If You Can't Get Them to the Lab: Evaluating a Virtual Study Environment with Security Information Workers

Nicolas Huaman [C] Alexander Krause [C] Dominik Wermke[C] Jan H. Klemmer [*]
Christian Stransky [*] Yasemin Acar [†] Sascha Fahl [C]

[C]*CISPA Helmholtz Center for Information Security, Germany,*
{nicolas.huaman,alexander.krause,dominik.wermke,sascha.fahl}@cispa.de
[*]*Leibniz University Hannover, Germany,* {klemmer,stransky}@sec.uni-hannover.de
[†]*George Washington University, USA,* acar@gwu.edu

Abstract

Usable security and privacy researchers use many study methodologies, including interviews, surveys, and laboratory studies. Of those, lab studies allow for particularly flexible setups, including programming experiments or usability evaluations of software. However, lab studies also come with challenges: Often, it is particularly challenging to recruit enough skilled participants for in-person studies. Especially researchers studying security information workers reported on similar recruitment challenges in the past. Additionally, situations like the COVID-19 pandemic can make in-person lab studies even more challenging. Finally, institutions with limited resources may not be able to conduct lab studies.

Therefore, we present and evaluate a novel virtual study environment prototype, called OLab, that allows researchers to conduct lab-like studies remotely using a commodity browser. Our environment overcomes lab-like study challenges and supports flexible setups and comprehensive data collection. In an iterative engineering process, we design and implement a prototype based on requirements we identified in previous work and conduct a comprehensive evaluation including a cognitive walkthrough with usable security experts, a guided and supervised online study with DevOps, and an unguided and unsupervised online study with computer science students. We can confirm that our prototype supports a wide variety of lab-like study setups and received positive feedback from all study participants.

1 Introduction

Laboratory studies are common in usable security and privacy research and find broad application in many experiments with end-users or expert users. Researchers can flexibly set up very specific study environments and collect a wide variety of data, including video and audio recordings [31, 50, 41], think-aloud data [2, 16, 5, 37], or user behavior for particular software and tooling [4, 12, 10, 5]. While laboratory studies support flexible experimental setups and data collection, they come with the following challenges:

Participant Recruitment. It is often challenging to recruit sufficiently skilled participants for in-person lab studies. Due to the geographic location of a research lab or specific requirements for participants such as age [35], gender [19], or professional experience [2, 5, 30, 47], a laboratory study might not be feasible. In all scenarios, conducting expert studies with *security information workers* (SIWs) to test security development and system design [3, 11, 1, 37, 23, 10], system configuration and administration [16, 46, 45], or test and analyze those systems' security [31, 12, 5, 41, 32] is challenging. Local expert participant pools are usually too small and might lack diversity, representativeness, or statistical power. These challenges required researchers to be pragmatic in their study design, e. g., by recruiting computer science students as stand-ins for developers for lab studies [2, 24, 25, 16, 12, 46, 32] or by simplifying programming tasks to a few lines of code that can be studied online. Hence, researchers have started to conduct expert studies remotely [43, 38, 27, 5, 50, 37].

Complicated Circumstances. Laboratory studies in-person are feasible as long as no circumstances prohibit inviting participants to a research lab. However, events such as the COVID-19 pandemic make in-person lab studies even more challenging. Alternatively, researchers conduct studies online [52, 1, 4, 31, 10] dealing with the same restrictions as described above. Additionally, lab studies require certain resources, including space, personnel to supervise participants, and equipment, e. g., workstations to conduct studies.

To address the challenges above, we make the following contributions:

- **We use a literature-based requirements engineering process** to identify requirements for typical lab studies in usable security and privacy research based on previous

USENIX Symposium on Usable Privacy and Security (SOUPS) 2022.
August 7–9, 2022, Boston, MA, United States.

work. Therefore, we analyze 24 publications, including SIW studies, since they require particularly skilled participants who might be geographically widely distributed and usually hard to recruit.

- **We design and implement a virtual study environment prototype** that we call *OnlineLaboratory* (short OLab). OLab supports highly flexible lab-like online studies with extensive data collection. This virtual study environment allows researchers to recruit participants from anywhere and conduct highly customized studies in a commodity browser. Researchers can freely choose operating systems and tooling and collect a wide variety of data from participants, including edited files, copy and paste events, browser histories, and screen and audio recordings.
- **We evaluate our virtual study environment prototype in three studies** (Section 4) to illustrate its applicability to security studies with expert users. First, we conduct a cognitive walkthrough with four usability experts; second, a guided study with nine experienced DevOps; and third, an online programming experiment with 16 computer science students.
- Based on the evaluation results, we iteratively improved OLab's usability and user experience.

We use a literature-based requirements engineering process to design and implement a virtual study environment prototype that we call *OnlineLaboratory* (short OLab). OLab supports highly flexible lab-like online studies with extensive data collection. This virtual study environment allows researchers to recruit participants from anywhere and conduct highly customized studies in a commodity browser. Researchers can freely choose operating systems and tooling and collect a wide variety of data from participants, including edited files, copy and paste events, browser histories, and screen and audio recordings.

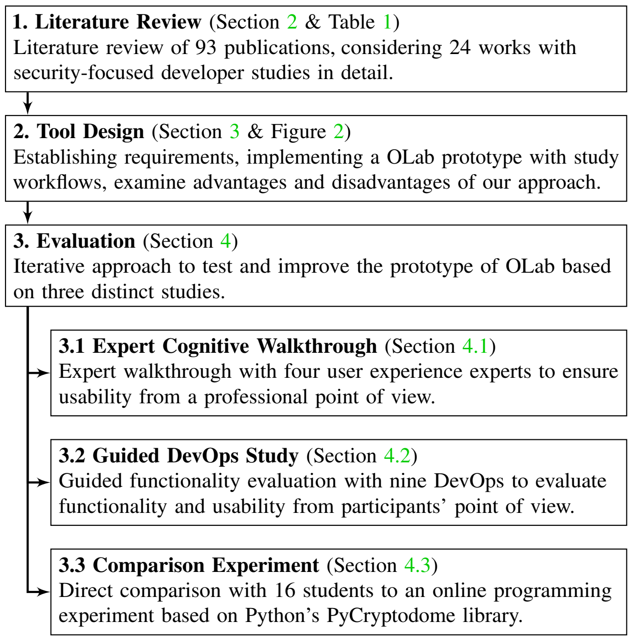
Figure 1 depicts the overall structure of this work. We provide further information regarding OLab on an accompanying website.[1]

While we designed and evaluated our prototype in the light of usable security experiments with SIWs, we are convinced that it can be seen as a blueprint for a general-purpose platform to conduct lab-like usable security and privacy user studies with expert users and end-users, during the COVID-19 pandemic and beyond.

2 Related Work

We discuss related work focusing on security information workers in three key areas: Laboratory experiments, remote studies that are not browser-based, and browser-based online studies. Finally, we aimed to identify the most recent and

[1] https://publications.teamusec.de/2022-soups-olab/


1. Literature Review (Section 2 & Table 1)
Literature review of 93 publications, considering 24 works with security-focused developer studies in detail.

2. Tool Design (Section 3 & Figure 2)
Establishing requirements, implementing a OLab prototype with study workflows, examine advantages and disadvantages of our approach.

3. Evaluation (Section 4)
Iterative approach to test and improve the prototype of OLab based on three distinct studies.

3.1 Expert Cognitive Walkthrough (Section 4.1)
Expert walkthrough with four user experience experts to ensure usability from a professional point of view.

3.2 Guided DevOps Study (Section 4.2)
Guided functionality evaluation with nine DevOps to evaluate functionality and usability from participants' point of view.

3.3 Comparison Experiment (Section 4.3)
Direct comparison with 16 students to an online programming experiment based on Python's PyCryptodome library.

Figure 1: Diagram illustrating literature review, OLab's design, and its evaluation.

relevant requirements for OLab based on task-based studies with SIWs, published at high-quality venues in the recent past (cf. Table 1). Therefore, we included work from USENIX SOUPS, ACM CHI, IEEE Security and Privacy, USENIX Security, NDSS, ACM CCS, and (Euro)USEC. We use them as a foundation to design, implement, and evaluate our OLab environment.

Lab Studies. Conventional lab studies often focus on participants performing a task on-site, either with a researcher or while being observed.

Acar et al. investigated in a lab study with 54 Android developers, how different information sources affect code security when solving security and privacy-related programming tasks [2]. Krombholz et al. conducted a lab study with 28 participants that had to securely configure TLS on a web server to explore and identify usability challenges in that process [16]. Follow-up work by Tiefenau et al. applied the original study methodology to *Let's Encrypt* and *Certbot*, finding better usability leading to a higher number of secure TLS deployments [46]. Naiakshina et al. performed a qualitative usability study with 20 computer science students to understand better how developers handle secure password storage [24]. In 2018, Naiakshina et al. replicated a study examining ecological validity by priming participants using the deception of a real-world task; they concluded that it has a significant impact, resulting in more secure solutions [25]. Hänsch et al. examined the understanding of obfuscated source code in reverse engineering process in a lab study with 66 students [12]. Nosco et al. proposed a new search strategy for finding bugs and software vulnerabilities. They grouped 12 participants into small teams within a ten-day lab experiment and had to discover vulnerabilities in several services [30]. Smith et al. conducted both a heuristic walkthrough and a lab study evaluating the usability of four security-focused static analysis tools, finding several usability issues. In the lab study, 12 developers had to fix warnings reported by those tools [41]. A similar study by Tupsamudre et al. identified several usability problems in two open-source *Static Application Security Testing* (SAST) tools; in a lab study, eight developers had to solve a password storage task in a web application while using the tools [47]. Plöger et al. conducted a lab study evaluating the usability of the Clang Static Analyzer and libFuzzer with 32 local CS Students and Capture-the-flag players, finding that libFuzzer performs a lot worse on usability compared to Clang Static Analyzer [32]. All in all, this research has in common the limitation of a local sample, often using students as stand-ins for developers or administrators.

Remote Studies. Besides conventional lab studies, some study setups work remotely over the internet, using online calls or self-reporting so participants can solve the tasks in any location using their own computers.

For example, Ruef et al. presented a novel cybersecurity contest that included breaking code and encouraging developers to build secure applications [38]. The authors hosted the contest and evaluated the solutions via an automated system regarding security. Based on the contests, Votipka et al. conducted an in-depth qualitative analysis to understand common security mistakes made by developers [49]. Nguyen et al. demonstrated in a remote experiment with 39 Android developers and students that IDE security plugins can be an effective measure helping developers with writing secure code [27]. Aksu et al. evaluated the open-source vulnerability scanner *OpenVAS* concerning its usability by employing heuristic walkthroughs and an experiment with 10 security experts at a single cybersecurity company. The participants had to solve six different tasks, ranging from scanning a system to choosing remediation actions [5]. In 2021, Roth et al. investigated the misconceptions web developers have with *Content Security Policies (CSP)* through a qualitative Zoom interview study with 12 participants [37].

Multiple papers replicated and extended studies by Naiakshina et al. [24, 25]: In 2019, Naiakshina et al. showed that online freelancers behave similarly as students [23]; in 2020, Danilova et al. replicated the original lab study with freelancers and the deception of a real-world project which had a negligible effect [7]; also in 2020, Naiakshina et al. demonstrated that professional developers perform better than students and freelancers [22]. Another study of Votipka et al. conducted semi-structured observational interviews with 16 reverse engineers to understand the reverse engineering process and to improve interactions with reverse engineering tools [50]. Several of these studies involved downloading code and uploading solutions, with no possibility of observing intermediate attempts or behavior.

Browser-Based Studies. A specific type of remote studies utilizes browser-based environments that participants can access via a web browser developed explicitly for the respective study. These are more closely related to OLab.

For example, Yakdan et al. conducted an online experiment to evaluate the usability of different decompilers for reverse engineering with nine professional malware analysts and 21 students [52]. Oliveira et al. conducted an experiment with 109 developers who had to solve six programming puzzles in Java, which include so-called *API blind spots* [31]. The results underline the importance of well-designed APIs, as (security) blind spots reduced the number of functional and secure solutions.

Acar et al. evaluated the usability of different cryptography Python APIs with 256 GitHub developers, who had to solve basic cryptography tasks with the APIs in a web-based study environment [1]. Based on this setup, Gorski et al. conducted another experiment with 53 developers, examining the effect of integrated security advice and warning messages on code security [10]. Furthermore, Fischer et al. evaluated the effect of Google search ranking results on code security and functionality with 410 GitHub developers using the same setup [9]. In another paper, Acar et al. conducted an online experiment with four different security-critical programming tasks (e. g., encryption, password storage) with 307 developers from a GitHub convenience sample to assess the validity of experiments with GitHub users [4]. Based on this study and the previously mentioned one by Acar et al. [1], Stransky et al. presented a browser-based virtual laboratory called *Developer Observatory* and experiences from using it for developer studies [43]. The main idea of *Developer Observatory* is similar to this paper's approach, but limited to languages supported by Jupyter Notebook kernels [15]. OLab follows a more holistic approach, combining multiple different steps (e. g., introduction, consent form, tasks, surveys, information pages) in a single integrated workflow.

To summarize, these publications made significant contributions to our research community, thus underlining the importance of controlled experiments in which software developers, operators, and others solve tasks. Therefore, we derive and evaluate requirements for a remote study platform to facilitate research with such methods.

3 OLab Design and Implementation

In this section, we describe the requirements we identified in previous work, illustrate constructed study workflows for OLab both from a researcher's and participant's point of view, and discuss key features of OLab.

Table 1: Overview over our categorization of related work in the field of developer security.

	Year	Type			Channel			N	Time	Recruitment				Environment				Tools (Explicit Mention)									Data Collection																In OLab		
		Security Analysis	Security Configuration	Development Task	Lab	Remote	Browser Based	Number of participants	Duration in hours	Organisations	Online Mass Recruitment	Online Targeted Recruitment	Local University Students	Windows	Linux	Variable	Unspecified	Git/Gitlab	Custom CA	IDE	Android Studio Plugin	Static Analyzer/Fuzzer	Browser-Based Platform	Android Emulator	Qualtrics Survey	Programming Language[1]	Screen Recordings	Transcripts	Video Recording	Audio Recording	Observation Notes	Participation Diaries	Think Aloud	Source Code	Copy and Paste Events	Eclipse Actions	Config Files	Bash History	Gitlab Issues/Comments	Copy of image	Qualtrics responses	Browser History	Implemented	Possible	Impossible
[38]	2016			●	●	●		156	3x2w			●	●		●			●								C								●										●	
[2]	2016			●	●			54	01:00	●			●		●					●				●		J							●	●								●		●	
[52]	2016	●					●	30	01:00		●		●													C								●									●		
[1]	2017			●			●	256	01:00		●												●			PY								●	●							●	●		
[24]	2017			●	●			20	08:00				●							●						J								●									●		
[27]	2017			●		●		39	01:00		●									●	●					J								●	●								●		
[4]	2017			●			●	307	01:00		●												●			PY								●	●							●	●		
[16]	2017		●		●			35	02:00				●		●				●												●		●				●	●				●		●	
[25]	2018			●	●			40	08:00				●		●											J	●							●	●							●	●		
[31]	2018	●					●	109	>00:20		●		●												●	J				●											●		●		
[12]	2018	●			●			66	00:47				●				●			●						J								●		●							●		
[10]	2018			●			●	53	?		●												●			PY								●	●								●		
[46]	2019		●		●			31	05:00				●		●												●											●		●		●	●		
[5]	2019	●				●		10	?	●							●						●								●		●										●		
[23]	2019			●		●		43	06:30			●														J								●									●		
[50]	2020	●				●		16	01:10		●					●													●	●	●													●	
[7]	2020			●		●		43	72:00			●														J								●											●
[22]	2020			●		●		36	08:00		●															J								●									●		
[30]	2020	●			●			20	80:00	●					●			●								C,PY													●					●	
[41]	2020	●			●			12	01:00			●					●			●						J,C,P	●			●														●	
[47]	2020			●	●			8	00:30	●				●						●																						●	●		
[32]	2021	●				●		38	20:00				●				●					●						●		●		●		●									●		
[37]	2021			●		●		12	01:33		●					●										J,PY,P							●	●									●		
[9]	2021	●					●	410	?		●						●						●			-								●	●								●		

[1] J=Java; PY=Python; C=C/C++; P=PHP

3.1 Identifying Requirements

To identify requirements for the OLab environment, we considered all identified previous work (cf. Section 2). We started with high-level categories (cf. Table 1), collecting prevalent study environments, tools, and approaches. Six researchers created, merged and revised categories jointly and then decided on definitions based on these categories, resulting in our final codebook. Two or more researchers used "iterative categorization" [26] and re-coded all publications using the final codebook, resolving any emerging conflicts immediately, so we refrain from reporting the inter-rater reliability (IRR) [20].

Diverse Study Setups. We identified three different types of tasks for SIWs. 13 studies (54.1%) included security development tasks, referring to the implementation or use of security relevant source code (e. g., studies investigating the use of cryptography libraries). This type of study was most common in our dataset. Less common were 9 security analyses (37.5%), which included tasks such as reverse engineering binaries or finding vulnerabilities in code. In these studies, researchers provided participants with example binaries and tools. Additionally, 2 papers (8.3%) included security configuration studies. They provided participants with a setup that they should configure to be secure, e. g., a server stack. To move such lab studies to an online environment, OLab needs to be capable of handling diverse study setups. These setups include providing and editing source code, configuration files, network connections, and running arbitrary applications.

High Accessibility for Participants. The top 5 studies with most participants (between 156 and 410) all were either remote- or browser-based. Browser-based studies rely on online mass recruitment, using platforms like Amazon MTurk or emails to reach developers globally. 10 studies (41.6%), of that 7 Lab Studies (70%of Lab studies) relied on university students for their sampling, only two of which recruited additional non-student participants to improve their sample diversity and size. To address the limitations of Lab study recruitment and allow for more diverse sampling procedures, OLab should obtain the ease of browser-based studies. It needs to be easy to access using a commodity browser. Furthermore, it should scale to many concurrent participants.

Data Collection. In previous work, researchers collected a wide variety of data from participants, including source code (used by 16, 66.6%) and browser profiles (7 studies, 29.2%). They also tracked copy & paste events (6 studies, 25.0%) and more fine-grained browser or IDE behavior (1 study, 4.2%). Furthermore, they recorded screen and audio (3, 12.5% and 4, 16.7%, respectively). Hence, OLab needs to be able to collect all of the above information.

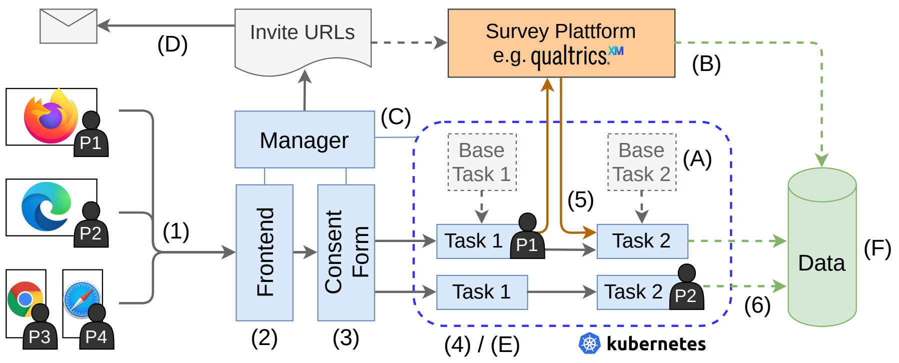


Figure 2: Overview of a typical setup with OLab. Walkthroughs labeled for researchers (A–F) and participants (1–6). See Section 3.2 for an in-depth label description.

3.2 Study Workflow

This section describes the interaction process between OLab, researchers, and study participants. Figure 2 illustrates an example study setup both from a researcher's (A–F) and participant's (1–6) perspective.

Participant Perspective. Steps 1–6 in Figure 2 illustrate the participants' perspective.

(1) **Receive Invite:** Invitees can participate in a study remotely by accessing a (unique) invite URL with a HTML5-capable commodity browser on a desktop or laptop computer, using a sufficiently stable internet connection (validated for 8.0 Mbit/s downlink and 0.8 Mbit/s uplink).

(2) **Landing Page & Consent Form:** After clicking the invite URL, OLab forwards invitees to a landing page showing study information and a consent form (cf. Figure 7a).

(3) **Briefing:** After giving consent, OLab presents participants a full study description, including an introduction to the study environment (cf. Figure 7b).

(4) **Solving Tasks:** Participants are encouraged to work on tasks in full-screen mode, look up the study and task descriptions with a mouse click, skip a current task, or finish the entire study. OLab aims to provide a working experience as close to a regular desktop environment as possible (cf. Figure 7c).

(5) **Survey Questionnaires:** At any point in a study, OLab allows researchers to forward participants to external websites, including surveys (e. g., using Qualtrics).

(6) **Debriefing & Exit:** After solving all tasks, OLab allows researchers to forward participants to an exit survey and a debriefing website.

Overall, we designed and implemented OLab to be unobtrusive, engaging, and fail-safe for participants.

Researcher Perspective. Steps A–F in Figure 2 illustrate the researcher's perspective.

(A) **Setup Study Environment:** During the study setup, researchers can freely choose operating systems, applications, tools, file access, and connection control.

(B) **Setup Tasks & Conditions:** OLab supports within-subjects, between-subjects, and mixed studies. Tasks and conditions can be randomized or arranged using the *Latin squares* method [42].

(C) **Scaling:** OLab is based on a highly scalable *Kubernetes* cluster [17] and allows researchers to run studies in different geographical regions with many concurrent participants to optimize connection speeds and scale available environments.

(D) **Generate Invites:** OLab supports individual invite tokens for participants, forgoing the need to save participants' personally identifiable information (PII). Invite tokens can be used to track participants across other services (e. g., Amazon's MTurk).

(E) **Study Progress:** Researchers can track the study progress and modify and manage scaling options using a dashboard.

(F) **Data Access:** After study completion, researchers can gather the collected data (e. g., specific study metrics, metadata, and questionnaire answers) with a mouse click.

3.3 Key Features

Below, we illustrate key features, and discuss their advantages and limitations.

Common Task Support. OLab supports the automation of common tasks. These include collecting informed consent before starting a study and integration with external tools that provide surveys during and after the tasks. The tool automatically stores collected data on a per task and participant base.

Scalability. The OLab prototype allows scaling resources up and down to adapt to the number of concurrent participants. Furthermore, the prototype allocates resources dynamically for all participants. This scaling is possible as OLab relies on a *Kubernetes* cluster [17] to spin up, secure, scale, and orchestrate study environments. For more technical details, we refer to Appendix A.

Supported Study Types and Tasks. OLab can cover numerous study types and tasks. To refer back to the related work evaluation in Table 1, we commonly identified developer studies utilizing programming tools like IDEs and git (8, 33.3%) which OLab supports. Secure configurations allow for apps requiring restricted or encrypted network access (e. g., git or web servers). Support for volumes by Kubernetes allows for persistent storage across tasks, and GPU support is available [18].

Operating System and Tool Support. Six studies (25.0%) from related work relied on Linux, and one on Windows. Therefore, we decided to support Linux containers mainly.[2] This setup enables customized virtual environments focusing on the applications relevant to a specific study. In addition, the containers can provide environments for all existing programming languages (e. g., Python, Java, and C/C++). OLab supports full desktop environments with pre-installed applications and configuration files for more complex studies.

Data Collection. OLab allows collecting a large variety of different data including source code, configuration files, or other files. Additionally, OLab observes user interactions by recording copy-and-paste events, mouse clicks, or keyboard strokes. These observations can be further complemented with screen recordings. If required, system events can be captured, e. g., kernel events and logs. Researchers can freely configure all of the above data types.

Internet Connectivity. Unsupervised participants using the internet on containers can theoretically access any resources reachable by the parent network, e. g. the university, or cloud infrastructure (depending on the hosting setup). Researchers can address this by using firewalls or proxies provided by Kubernetes for network access. These can be configured as part of the study environment. By default, OLab allows full access to the internet except to other study containers.

Access Control. Since internet connection and other security risks exist with the kinds of setup we provide, we describe measures that we took to allow researchers of OLab to identify individuals misbehaving in the infrastructure: By default, OLab generates secure random tokens. These tokens serve as personalized tickets for participants, which researchers can link to participant profiles (e. g., MTurk profiles). OLab assigns containers to these tokens, tracks timings and container addresses, and participant-specific data, which can serve as a chain of accountability. Hence, researchers can trace potential abuse back to individual participants.

Participants' User Experience. Overall, OLab aims to provide good usability for participants. First, the effort to participate in a study is low, as participants only need a commodity web browser. Second, OLab allows easy navigation through study parts by offering progress indicators (cf. Fig. 7b) and "Start" and "Continue" buttons. Third, participants can access study and task descriptions at any point. Finally, participants that re-access a study after interruption are by default redirected to their current step instead of having to restart or navigate themselves.

Lab Study Support. In addition to using OLab for online studies, researchers can use it in traditional in-person laboratory settings. In that case, the experiment computer can access the OLab frontend, so that OLab provides the automatic study setup and data collection.

4 Evaluation

Overall, we followed an iterative usability evaluation and engineering approach [29]. We focused on participants we could easily approach (e. g., researchers, local CS students) and stopped recruiting once an evaluation step detected no further usability problems. We conducted studies with smaller but increasing sample sizes, instead of one large-scale usability study, following best practices for usability engineering [48]. We think our approach is suitable to provide good usability for OLab.

We conducted three studies, including (1) a cognitive walkthrough with experienced usable security researchers, (2) an evaluation from the participant's perspective, and (3) a comparison to an alternative online study setup. The first study, a cognitive walkthrough with experienced usable security researchers, focused on gaining first insights into participant usability (Section 4.1). The second study focused on a qualitative usability evaluation from the participants' perspective (Section 4.2). Finally, the third study compared OLab to an online task-download study (Section 4.3). While the first two studies are formative, guided studies to collect feedback for an iterative improvement of the OLab, the third was summative and inspired by a study setup from previous work [1]. This study setup allowed us to construct a well-evaluated version of the participant view. We also chose this setup to evaluate two different sets of expert populations: developers and DevOps. Furthermore, the two setups demonstrate the flexibility of OLab regarding different study types and tasks (e. g., programming and system configuration), different requirements for data collection, and a diverse participant pool.

Below, we summarize the ethical aspects of all three studies and provide an overview of our goals for each study, recruitment, participants' demographics, and limitations. Finally, we

[2]For additional OS support, a Windows Docker image (requiring Windows Server with an appropriate license) is available [8].

describe the three studies and their results in detail in the following subsections.

Ethics. Our institution did not require formal Institutional Review Board (IRB) approval for the types of studies we conducted in this work. However, compliance with standard IRB requirements is a focus of OLab. Participants agreed to a consent form modeled after IRB-approved consent forms in previous work [51].

We handled the collected data in our studies under strict German data and privacy protection laws and the European Union General Data Protection Regulation (GDPR). Furthermore, to prevent exposure of any data to third parties, the OLab infrastructure runs entirely self-hosted.

Evaluation Goals. During the studies, we aimed for the following evaluation goals:

1. **EG1: Usability.** How well does the OLab follow common usability guidelines?
2. **EG2: Perception.** How do participants perceive studies using the OLab prototype?
3. **EG3: Limitations.** What problems can occur during the study? What requirements do all participants need to fulfill to use our OLab?
4. **EG4: Comparison.** How do studies with OLab compare to other conventional online study approaches?

For EG1, we consider usability goals and rules to be generally unknown to participants. Therefore, we decided to evaluate EG1 by conducting expert walkthroughs (Section 4.1). EG2 and EG3 are the focus of a guided DevOps study (Section 4.2). This study measures physical requirements like hardware and Internet bandwidth, but also collects feedback on the perception of participants regarding the study and uncovers misconceptions. In the third study, we focus on the comparison to other study types as detailed in EG4 (Section 4.3). This unsupervised study identified a few more technical limitations and usability challenges that did not come up in the previous supervised studies.

Recruitment and Demographics. Below, we describe all three studies' recruiting process and participant demographics. Table 2 provides an overview of the collected demographics. For most demographic questions, we allowed multiple answers (cf. replication package in Section 5).

For the cognitive walkthroughs, we recruited four experienced usable security researchers (Section 4.1). The experts were not involved in the development or previous test phases of OLab. All participants have a Master's degree or Ph.D. in computer science. The average experience in usability and conducting studies was 3.88 years (median = 3.5). We consider them all experienced usability security researchers, as they actively research and conduct studies in usable security and privacy.

For the second study (Section 4.2), we recruited nine experienced DevOps. We chose three recruitment channels: students from our university that worked in small- and medium-sized enterprises (2 participants), an online forum for DevOps (1 participant), and posts on four Subreddits related to DevOps (6 participants). Three DevOps had secondary education, three had a Bachelor's degree, and three had a Master's degree or a Ph.D.

Table 2: Demographics for valid participants from all three studies. Omitting "Don't know"/"Don't want to answer" answers.

	Expert Walkthrough	Guided Study	Comparison Experiment
Participants			
Started	4	9	23
Finished	4	9	19
Valid ($n =$)	4	9	16
Gender			
Male	50.0%	100.0%	93.8%
Female	50.0%	0.0%	6.2%
Not M/F (Free Text)	0.0%	0.0%	0.0%
Education			
Secondary	0.0%	33.3%	62.5%
Bachelor's	0.0%	33.3%	37.5%
Master's or higher	100.0%	33.3%	0.0%
Age (in years)			
Median	27.5	29.0	22.0
Mean (μ)	27.75	29.88	22.06
Std. dev. (σ)	2.5	6.33	1.57
Relevant Experience (in years)†			
Median	3.5	-	1.5
Mean (μ)	3.88	-	2.27
Std. dev. (σ)	2.32	-	2.05

† Conducting studies and Python programming respectively.

For the third study (Section 4.3), we recruited a sample of 23 computer science students from our university. The study took two hours, and we compensated participants with €100. Four participants dropped out during the experiment, and 19 participants completed the study. We excluded another three participants due to longer breaks. Hence, 16 valid participants completed the study overall. Most of them were male (93.8%; 15). The majority studied for a Bachelor's degree (62.5%; 10), while the remaining strived for a Master's degree (37.5%; 6). The average Python programming experience was 2.27 years (median = 1.5).

Limitations. Our studies share limitations common among qualitative and task-based studies, like an opt-in bias concerning participants' voluntary participation. We recruited people in a snowball sampling from our network for the cognitive walkthroughs. While we believe these are appropriate professionals, they may be biased towards our team and tool. We, therefore, refrain from evaluating and including their usability ratings beyond the walkthroughs themselves and exclude them from further conclusions for EG1, the usability of our tool. We recruited students to perform tasks that might not represent real-world developers in the comparison study. However, students were used in previous studies and found to be acceptable proxies for professional software developers [4, 39, 44] for the type of study tasks we performed [1]. We explicitly pointed participants to the fact that we aimed to collect self-reported usability assessments for the infrastructure and not single components of the studies (e. g., the IDE we provided). While this worked smoothly for the supervised cognitive walkthroughs to help participants, we could not intervene in the comparison study. Some participants might have misunderstood our framing or explanations, as is natural in meta-evaluation studies. We evaluated descriptions in our pilot run with students to minimize this risk.

4.1 Cognitive Walkthrough

After developing and piloting the OLab environment, we evaluated the usability from four usability experts' points of view via cognitive walkthroughs.

Methodology. We conducted four cognitive walkthroughs in July 2020 during the COVID-19 pandemic using an online meeting software with screen and audio sharing. Two researchers accompanied and recorded each walkthrough with the participants' consent for later transcription. The experts provided usability feedback using different operating systems (macOS, Windows, and Linux) and web browsers (Chrome/Chromium and Firefox). Before the cognitive walkthroughs, we asked the participants to watch an animated video that explained and reminded them about Nielsen's ten usability heuristics [28]. We also told the participants to write down bullet points for each heuristic to remember them during the walkthrough.

We asked the experts to perform a study in the role of participants, except they did not have to solve the provided programming tasks. Instead, the experts should focus on the usability of the OLab prototype. To guide the walkthrough, we identified ten typical workflows for participants within the study environment. The experts had to pass each workflow step to finish the walkthrough successfully. During the walkthrough, we collected usability feedback based on Nielsen's heuristics and further feedback on the user interface (UI) and experience (UX). After completing the walkthrough, we discussed the comments and feedback and implemented the required changes before the following walkthrough.

Results. Table 3 provides detailed background information of the recruited experts, including both their study background and experience within their research field. The experts were generally optimistic about OLab's usability. Each expert completed all walkthrough steps without any significant issues. As Brooklyn summarized it:[3] "*Everything was running fine, without any problems. [...] I didn't have any lag, it was like I was on my own system. [...] I didn't even notice that I was not working on my own computer.*" (Brooklyn).

[3] We translated all quotes in this paper from German to English.

Most expert feedback was on UI and UX. For example, we received feedback to name the buttons and links clearer and more consistent. Brooklyn mentioned that clicks within a popup should not close it and that all UI elements should receive mouseover tooltips or have their text improved to enhance clarity for participants during a study. As a result, we also added a help button to the sidebar (cf. Figure 7). Further on, Charlie suggested better framing of the study process by initially displaying the number of tasks that participants are going to do and generally improving the wording for indicating the study progress. Moreover, Dakota suggested adding functionality for participants to review content from previous pages, e. g., the consent form or introduction videos, and the addition of an information graphic introducing participants to the study scenario.

The remaining feedback focused on the survey's content or structure. Here, some clarifications targeted the consent form (Brooklyn). We implemented a redesign for questions in Qualtrics, so they match the overall layout and design of OLab (Ash). Dakota further mentioned that the consent form should be simplified to reduce cognitive load on participants and to include missing information regarding speed tests we are running in the background. Charlie also noted that OLab should show the consent form as the first item within the study environment.

4.2 Guided DevOps Study

In the second study, we evaluated the usability and participant interaction of OLab in a study with nine DevOps from small and medium-sized enterprises (SMEs). The study structure derived from a different project with DevOps from a local meetup. We then conducted the study within OLab with an additional focus on the usability of OLab. We observed the participants in a think-aloud study. These requirements are ideal for a functionality test since we could ask about the participants' perception of specific OLab prototype features during the study and observe and assist with issues that occurred to improve the prototype iteratively. Therefore, we designated this study as "guided".

Scenario & Task. In a hypothetical scenario, we asked participants to imagine they were leading a DevOps team in a company that experienced a customer data leak recently. We required them to investigate how the leaks happened and who was responsible for them. We asked participants to express their thoughts in a think-aloud setup during the study. Think-aloud included talking about their experiences in similar scenarios, questions they would ask colleagues in the imaginary company, tools they would typically use, their experience with the tools we provided, and their suspicions on what caused the data leaks. After identifying the leaks and their root causes, we asked the participants how they would resolve the found issues in their company. We also asked for general feedback regarding OLab.

Table 3: Detailed overview of experts, their background, experience of conducting studies, and their main operating system.

Alias[†]	OS	Study Background	Study Experience
Ash	Linux	Online developer studies with a focus on programming tasks.	4 years
Brooklyn	Linux	End-user studies with a focus on crowd-worker platforms.	3 years
Charlie	macOS	Both developer and end-user studies, with a focus on lab experiments and surveys.	7 years
Dakota	Windows	Developer studies with a focus on qualitative interviews.	1.5 years

[†] Gender-neutral aliases assigned alphabetical to all experts.

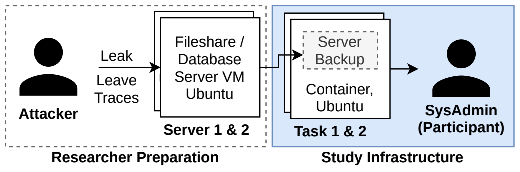


Figure 3: Overview over the guided usability study's setup.

Figure 3 provides an overview of the task creation process. Since we aimed to test the participants' abilities to manage security incidences in a company setting, we provided a virtual server backend within OLab. We set up two containers, a database, and a file server. We then simulated a company-internal attacker with access to the server that used social engineering to leak company-internal information. The resulting logs and system states were then backed up and included in a Docker image that provided a visual interface and tools to inspect the backups.

We added a hint regarding emails from the attacker to the admin using the second container, pointing towards the internal attackers, as an experiment condition. Hence, participants might have an easier time identifying the exact circumstances of the attack in this condition. OLab automatically assigned the condition to half of all participants. Within the study, conditions and the task order were randomized.

Two authors supervised the participants virtually during the study, took notes, and answered scenario-specific questions that participants asked. Participants were asked to screen-share the tab containing OLab, which we recorded to complement our notes (cf. replication package, Section 5).

Coding and Evaluation. Using the recorded videos and notes, two authors coded participants' free-text responses in an "iterative categorization" [26] approach. The authors focused on the advantages and disadvantages the participants reported while interacting with OLab and their general survey sentiment. We focused on these general categories because a notable amount of feedback came up while participants were working on the tasks, not as a result of individual questions in the post-survey. After assigning initial codes to all feedback, both authors reviewed the resulting coding and resolved conflicts in a consensus discussion or introduced new codes. When new codes emerged, the already coded videos were revisited and re-coded. Since both researchers coded all participant responses with immediate conflict resolution, we refrain from reporting the inter-rater reliability (IRR) [20].

Results. In general, OLab was well-received by all nine participants, while only some minor problems occurred that were related to OLab. Seven participants (P1, P3, P5–9) explicitly mentioned that they were impressed by OLab and its workflow. From our observations, the prototype was very fluid for all participants. Four of them (P1, P5, P8, P9) mentioned low latency, e. g., "*It worked flawlessly. I was very surprised that this works so well in the browser.*" (P5). Only P7 reported minor latency issues due to a low-quality mobile 4G/LTE connection. Other positive aspects mentioned by the participants were full functionality despite the use of ad-blockers (P5), the internet access with the possibility to install arbitrary additional software (P5), and that it works with non-German keyboard layouts (P7). Additionally, participants liked the visual appearance. To cite P1: "*The tools we are working with are modern, fast, looking good, I like that very much.*" (P1).

The most common limitation, mentioned by seven participants (P2, P3, P5–P9), were differences between the study's infrastructure and the users' typical environment. For example, as the environment in OLab cannot be customized for every user, the participants might miss any custom programs they like to use. As P6 put it: "*So I have some standard suite of programs that I have installed [. . .]. Well, you cannot take that for given. That would be [. . .] nice-to-have and not absolutely necessary.*" (P6).

Besides that, two participants encountered technical limitations. P2 tried to change the keyboard layout, but this is technically impossible during the study, and can only be changed when initializing the VNC connection. In addition, P2 and P3 noted the unavailability of `chroot`; this is disabled by default for security reasons. However, in researcher-supervised studies this could be enabled. Two participants (P1, P4) reported problems that were not related to OLab.

We queried participants on how they would solve the tasks in their everyday setup, i. e., not in a study within OLab. P2, P3, P4, and P8 reported differences that were not related to OLab. P5 explicitly mentioned that he would do the tasks as done in the study. The other four participants (P1, P6, P7, P9) highlighted that they would incorporate some form of social interaction during the tasks in a real-world scenario, e. g., contacting and talking to colleagues. We consider this to be out of scope for OLab, as it is impossible to simulate this social interaction in a software tool.

We asked all participants for additional features they would appreciate. They mentioned missing tools that we could set up in future DevOps studies. P3, however, proposed that OLab should show the correct solution for self-evaluation after completing a task. We consider implementing this as an optional feature for future studies. The qualitative coding results can be found in the appendix (cf. Table 5).

4.3 Comparison Experiment

In this comparison experiment, we compare a study setup using OLab with a more conventional browser study regarding feasibility and usability. We based this experiment on the browser-based study setup of Acar et al., which consists of a developer study with two programming tasks that test cryptographic APIs and their documentation for usability [1]. This setup provides a good fit for a virtual study environment and a suitable starting point for a first unsupervised study with the OLab prototype.

Study Setup. We started with a Ubuntu 20.04 Docker container similar to the previous study setup. In that container, we installed Python including PyCryptodome [34] and the IDE PyCharm [33]. In PyCharm, we set up a Python project consisting of dependencies, a virtual environment, and a skeleton containing pre-written function names and comments with precise task descriptions. The original study relied on a browser-based approach using Jupyter Notebooks [14], likely due to the limitation that a fully virtualized setup containing an IDE was not available.

We decided to have each participant perform one task in a more conventional download setting for the comparison. We provided a website with the same structure, text, and study flow as in OLab. However, instead of redirecting to the virtualized environment, we provided them with a page to download the PyCharm project and upload their solution after completing the task on their computer.

We piloted the study internally and with students recruited in a snowball sample to evaluate task description clarity.

To ensure fair compensation and comparable internal validity, we instructed all participants to stop after at most 60 minutes per task and use PyCharm as a common development environment for the download condition. In the OLab prototype, we built the same setup based on an Ubuntu container image. It includes PyCharm with dependencies set up and the Python file containing the task opened in the IDE. Additionally, Chromium starts with the crypto API's documentation opened in a new tab. OLab collected the browser history and source code of the PyCharm project for our evaluation. An overview of the study setup can be found in Figure 4.

Task Setup. In our scenario, the developers had to implement (1) secure communication using an asymmetric encryption scheme of their choice and (2) encrypted storage using a

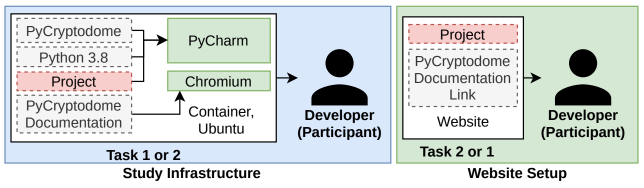


Figure 4: Overview over the comparison study's setup.

symmetric scheme. We required participants to implement this using the PyCryptodome library, which is a fork of the original PyCrypto library used by Acar et al. [3]. To combat the effects of learnability, we randomized which task our participants performed in which environment (download and OLab). We also randomized the order of study environments, i. e., half of the participants had to complete the download task first and use the OLab prototype second, with the other half completing those vice versa.

After each task, participants filled a short survey regarding feedback about the study environment and the cryptographic task. As mentioned in the limitations, we tried to differentiate between the environment and the task through diagrams and descriptions. The survey collected the *System Usability Scale* (SUS) score by Brooke [6] and the *Net Promoter Score* (NPS), a more industrially used usability score for product recommendation rates by Reichheld [36]. We decided to collect a self-assessment on security and functionality in line with the original study, but exclude other factors to prevent participant exhaustion.

Evaluation. To rate results for their security and functionality, we used an open-coding approach with two researchers to review the source code that participants submitted in OLab and the download environment. First, two authors executed the code to determine functionality. Then they rated all submissions for security, grading factors like usage of weak algorithms and insecure password generation. Finally, the coders discussed individual ratings and resolved conflicts to arrive at a complete security rating.

Since this study was unsupervised, we could not collect the degree of qualitative feedback obtained in the previous two studies. To alleviate that, we contacted participants individually after the study and asked a few follow-up questions regarding the issues and differences. For these responses, we used an "iterative categorization" [26] approach; two researchers classified the types of issues, advantages, and disadvantages participants reported on. As both researchers coded all source code and all participant responses and immediately resolved conflicts, we refrain from reporting the inter-rater reliability (IRR) [20].

Task Results. Overall, 24 out of 32 solutions were functional and secure according to our coding. Out of those, 9 of 16 solutions were secure when worked on in OLab, while 15 out of 16 were securely provided via our download environment.

We noticed that this difference comes from the second task (symmetric cryptography). The timings indicate that participants spend 60 minutes on that task using OLab. We asked the participants why they did not complete the second task. The participants who responded found it more complicated than the first task and noted that the clearly shown "skip"-button provided in OLab's interface (cf. Figure 7c) in combination with the reminder to only spend 60 minutes on this task lead to submitting earlier when using OLab. These factors were not as present in the download condition, since there was no "skip"-button. Furthermore, participants might not see the time limit when using the website because they opened the documentation in a new tab on the same window. These responses indicated that the second task was infeasible within the provided time for students having no experience in cryptography. We think the qualitative results produced by this task are still valuable for the evaluation, as they indicate higher compliance with the study protocol of this unsupervised study when participants used the OLab. Studies within OLab need to set fair requirements and cannot rely on participants ignoring time constraints or the study conditions.

We also asked which editor participants used in both conditions in the survey, which revealed another compliance difference. All participants used PyCharm within OLab – likely because it was already set up and automatically started. However, 5 used other editors (4 VSCode, 1 VIM) in the download condition. We explicitly instructed participants to use PyCharm with our PyCharm project. Therefore, this difference represents a threat to internal validity not present in OLab. While this demonstrates the more challenging enforcement of requirements in download studies, researchers may offer multiple editors in OLab to accommodate for preferred software. However, this would increase the required time for study setup and piloting.

Usability Rating. We asked participants to rate each environment using the SUS score and the NPS. Regarding the SUS score, we had participants rate the environment after completing each task. The environment using OLab received an average SUS score of 80.0, which corresponds to a Grade A– according to Sauro and Lewis [40], with the download environment receiving a SUS score of 78.125, classified as B+. While the ratings are limited to the explanatory power of the study, they indicate at least comparable usability of OLab and the download environment. The NPS for OLab did not result in more promoters, but is equal to the download environment. We include an overview of both scores in Table 4.

This is further reflected in the participants' preference for the environments. 9 preferred OLab while 7 preferred the download environment. As reasons for preferring the download study, three participants mentioned bandwidth limitations that lead to unresponsive or unstable experiences with OLab. While we found that OLab runs fine with typical desktop bandwidths (starting at around 0.5 Mbit/s), we assume higher round-trip times (RTTs)/pings cause a noticeably slower experience compared to a native interface.

Table 4: SUS and NPS scores for both OLab and the download condition.

Score	OLab	Download Condition
SUS Score (mean)	80.0 (A-)	78.125 (B+)
NPS Promotors	10	10
NPS Passivers	3	3
NPS Detraktors	3	3

The participants provided different reasons for preferring the download environment. To quote a participant: "*I can use my own IDE, which is adapted to my requirements. Also, I can open the documentation on a second screen, making research and reading easier.*" (P2). This preference is in line with the finding that 5 participants used a different IDE than specified for solving the task. Multiscreen support is currently impossible with OLab due to being limited to a single browser window. However, these advantages also affect the internal validity of the study results in the download study. Furthermore, they are only present for studies where the task can be downloaded to a participant's machine, not in lab studies or studies using server environments like our previous setup.

When participants stated to prefer OLab over the download study, all 9 participants mentioned the much lower setup efforts as the main reason. A participant stated:

> "*On my desktop PC, I had problems importing* `crypto`*. Therefore, I had to switch to my laptop, on which working was much harder. This resulted in a lot of time spent on a problem that I didn't have in the virtual desktop environment. In this environment, everything was prepared and I could immediately start working. There was also less distraction by open tabs or pop-up messages.*" — P3

The virtualized environment within OLab can lower entry barriers for participants and reduce the time participants spend on tasks while still providing them with a fully-featured development environment that reflects their actual environment, even if customization might be missing.

To compare the timings for both studies, we asked participants about the perceived time spent on preparation, from 0 (very low) to 6 (very high). We found that people rated OLab 0.31 on average, indicating a lower setup time compared to the download environment that participants rated 2.13 on average. This confirms the suspected advantage in participants' preparation time for studies using OLab.

5 Discussion

Below, we discuss our results in the context of the evaluation goals we presented in Section 4 and discuss how the study results address them. We also elaborate on future directions for virtual study environments we derive from our results and how we plan to implement them.

EG1: Usability. After implementing the feedback we collected during the cognitive walkthrough, we could improve the usability of the OLab environment.

We found the OLab prototype to be easy to use for all participants (cf. the SUS scores in Section 4.3), with a low entry barrier, and safe to use in all scenarios we provided since it automatically stored participant results without storing results manually. The lower preparation time through pre-setup dependencies that participants reported in our comparison study demonstrated how this approach could be more efficient than conventional approaches. In our comparison study, participants were willing to spend more time with the tasks in their own environment, leading to more complete solutions. We believe this can be addressed through smaller tasks or more straightforward instructions.

In summary, we think that our approach can indeed fulfill the high accessibility requirements that we identified in Section 3.1.

EG2: Perception. Even when encountering minor latency issues or unknown setups in our study environments, participants remarked on the smooth study procedure possible through OLab. Participants also mentioned the low entry barrier through the provided tooling and setup as an advantage. In addition, the setup allowed us to test an unconventional setup in the form of servers that participants had to analyze for security issues. In supervised studies structured like interviews or remote think-alouds like our second study, we can even allow participants to use root access on the machines and install their own applications to complement the setups we provide them.

EG3: Limitations. We also encountered a few limitations, mostly related to security. One of these is the ability to use features like `chroot`, KVM, and `systemctl` that require privilege escalation beyond what is considered safe in a container. These can be ignored to some extent in supervised studies, where a researcher can ensure that participants do not abuse permissions on the container and therefore can declare the containers to be privileged. However, this poses a security risk for the entire infrastructure, including other participants and the host systems, when done without supervision.

Finally, latency is a significant limitation of the environment, and participants with a high connection latency reported difficulties using the OLab environment.

EG4: Comparison. From our previous findings, we conclude that in comparison to more conventional setups, the OLab environment provides the option to enforce higher internal validity at the cost of customization for participants. In our comparison study, we also found that the time spent on our OLab was lower on average. We assume that when providing participants the time to customize their setup in the OLab, this advantage will vanish. In general, providing participants with a fully working setup in our OLab environment will always be faster than download tasks or tasks requiring setup time beyond reading the task description. We hope to capitalize on this advantage to conduct new types of studies that were previously hard or even infeasible to conduct online.

Replication. To allow for better replication of our work, we make the following items available as part of a replication package [13]: We provide the study protocols for the cognitive walkthrough, the guided and the comparison study including the study scenarios, the tasks descriptions, between-task surveys, and end surveys.[4]

Future Work. In future work, we plan to improve the usability of the researcher's web interface to illustrate the current state of a study, and to manage participants and study instances.

We plan to evaluate OLab in multiple large-scale studies, test more edge cases, and improve flexibility. Furthermore, support for complex features like interaction between participants or with researchers can expand the scope of possible studies for OLab.

6 Conclusion

In conclusion, we identified common requirements for lab-like studies with SIWs. We designed, implemented and evaluated the OLab environment as a novel approach to conduct lab-like studies online, and found that:

1. OLab can provide high usability for participants in online studies while enabling complex study setups such as programming and server administration studies, as evaluated through our expert walkthrough (cf. Section 4.1) and through the SUS scores (cf. Sections 4.2 & 4.3).
2. OLab handles typical research tasks like data and consent form collection and study parameters like task order, conditions, and the inclusion of external questionnaires, offering a flexible setup for complex studies to researchers (cf. Section 3.3)
3. OLab provides higher internal validity than approaches that involve external working environments, both regarding task compliance and regarding tools and environmental variables used (cf. Section 4.3).

Based on our results, we consider OLab to be a highly functional prototype that we plan to expand on for future real-world studies. Although it is not yet fit for a general release, we formally invite interested researchers to contact us regarding the collaboration and extension of OLab.

[4]The replication package is also available via this paper's accompanying website: https://publications.teamusec.de/2022-soups-olab/.

References

[1] Yasemin Acar, Michael Backes, Sascha Fahl, Simson Garfinkel, Doowon Kim, Michelle L. Mazurek, and Christian Stransky. “Comparing the Usability of Cryptographic APIs”. In: *Proc. 38th IEEE Symposium on Security and Privacy (SP’17)*. IEEE, 2017.

[2] Yasemin Acar, Michael Backes, Sascha Fahl, Doowon Kim, Michelle L Mazurek, and Christian Stransky. “You Get Where You’re Looking For: The Impact of Information Sources on Code Security”. In: *Proc. 37th IEEE Symposium on Security and Privacy (SP’16)*. IEEE, 2016.

[3] Yasemin Acar, Sascha Fahl, and Michelle L. Mazurek. “You are Not Your Developer, Either: A Research Agenda for Usable Security and Privacy Research Beyond End Users”. In: *Proc. 2016 IEEE Secure Development Conference (SecDev’16)*. IEEE, 2016.

[4] Yasemin Acar, Christian Stransky, Dominik Wermke, Michelle L. Mazurek, and Sascha Fahl. “Security Developer Studies with GitHub Users: Exploring a Convenience Sample”. In: *Proc. 13th Symposium on Usable Privacy and Security (SOUPS’17)*. USENIX Association, 2017.

[5] Muharrem Aksu, Enes Altuncu, and Kemal Bicakci. “A First Look at the Usability of OpenVAS Vulnerability Scanner”. In: *Proc. Workshop on Usable Security (USEC’19)*. The Internet Society, 2019.

[6] John Brooke. “SUS: a retrospective”. In: *Journal of Usability Studies* 8.2 (2013), pp. 29–40.

[7] Anastasia Danilova, Alena Naiakshina, Johanna Deuter, and Matthew Smith. “Replication: On the Ecological Validity of Online Security Developer Studies: Exploring Deception in a Password-Storage Study with Freelancers”. In: *Proc. 16th Symposium on Usable Privacy and Security (SOUPS’20)*. USENIX Association, 2020.

[8] *Docker images of Windows*. https://hub.docker.com/_/microsoft-windows.

[9] Felix Fischer, Yannick Stachelscheid, and Jens Grossklags. “The Effect of Google Search on Software Security: Unobtrusive Security Interventions via Content Re-Ranking”. In: *Proc. 28th ACM Conference on Computer and Communication Security (CCS’21)*. ACM, 2021.

[10] Peter Leo Gorski, Luigi Lo Iacono, Dominik Wermke, Christian Stransky, Sebastian Möller, Yasemin Acar, and Sascha Fahl. “Developers Deserve Security Warnings, Too: On the Effect of Integrated Security Advice on Cryptographic API Misuse”. In: *Proc. 14th Symposium on Usable Privacy and Security (SOUPS’18)*. USENIX Association, 2018.

[11] Matthew Green and Matthew Smith. “Developers are Not the Enemy!: The Need for Usable Security APIs”. In: *IEEE Security & Privacy* 14.5 (2016), pp. 40–46.

[12] Norman Hänsch, Andrea Schankin, Mykolai Protsenko, Felix Freiling, and Zinaida Benenson. “Programming Experience Might Not Help in Comprehending Obfuscated Source Code Efficiently”. In: *Proc. 14th Symposium on Usable Privacy and Security (SOUPS’18)*. USENIX Association, 2018.

[13] Nicolas Huaman, Alexander Krause, Dominik Wermke, Jan H. Klemmer, Christian Stransky, Yasemin Acar, and Sascha Fahl. *Replication Package: “If You Can’t Get Them to the Lab: Evaluating a Virtual Study Environment with Security Information Workers”*. https://doi.org/10.25835/spxeaic7. 2022.

[14] *Jupyter Notebook*. http://jupyter.org/. visited. Nov. 2016.

[15] Jupyter Project. *Jupyter Notebook Kernels*. https://github.com/jupyter/jupyter/wiki/Jupyter-kernels.

[16] Katharina Krombholz, Wilfried Mayer, Martin Schmiedecker, and Edgar Weippl. ““I Have No Idea What I’m Doing” - On the Usability of Deploying HTTPS”. In: *Proc. 26th Usenix Security Symposium (SEC’17)*. USENIX Association, 2017.

[17] *Kubernetes*. https://kubernetes.io/.

[18] *Experimental Kubernets GPU support*. https://kubernetes.io/docs/tasks/manage-gpus/scheduling-gpus/.

[19] Deepak Kumar, Patrick Gage Kelley, Sunny Consolvo, Joshua Mason, Elie Bursztein, Zakir Durumeric, Kurt Thomas, and Michael Bailey. “Designing Toxic Content Classification for a Diversity of Perspectives”. In: *Proc. 17th Symposium on Usable Privacy and Security (SOUPS’21)*. USENIX Association, 2021.

[20] Nora McDonald, Sarita Schoenebeck, and Andrea Forte. “Reliability and Inter-Rater Reliability in Qualitative Research: Norms and Guidelines for CSCW and HCI Practice”. In: *ACM on Human-Computer Interaction* 3.CSCW (2019).

[21] *minikube*. https://minikube.sigs.k8s.io/.

[22] Alena Naiakshina, Anastasia Danilova, Eva Gerlitz, and Matthew Smith. “On Conducting Security Developer Studies with CS Students: Examining a Password-Storage Study with CS Students, Freelancers, and Company Developers”. In: *Proc. CHI Conference on Human Factors in Computing Systems (CHI’20)*. ACM, 2020.

[23] Alena Naiakshina, Anastasia Danilova, Eva Gerlitz, Emanuel von Zezschwitz, and Matthew Smith. “"If You Want, I Can Store the Encrypted Password": A Password-Storage Field Study with Freelance Developers”. In: *Proc. CHI Conference on Human Factors in Computing Systems (CHI'19)*. ACM, 2019.

[24] Alena Naiakshina, Anastasia Danilova, Christian Tiefenau, Marco Herzog, Sergej Dechand, and Matthew Smith. “Why Do Developers Get Password Storage Wrong?: A Qualitative Usability Study”. In: *Proc. 24th ACM Conference on Computer and Communication Security (CCS'17)*. ACM, 2017.

[25] Alena Naiakshina, Anastasia Danilova, Christian Tiefenau, and Matthew Smith. “Deception Task Design in Developer Password Studies: Exploring a Student Sample”. In: *Proc. 14th Symposium on Usable Privacy and Security (SOUPS'18)*. USENIX Association, 2018.

[26] Joanne Neale. “Iterative categorization (IC): a systematic technique for analysing qualitative data”. In: *Addiction* 111.6 (2016), pp. 1096–1106.

[27] Duc Cuong Nguyen, Dominik Wermke, Yasemin Acar, Michael Backes, Charles Weir, and Sascha Fahl. “A Stitch in Time: Supporting Android Developers in Writing Secure Code”. In: *Proc. 24th ACM Conference on Computer and Communication Security (CCS'17)*. ACM, 2017.

[28] Jakob Nielsen. “Enhancing the Explanatory Power of Usability Heuristics”. In: *Proc. SIGCHI Conference on Human Factors in Computing Systems (CHI'94)*. ACM, 1994.

[29] Jakob Nielsen. “The usability engineering life cycle”. In: *Computer* 25.3 (1992), pp. 12–22.

[30] Timothy Nosco, Jared Ziegler, Zechariah Clark, Davy Marrero, Todd Finkler, Andrew Barbarello, and W. Michael Petullo. “The Industrial Age of Hacking”. In: *Proc. 29th Usenix Security Symposium (SEC'20)*. USENIX Association, 2020.

[31] Daniela Seabra Oliveira et al. “API Blindspots: Why Experienced Developers Write Vulnerable Code”. In: *Proc. 14th Symposium on Usable Privacy and Security (SOUPS'18)*. USENIX Association, 2018.

[32] Stephan Plöger, Mischa Meier, and Matthew Smith. “A Qualitative Usability Evaluation of the Clang Static Analyzer and libFuzzer with CS Students and CTF Players”. In: *Proc. 17th Symposium on Usable Privacy and Security (SOUPS'21)*. USENIX Association, 2021.

[33] *PyCharm*. https://www.jetbrains.com/pycharm.

[34] *PyCryptodome*. http://pycryptodome.readthedocs.io.

[35] Hirak Ray, Flynn Wolf, Ravi Kuber, and Adam J. Aviv. “Why Older Adults (Don't) Use Password Managers”. In: *Proc. 30th Usenix Security Symposium (SEC'21)*. USENIX Association, 2021.

[36] Frederick F. Reichheld. “The One Number You Need to Grow”. In: *Harvard Business Review Press* 81.12 (2003), pp. 46–55.

[37] Sebastian Roth, Lea Gröber, Michael Backes, Katharina Krombholz, and Ben Stock. “12 Angry Developers - A Qualitative Study on Developers' Struggles with CSP”. In: *Proc. 28th ACM Conference on Computer and Communication Security (CCS'21)*. ACM, 2021.

[38] Andrew Ruef, Michael Hicks, James Parker, Dave Levin, Michelle L. Mazurek, and Piotr Mardziel. “Build It, Break It, Fix It: Contesting Secure Development”. In: *Proc. 23nd ACM Conference on Computer and Communication Security (CCS'16)*. ACM, 2016.

[39] Iflaah Salman, Ayse Tosun Misirli, and Natalia Juristo. “Are Students Representatives of Professionals in Software Engineering Experiments?” In: *Proc. 37th IEEE/ACM International Conference on Software Engineering (ICSE'15)*. IEEE, 2015.

[40] Jeff Sauro and James R Lewis. *Quantifying the User Experience: Practical Statistics for User Research*. Morgan Kaufmann Publishers, 2016.

[41] Justin Smith, Lisa Nguyen Quang Do, and Emerson Murphy-Hill. “Why Can't Johnny Fix Vulnerabilities: A Usability Evaluation of Static Analysis Tools for Security”. In: *Proc. 16th Symposium on Usable Privacy and Security (SOUPS'20)*. USENIX Association, 2020.

[42] Springer Verlag GmbH, European Mathematical Society. *Latin Square*. http://encyclopediaofmath.org/index.php?title=Latin_square&oldid=47587. Accessed on 2022-02-18.

[43] Christian Stransky et al. “Lessons Learned from Using an Online Platform to Conduct Large-Scale, Online Controlled Security Experiments with Software Developers”. In: *Proc. 10th USENIX Workshop on Cyber Security Experimentation and Test (CSET'17)*. USENIX Association, 2017.

[44] Mikael Svahnberg, Aybüke Aurum, and Claes Wohlin. “Using Students as Subjects - an Empirical Evaluation”. In: *Proc. Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'08)*. ACM.

[45] Christian Tiefenau, Maximilian Häring, Katharina Krombholz, and Emanuel von Zezschwitz. “Security, Availability, and Multiple Information Sources: Exploring Update Behavior of System Administrators”. In: *Proc. 16th Symposium on Usable Privacy and Security (SOUPS’20)*. USENIX Association, 2020.

[46] Christian Tiefenau, Emanuel von Zezschwitz, Maximilian Häring, Katharina Krombholz, and Matthew Smith. “A Usability Evaluation of Let’s Encrypt and Certbot: Usable Security Done Right”. In: *Proc. 26th ACM Conference on Computer and Communication Security (CCS’19)*. ACM, 2019.

[47] Harshal Tupsamudre, Monika Sahu, Kumar Vidhani, and Sachin Lodha. “Fixing the Fixes: Assessing the Solutions of SAST Tools for Securing Password Storage”. In: *Proc. 24th International Conference on Financial Cryptography and Data Security (FC’20)*. Springer, 2020.

[48] Carl W. Turner, James R. Lewis, and Jakob Nielsen. “Determining Usability Test Sample Size”. In: *International Encyclopedia of Ergonomics and Human Factors*. 2nd ed. Vol. 3. CRC Press, 2006, pp. 3084–3088.

[49] Daniel Votipka, Kelsey R. Fulton, James Parker, Matthew Hou, Michelle L. Mazurek, and Michael Hicks. “Understanding security mistakes developers make: Qualitative analysis from Build It, Break It, Fix It”. In: *Proc. 29th Usenix Security Symposium (SEC’20)*. USENIX Association, 2020.

[50] Daniel Votipka, Seth Rabin, Kristopher Micinski, Jeffrey S. Foster, and Michelle L. Mazurek. “An Observational Investigation of Reverse Engineers’ Processes”. In: *Proc. 29th Usenix Security Symposium (SEC’20)*. USENIX Association, 2020.

[51] Dominik Wermke, Nicolas Huaman, Yasemin Acar, Brad Reaves, Patrick Traynor, and Sascha Fahl. “A Large Scale Investigation of Obfuscation Use in Google Play”. In: *Proc. 34th Annual Computer Security Applications Conference (ACSAC’18)*. ACM, 2018.

[52] K. Yakdan, S. Dechand, E. Gerhards-Padilla, and M. Smith. “Helping Johnny to Analyze Malware: A Usability-Optimized Decompiler and Malware Analysis User Study”. In: *Proc. 37th IEEE Symposium on Security and Privacy (SP’16)*. IEEE, 2016.

A Technical Details of OLab

OLab’s Kubernetes cluster runs entirely self-hosted on the researchers’ servers. This setup provides maximum security and data protection for participant data – without any third party involved. For a technical overview, see Figure 8.

Depending on the number of participants, OLab supports other deployment options. For minimal setups or testing purposes, *minikube* [21] requires only a single machine. In studies that exceed the researchers’ server resources, it is possible to host and operate OLab within a Kubernetes cloud environment, e. g., Amazon Web Services (AWS).

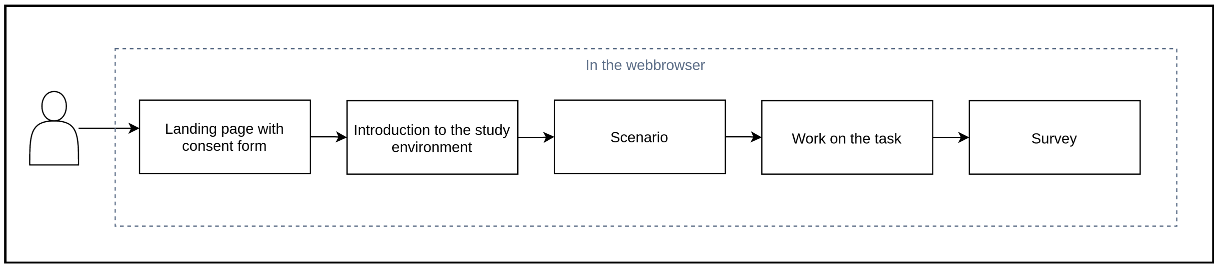

In the webbrowser
Landing page with consent form
Introduction to the study environment
Scenario
Work on the task
Survey

Figure 5: Comparison study (Section 4.3) setup for the condition that uses OLab.

Table 5: Qualitative coding of study results for the guided DevOps study (cf. Section 4.2).

Participants		Positive Points								Normal WE			Problems					Features	
	Recruited at	OK	Works despite adblockers/addons	Not related to infrastructure	Low Latency	Visually Appealing	Very good	Internet Access, Software can be installed	translates us layout	Not related to infrastructure	Social Interaction	Similar to Infrastructure	Not related to infrastructure	Change keyboard layout	Not the usual approach/environment	No chroot available	some lags (LTE)	Not related to infrastructure	Show correct Solution
P1	Admin-Forum				●	●	●				●		●					●	
P2	University			●						●				●	●	●		●	
P3	University						●			●					●	●			●
P4	Reddit	●								●			●					●	
P5	Reddit		●		●		●	●				●			●			●	
P6	Reddit	●					●				●				●			●	
P7	Reddit						●		●		●				●		●	●	
P8	Reddit	●			●		●			●					●			●	
P9	Reddit	●			●		●				●				●			●	

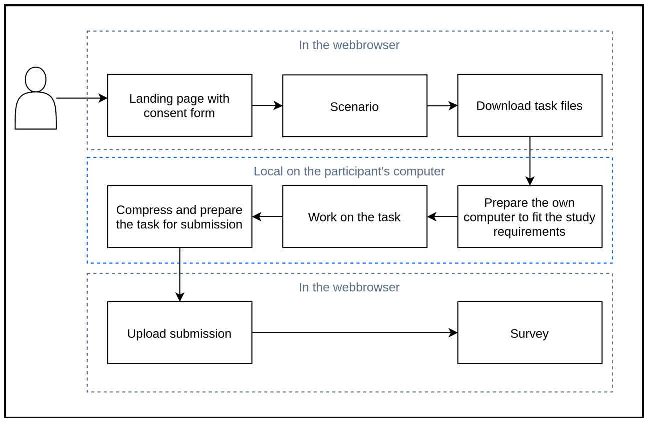
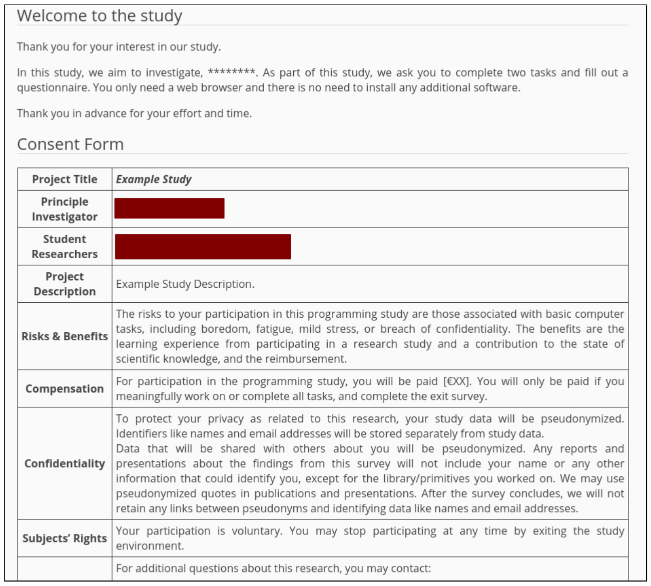

In the webbrowser
Landing page with consent form
Scenario
Download task files
Local on the participant's computer
Compress and prepare the task for submission
Work on the task
Prepare the own computer to fit the study requirements
In the webbrowser
Upload submission
Survey

Figure 6: Comparison study (Section 4.3) setup for the download condition.



Welcome to the study

Thank you for your interest in our study.

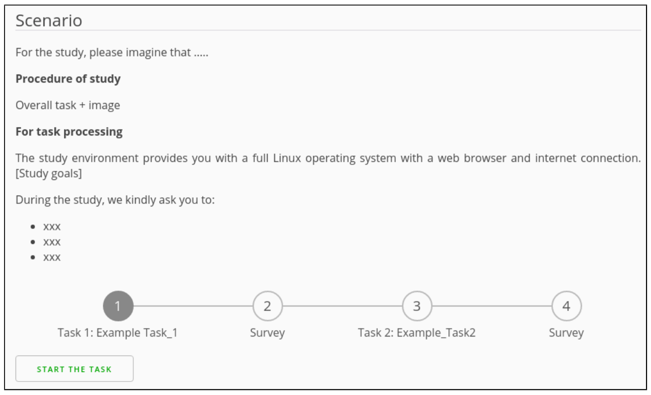
In this study, we aim to investigate, ********. As part of this study, we ask you to complete two tasks and fill out a questionnaire. You only need a web browser and there is no need to install any additional software.

Thank you in advance for your effort and time.

Consent Form

Project Title	***Example Study***
Principle Investigator	[redacted]
Student Researchers	[redacted]
Project Description	Example Study Description.
Risks & Benefits	The risks to your participation in this programming study are those associated with basic computer tasks, including boredom, fatigue, mild stress, or breach of confidentiality. The benefits are the learning experience from participating in a research study and a contribution to the state of scientific knowledge, and the reimbursement.
Compensation	For participation in the programming study, you will be paid [€XX]. You will only be paid if you meaningfully work on or complete all tasks, and complete the exit survey.
Confidentiality	To protect your privacy as related to this research, your study data will be pseudonymized. Identifiers like names and email addresses will be stored separately from study data. Data that will be shared with others about you will be pseudonymized. Any reports and presentations about the findings from this survey will not include your name or any other information that could identify you, except for the library/primitives you worked on. We may use pseudonymized quotes in publications and presentations. After the survey concludes, we will not retain any links between pseudonyms and identifying data like names and email addresses.
Subjects' Rights	Your participation is voluntary. You may stop participating at any time by exiting the study environment.
	For additional questions about this research, you may contact:

(a) Landing page with required consent form and further study information.



Scenario

For the study, please imagine that .....

Procedure of study

Overall task + image

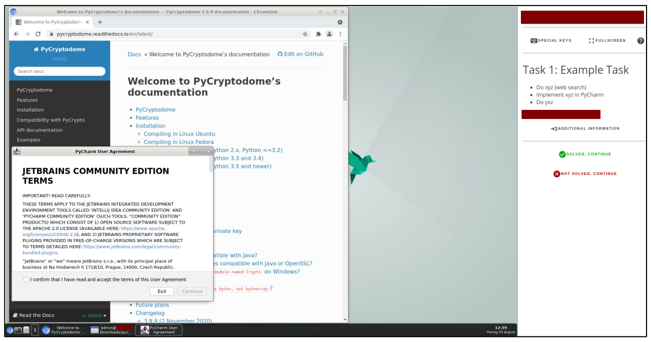
For task processing

The study environment provides you with a full Linux operating system with a web browser and internet connection. [Study goals]

During the study, we kindly ask you to:

- xxx
- xxx
- xxx

(b) In-between task progression status page, including survey steps.


[redacted]

Welcome to PyCryptodome's documentation — PyCryptodome 3.9.9 documentation - Chromium
pycryptodome.readthedocs.io/en/latest/
PyCryptodome
Docs » Welcome to PyCryptodome's documentation
Edit on GitHub
PyCharm User Agreement
SPECIAL KEYS
FULLSCREEN
ADDITIONAL INFORMATION
SOLVED, CONTINUE
NOT SOLVED, CONTINUE
Exit
Continue

Task 1: Example Task

- Do xyz (web search)
- Implement xyz in PyCharm
- Do yxz

[redacted]

JETBRAINS COMMUNITY EDITION TERMS

IMPORTANT! READ CAREFULLY:

THESE TERMS APPLY TO THE JETBRAINS INTEGRATED DEVELOPMENT ENVIRONMENT TOOLS CALLED 'INTELLIJ IDEA COMMUNITY EDITION' AND 'PYCHARM COMMUNITY EDITION' (SUCH TOOLS, "COMMUNITY EDITION" PRODUCTS) WHICH CONSIST OF 1) OPEN SOURCE SOFTWARE SUBJECT TO THE APACHE 2.0 LICENSE (AVAILABLE HERE: https://www.apache.org/licenses/LICENSE-2.0), AND 2) JETBRAINS PROPRIETARY SOFTWARE PLUGINS PROVIDED IN FREE-OF-CHARGE VERSIONS WHICH ARE SUBJECT TO TERMS DETAILED HERE: https://www.jetbrains.com/legal/community-bundled-plugins.

"JetBrains" or "we" means JetBrains s.r.o., with its principal place of business at Na Hrebenech II 1718/10, Prague, 14000, Czech Republic.

I confirm that I have read and accept the terms of this User Agreement

(c) Virtual study environment running Chromium & PyCharm. The right sidebar includes task descriptions and control buttons.

Figure 7: Screenshots of the OLab prototype, during a generic programming study.

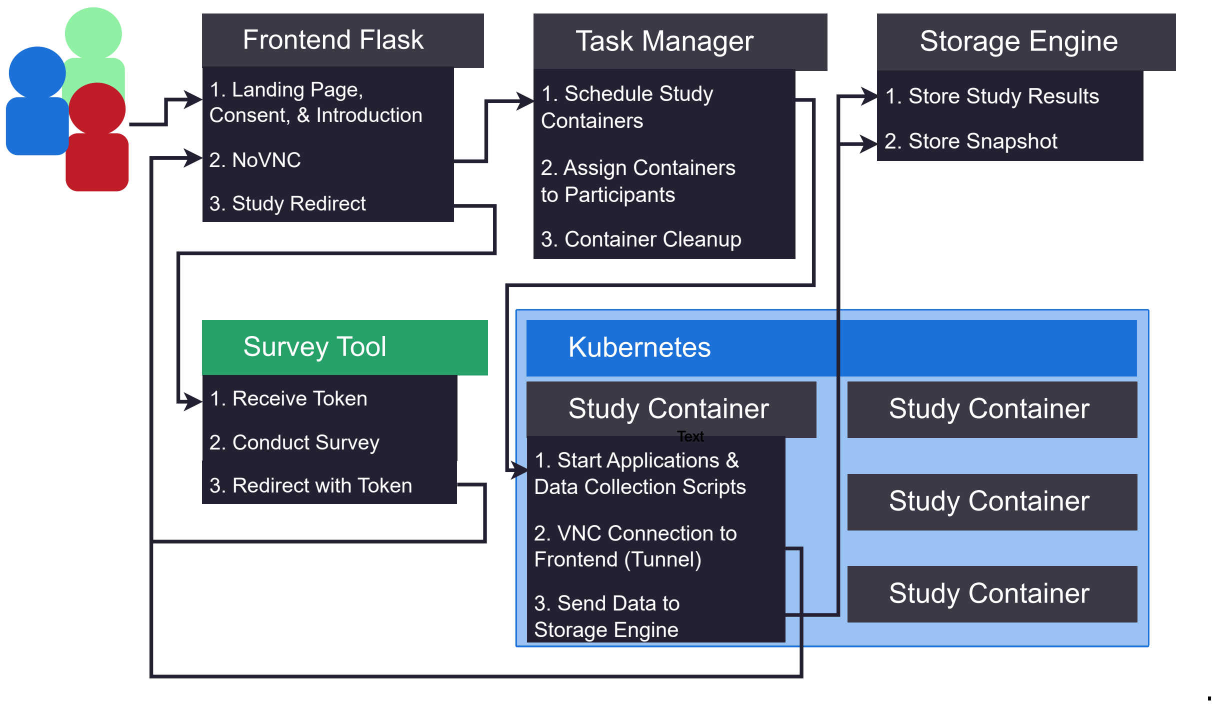


Figure 8: OLab architecture diagram.