical schoolteacher was not fully realized. While significant automation simplified the task, some formidable obstacles remain that are not readily amenable to automation and involve concepts that are quite foreign to the typical consumer. Future work could simplify the process of registering a domain name and assigning it an IP address with a DNS service provider, perhaps by providing a video tutorial to help the user navigate the issues.

Additionally, new CVE's are continually discovered. Automatic security updates are enabled, but as has been shown in the security evaluation for this project, those updates cannot be fully relied upon to remove all high and critical CVE's. This is partially mitigated by adding repositories for Jitsi, nginx and prosody which are not as stale as Ubuntu's repository. Nevertheless, the problem of prompt application of security patches is a vexing one and has been shown to affect even professionally-managed servers in various high-profile data breaches such as the Equifax breach, which stemmed, at least in part, from not promptly applying security patches [37].

VI. Conclusion

This project advances privacy by providing a more accessible private and secure videoconferencing solution based on *Jitsi-Meet*. There are two solutions provided, one utilizing a cloud provider, which has cost and scaling advantages, and one based on user-provided hardware which provides additional privacy through independence from a cloud provider. It is available open source under the Apache 2.0 license from this repository https://github.com/fgamgee/Jitsi-Meet-Secure-Server.

Acknowledgments

I thank the entire Jitsi development team for making Jitsi available to the open source community under the Apache 2.0 license, without which this project would not have been feasible. Particular thanks go to the developers who work at 8x8 for the technical assistance they provided, particularly Saúl Ibarra Corretgé, Emil Ivov, Boris Grozev, Damian Minkov and Jonathen Lennox.

This project was done in partial fulfillment of the requirements of Georgia Institute of Technology class CS 6727 "Cybersecurity Practicum". I thank Professor Mustaque Ahamad, PhD for his guidance during the course.

I thank Peter Wilson for his technical help with the Ubiquiti EdgeRouter, his technical advice throughout the project, and his proofreading the manuscript.

I thank Catherine Trotter Wilson, PhD for suggesting the project to me and for her help in proofreading the manuscript.

References

[1] S. Warren and L. Brandeis, "The Right to Privacy," *Harvard Law Review*, vol. 4, p. 193, 1890.

[2] A. S. John, "It's Not Just Zoom. Google Meet, Microsoft Teams, and Webex Have Privacy Issues, Too.," *Consumer Reports*, 30 April 2020.

[3] B. Marczak and J. Scott-Railton, " Move Fast and Roll Your Own Crypto: A Quick Look at the Confidentiality of Zoom Meetings," 3 April 2020. [Online]. Available: https://citizenlab.ca/2020/04/move-fast-roll-your-own-crypto-a-quick-look-at-the-confidentiality-of-zoom-meetings/.

[4] B. Schneier, "Security and Privacy Implications of Zoom," 3 April 2020. [Online]. Available: https://www.schneier.com/blog/archives/2020/04/security_and_pr_1.html.

[5] "Can Zoom be trusted with users' secrets?," *The Economist*, 20 June 2020.

[6] " 8×8 Privacy Notice," February 2020. [Online]. Available: https://www.8x8.com/terms-and-conditions/privacy-policy.

[7] krbonne, "Security audit?," 24 March 2020. [Online]. Available: https://community.jitsi.org/t/security-audit/25401.

[8] E. Ivov, "This is what end-to-end encryption should look like!," Jitsi, 12 April 2020. [Online]. Available: https://jitsi.org/blog/e2ee/.

[9] A. Reiter and A. Marsalek, "WebRTC: your privacy is at risk.," in *Proceedings of the Symposium of Applied Computing*, 2017.

[10] R. L. Barnes and M. Thomson, "Browser-to-Browser Security Assurances for WebRTC," *IEEE Internet Computing*, pp. 18(6): 11-17, 2014.

[11] B. Feher, L. Sidi, A. Shabtai and R. Puzis, "The Security of WebRTC," *CoRR abs/1601.00184*, 2016.

[12] A. White, A. Matthews, K. Snow and F. Monrose, "Phonotactic Reconstruction of Encrypted VoIP Conversations:Hookt on fon-iks," in *Proceedings of IEEE Symposium on Security and Privacy*, 2011.

[13] C. Perkins and J. Valin, "Guidelines for the Use of Variable Bit Rate Audio with Secure RTP - RFC 6562," Internet Engineering Task Force (IETF), 2012.

[14] "Jitsi Community," [Online]. Available: https://community.jitsi.org/.

[15] K. E. Wilson, "Jitsi-Meet-Secure-Server," 2020. [Online]. Available: https://github.com/fgamgee/Jitsi-Meet-Secure-Server.

[16] P. Garg and L. Kohnfelder, "The STRIDE Threat Model," 12 November 2009. [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20)?redirectedfrom=MSDN.

[17] AWS, "Amazon EC2 T3 Instances," 2020. [Online]. Available: https://aws.amazon.com/ec2/instance-types/t3/.

[18] CIS, "CIS Ubuntu Linux 18.04 LTS Benchmark V.2.0.1 01-03-2020," 3 January 2020. [Online]. Available: https://www.cisecurity.org/cis-benchmarks/.

[19] F. Utz, " florianutz / Ubuntu18040-CIS," 2020. [Online]. Available: https://github.com/florianutz/Ubuntu1804-CIS.

[20] B. Grozev and E. Ivov, "Jitsi Videobridge Performance Evaluation," 26 11 2014. [Online]. Available: https://jitsi.org/jitsi-videobridge-performance-evaluation/.

[21] R. Brown, "Recommended Server Specs for 2020?," 9 April 2020. [Online]. Available: https://community.jitsi.org/t/recommended-server-specs-for-2020/32041.

[22] Jitsi Community, "jitsi-meet," [Online]. Available: https://github.com/jitsi/jitsi-meet/.

[23] EFF project, "certbot," [Online]. Available: https://certbot.eff.org/.

[24] Red Hat, "Red Hat Ansible," [Online]. Available: https://www.ansible.com/.

[25] Jitsi Community, "Jitsi-Meet Stable Tag 4627," 26 May 2020. [Online]. Available: https://github.com/jitsi/jitsi-meet/tree/stable/jitsi-meet_4627.

[26] K. Wilson, "Jitsi-Meet-Secure-Server Tag v0.11," 30 June 2020. [Online]. Available: https://github.com/fgamgee/Jitsi-Meet-Secure-Server/tree/v0.11.

[27] Qualys, "SSL Server Test," Qualys, [Online]. Available: https://www.ssllabs.com/ssltest. [Accessed July 2020].

[28] Immuniweb, "Immuniweb AI for Application Security," Immuniweb, [Online]. Available: https://www.immuniweb.com/ssl/. [Accessed July 2020].

[29] @aeris22, "tls.imirhil.fr/tls/," [Online]. Available: https://tls.imirhil.fr/tls/. [Accessed July 2020].

[30] URI, Report, "Security Headers," Report URI, [Online]. Available: https://securityheaders.com/. [Accessed July 2020].

[31] Mozilla, "Observatory moz://a," mozilla, [Online]. Available: https://observatory.mozilla.org/ . [Accessed July 2020].

[32] Greenbone, "Greenbone Sustainable Resilience," Greenbone, [Online]. Available: https://www.greenbone.net/en/community-edition. [Accessed July 2020].

[33] Talos, "Talos Vulnerability Report CVE-2018-4059," Cisco, 20 January 2018. [Online]. Available: https://talosintelligence.com/vulnerability_reports/TALOS-2018-0733.

[34] Talos, "Talos Vulnerability Report CVE-2018-4058," 29 January 2018. [Online]. Available: https://talosintelligence.com/vulnerability_reports/TALOS-2018-0732.

[35] Guru99, "Top 21 Cloud Computing Service Provider Companies in 2020," Guru99, 2020. [Online]. Available: https://www.guru99.com/cloud-computing-service-provider.html.

[36] P. Swire and D. Kennedy-Mayo, U.S. Private-Sector Privacy, Portsmouth, NH: IAPP (International Association of Privacy Professionals), 2018.

[37] J. Thomas, A Case Study Analysis of the Equifax Data Breach 1 A Case Study Analysis of the Equifax Data Breach, 2019.