"I'm trying to learn...and I'm shooting myself in the foot": Beginners' Struggles When Solving Binary Exploitation Exercises

James Mattei*, Christopher Pellegrini*, Matthew Soto*
Marina Sanusi Bohuk†, and Daniel Votipka*
*Tufts University; †MetaCTF

Abstract

Vulnerability discovery is an essential security skill that is often daunting for beginners. Although there are various supportive organizations and ample online resources to learn from, beginners often struggle, become frustrated, and quit. We conducted semi-structured observational interviews with 37 vulnerability discovery beginners attempting to exploit 51 vulnerable programs. We capture the questions beginners have when trying to identify and exploit vulnerabilities, how they search for answers, and the challenges they face applying their searches' results. We performed a rigorous qualitative coding of our dataset of 3950 events characterizing participants' actions to identify several behaviors and obstacles faced, along with quantitative measures to determine their most frequent issues.

We found beginners struggle to understand how to exploit vulnerabilities, craft their solutions, and even complete common technical tasks. They were often unable to find relevant information online to overcome these struggles, as they lacked the relevant vocabulary to craft effective keyword searches. When they did find relevant web pages, they struggled to properly transfer information from the web to their challenges due to misunderstandings and missing context. Based on our results, we offer suggestions for vulnerability discovery educators and resource creators to produce higher-quality materials to help facilitate beginner learning.

1 Introduction

As technology continues to proliferate into our daily lives and supports more critical infrastructure, we rely more on skilled practitioners to perform security reviews to ensure early vulnerability identification and mitigation [28, 40]. This need is compounded by the continual rise in cyberattacks [22, 63] and has been echoed by multiple governments' calls to grow the workforce [18, 64, 85].

There are a plethora of freely available resources online, including self-guided courses [24, 51], major conference talks about novel techniques [2, 3, 5], and thousands of hands-on exercises [86]. These resources are a testament to the security community's collective altruism and have led many to suggest beginners learning vulnerability discovery should utilize the abundance of online resources [86]. However, participants in previous studies reported difficulty finding the *right* resources and understanding how to apply them without prior knowledge [46]. Education literature suggests extraneous cognitive load spent understanding online resources can impede learning outcomes and burden beginners who may already be struggling with new material [84].

This paper seeks to understand what happens when beginners use these online resources. Because vulnerability discovery education is often in an unstructured setting [86, 90], learning relies on *finding* the right information, as well as being able to *understand* and *apply* it. Specifically, we focus on beginners' challenges when completing vulnerability discovery exercises (VDEs). The security community has often turned to VDEs to teach vulnerability identification skills because they provide hands-on, practical applications of security concepts and real-world exposure to various vulnerabilities [86, 90]. While these exercises can take many forms, the most common VDE is the Capture-the-Flag competition (CTF), which presents challenges as individual practice sets students can move through at their own pace. Challenges may cover web and binary ex-

ploitation, forensics, cryptography, and more. In each challenge, the goal is to discover a "flag" (a secret string of text) by exploiting some vulnerability, and participants can then submit that flag in exchange for points based on challenge difficulty.

These exercises are generally regarded as valuable educational tools—security experts report them as their most valuable learning tool [90]; high-profile technology companies highlight their value by hosting large external competitions [50] and use them internally for employee training [16, 49]; and much of the recent security education literature has focused on designing new types of VDEs [13, 20, 36, 37, 61, 67, 91].

Unfortunately, prior work has found that VDEs can be particularly challenging for beginners, often leading them to get discouraged and stop pursuing the field [10, 47, 70, 71, 73, 81, 90]. Therefore, any multi-challenge study investigating knowledge transfer and true learning beyond single task completion would suffer from survival bias. As a necessary first step toward improving learning for beginners, we must identify common roadblocks beginners face which introduce extraneous cognitive load, cause frustration, and prevent beginners from completing challenges. Improving task completion by removing these extraneous barriers can allow future work in security education and prior learning science literature suggests limiting extraneous load can have direct improvements on learning [84].

We seek to quantify what challenges beginners face during this process. Specifically, we strive to answer the following research questions:

RQ1: Where do beginners often struggle when attempting binary exploitation exercises?

RQ2: What types of information and web pages do beginners use when attempting binary exploitation exercises?

RQ3: Are these resources understandable and transferable to their context? What difficulties do beginners face applying the information within the resources?

To answer these questions, we performed a semi-structured observational study with 37 participants, asking them to identify and exploit a vulnerability in a simple binary program. Each participant was asked to find and exploit a vulnerability in one to two (of four) vulnerable programs, each with a different type of vulnerability. Our participants completed 51 challenges. We specifically considered binary exploitation problems, which require the review of a compiled binary program or related source code to identify and exploit a vulnerability. We chose to focus on a single challenge type, as we expected the knowledge and background needed for other categories to vary, complicating recruitment and our ability to produce detailed findings. We chose binary exploitation, as opposed to other potential categories (e.g., web exploitation), as this category has been shown to be the most challenging for beginners [27, 79]. Therefore, any improvements made by our work would be likely to have significant positive impact for beginners.

The challenge set was designed to expose students to the most common binary exploitation vulnerabilities [79]. Each challenge was designed as simply as possible, reflective of entry-level challenges beginners would encounter in binary exploitation exercises (BEEs). While participants solved their challenge, we recorded their approaches, their difficulties, and any resources (online or offline) they used.

We found that while there is no shortage of resources available to participants, they often need help crafting relevant queries and are prone to misinterpreting or misapplying the information they find. This typically occurred when understanding the vulnerability or crafting their exploits, and they even struggled with common technical tasks such as providing hex input. Participants often got stuck in a trial-and-error loop of trying various commands and keyword searches without a clear goal. We also noticed the utility of a given web page varied based on the participant's background. Less experienced beginners could not infer details not included on a page, preventing them from translating the information they found to their specific problem. Finally, some participants used ChatGPT to help solve their problems with varying levels of success.

Based on these results, we outline recommendations for BEE design and student support resources to make vulnerability discovery more approachable for all leaners regardless of prior experience.

2 Related works

Our interviews provide the first in-depth view of beginners' experiences solving binary exploitation challenges, their strategies, and the resources they use. Prior work has investigated security profession-

als' processes for vulnerability discovery, security education using VDEs, and the barriers to entry into the vulnerability discovery community.

Vulnerability discovery. A growing body of literature has sought to understand how security professionals perform vulnerability discovery tasks [17, 23, 25, 43, 59, 82, 88–90, 94]. Multiple studies have sought to identify security professionals' strategies when reverse engineering software to inform automation development [23, 25, 59, 89]. However, this work has focused primarily on defining security professionals' processes and interactions with code; investigations of beginners remain limited.

Perhaps the most similar to our work is a recent study by Ford et al., which classified mistakes 11 security professionals make during live hacking sessions while they solve BEEs [43]. Ford defined four mistake categories depending on their root cause: *Programming/Miscellaneous*, *Implementation*, *Strategy*, and *Tooling*. We drew inspiration from their research by analyzing our participants' mistakes, but we focused on a different population—beginners. Therefore, the mistakes and challenges identified for experts likely do not translate directly. Our work aims to understand how to get beginners *to* that level.

Security education using VDEs. The security and CS education communities have explored strategies for supporting beginners in security education. Much of this work has considered alternative VDE designs [41, 61, 67, 73]. Owens et al. found that gradually increasing challenge difficulty increased student engagement and reduced within-exercise dropout [67]. Researchers have also incorporated peer-based elements to encourage student collaboration [13, 61, 62]. Others have studied currently available exercise content (e.g., challenge topics and difficulty) to identify knowledge gaps [26] or pedagogy [86] to identify opportunities for alternative VDE formats. Some researchers have also proposed alternatives to CTF-style exercises [19, 34, 44, 69, 74, 75, 92, 93]. While this work has investigated several aspects of VDEs, our focus on the resources students use and their specific misunderstandings in practice offers a unique perspective on how the security community can produce resources and tools that better suit beginners' needs.

Barriers to entry into vulnerability discovery. Finally, recent research has investigated the barriers beginners face when entering the vulnerability discovery community [10, 42, 46, 53]. This work has motivated our current study, as early security education has shown to be a consistent barrier across all this prior work. Some of this research found that groups historically marginalized in security often do not have the community support needed to guide them through the large sea of available educational resources [46]. Women, in particular, feel othered and often face harassment, which makes it more challenging to seek help when stuck. Our work seeks to lower artificial barriers by making it easier for these beginners to find and use online resources, allowing them to focus on learning concepts germane to real-world vulnerability discovery. We hope this is a small initial support, especially for those who may continue to struggle to reach out for help due to the ostracization they have faced.

3 Methods

We now describe the methodology for our study. The interviews were conducted between April 2023 and April 2024.

3.1 Interview Structure

We performed video teleconference interviews where participants attempted a binary exploitation challenge (the "*interview challenge*"). Interviews lasted between 70 and 90 minutes and were conducted by a single interviewer.

Our interview challenges adopted a BEE-like structure where participants are given a vulnerable program that has read access to a secret string of text called a "flag." The challenge's goal is to reveal the flag by exploiting a vulnerability. We adopted this BEE format for our interviews to give our beginner population some structure and reflect how security professionals often report being introduced to the subject matter [90]. This format guides participant attention while allowing ample opportunity to troubleshoot and work through a difficult challenge. Participants were given the challenge source code, binary, and the IP address of the hosted challenge (to which they connected via `netcat`).

Available tools. Participants used a development environment preconfigured with command line tools such as GDB, Valgrind, and pwntools [48, 72, 83]. The challenges and development environment were hosted on t2.micro EC2 instances running Ubuntu 20.04. Participants could install any other tools they wanted; this minimized the chance participants would perform poorly due to an unfamiliar working environment. For participants unfamiliar

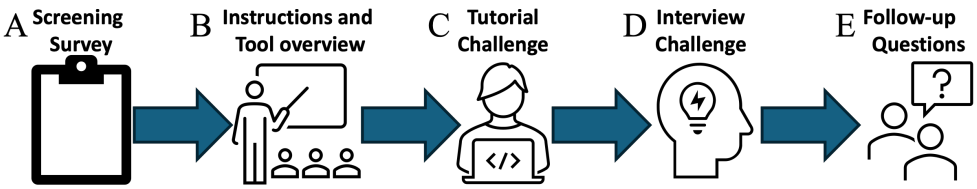

Figure 1: Overview of Just Google It observational study flow for participants

with the tools, we provided immediate help regarding their usage to ensure they could focus on vulnerability identification and exploitation. The complete list of tools is provided in the Supplemental material [6].

Provided materials and instructions (Figure 1.B). Before starting the interview challenge, participants were read an introductory script covering the structure of the interview and reviewing the functionality of the available tools [6]. Participants were also given a reference sheet with several commands, selected such that all the interview challenges could be solved only using the reference sheet [6]. The interviewer described each command on the sheet and allowed participants to ask clarifying questions.

Tutorial challenge (Figure 1.C). Before the interview challenge, participants were guided through a tutorial challenge that mirrored the interview challenge's structure. This problem did not involve exploiting a vulnerability but could be solved with basic code comprehension [1]. The tutorial familiarized participants with the task environment, the provided tools, and the challenge structure.

Interview challenge (Figure 1.D). After the tutorial challenge, participants were given one hour to solve one of four binary exploitation challenges. Section 3.2 includes descriptions of each challenge. No system defenses (e.g., ASLR, canaries, noexecute stack) were enabled in any challenge. Participants shared their screens, allowing us to record all the commands they used, visited webpages, and uses of the BEE reference sheet. Participants were asked to "think aloud" as they completed the challenge so we could capture their thought process. The interviews were recorded using Zoom's built-in recording functionality. Participants were instructed to refrain from viewing sensitive information such as email. If such information was accidentally recorded, the video was edited to redact the private information before further review and storage.

We limited the interview challenge to one hour after several discussions with VDE organizers. We believe this time pressure is comparable to what beginners face when completing a BEE for the first time while still allowing enough time for beginners to investigate the challenge sufficiently. One VDE organizer reviewed their logs and conservatively estimated beginners' work on BEE challenges for 53 minutes on average before finishing or giving up, not accounting for participants choosing to attempt other challenges first or taking breaks while solving the challenge. Another VDE organizer estimated beginners take 107 minutes on average to solve a similar challenge. However, this setting was less comparable as the problem is broken into seven distinct challenges requiring the participant to complete tasks not found in our challenges and participants are not given the program's source, making the task more difficult. Given these comparisons, we believe our time limit is a reasonable approximation as we provide regular hints described in Section 3.3, preventing participants from being stuck for long.

In some cases, participants were either reluctant to or struggled to craft web searches or use any tools. See Section 4.4.1 for a detailed discussion of this phenomenon. Because this prevented participants from progressing through the challenge, the interview protocol included cues to nudge participants to search the web for information or try commands on the terminal (See Section 4.2 for details).

Finally, we periodically asked probing questions to capture participants' processes. Some participants required this additional prompting as they focused on attempting to solve the challenge.

Probing and follow-up questions (Figure 1.E). To get a more detailed view of our participants' reasoning and information-gathering process, we asked participants to describe the quality of the webpages they visited, their solution strategy after a breakthrough, and what helped them improve their understanding. To avoid response fatigue, we limited questions to once every 10 minutes or after a notable breakthrough [15]. Once the interview challenge was completed, we asked participants to describe

their roadblocks, what resources helped them overcome them, and which were not helpful.

3.2 Types of Vulnerabilities

Participants were assigned one of four different interview challenges. We chose to focus on binary exploitation challenges as they were identified as the most difficult in Burns et al.'s review of 3600 CTF challenges [79]. We chose a difficult challenge type to investigate obstacles beginners face broadly, and findings from the most difficult challenges will more likely generalize to easier challenge types. We incorporated elements from six of the top seven most common vulnerability types identified—excluding the more sophisticated ROP challenges. A research team member with professional experience designing and hosting CTFs wrote the four challenges—each centered around a different vulnerability type. We chose a heap overflow, buffer overflow, integer overflow, and format string vulnerability and designed the challenges to be solvable within one hour. Table 1 lists the different challenge steps. The challenges are as follows:

- Heap overflow (Heap) - A basic buffer overflow on the heap that hinges on overwriting a function pointer to a new destination.
- Buffer overflow (Buff) - A slightly more complicated buffer overflow problem on the stack following the typical "Ret-2-win" format. "Ret-2-win" involves redirecting program execution to a specific function that, when run, will provide the flag value.
- Integer overflow (Int) - A more complicated version of Buff that requires participants to utilize an integer overflow to execute the vulnerable buffer overflow code.
- Format String (FmtStr) - A program requiring participants to perform a format string attack to leak the value of an un-referenced variable.

We designed each challenge to be difficult enough that participants would need to learn new information but not too easy that participants could accidentally solve the challenge. We piloted the entire interview process with four pilot participants (one for each challenge type) to assess challenge difficulty and validate our interview infrastructure. We increased the difficulty of the Int and Heap challenges, as our pilot participants were able to solve them too quickly. The change in difficulty was verified with new pilot participants whose data was included in the final dataset, as there were no further changes to the challenges afterward. We believe we achieved an appropriate difficulty balance as 41% of attempted challenges were solved, and 98% (all but one) made it to the final challenge step.

3.3 Hints

Our goal in this study was to identify beginners' struggles in vulnerability discovery, so we wanted to ensure every participant could make progress and attempt every facet of the challenge. To this end, participants were given a hint every 10 minutes, slowly revealing more detailed insights about the challenge. Prior work in this space has suggested quality hints that guide beginners through the steps of vulnerability discovery are a core aspect of good vulnerability discovery learning support [27, 31, 55, 79, 80]. We designed our hints to mirror the steps to solve each challenge. The complete list of hints can be found with the source code for the challenges [1]. Our hints progressed as follows:

10 minutes: Vulnerability name
20 minutes: Vulnerability location
30 minutes: General solution strategy
40 minutes: Specific first solution steps
50 minutes: Tailored in-depth help

The interviewer also delivered additional nudges to assist participants. These nudges never preempted information of scheduled hints. They were not delivered immediately to correct mistakes, as we wanted to capture the debugging behavior, but rather after the participant was stuck for an extended period. Nudges included help interpreting tool output, adjusting participant solution strategy, and overcoming roadblocks.

3.4 Recruitment

Because we wanted our results to generalize to a broad population of beginners, we used several recruitment methods to find participants:

University forums. We periodically posted the recruitment message in the supplemental material to our university's Computer Science Piazza page [6]. We contacted faculty at ten other universities to distribute the message, informed them of our target audience, and allowed them to decide the most appropriate venue to post our message.

Challenge Step	Description	Applicable Challenge
Identify vulnerability	Determine what vulnerability exists in the program	All Challenges
Locate vulnerability	Identify what line(s) or function call is vulnerable	All Challenges
Understand vulnerability	Understand how the vulnerability can be used to access the flag	All Challenges
Find the flag address	Certain challenges had a flag function that needed to be executed. Finding it's address is essential for the create payload	Heap, Buff, Int
Determine offset	Determine where in the payload the address of flag should be provided to leverage the vulnerability to print the flag	Heap, Buff, Int
Determine how to get past length check	When the participant correctly determined input sizes that would result in an integer overflow	Int
Find user input on stack	Identify the position on the stack at which user input starts	FmtStr
Provide flag at correct offset	Create a payload with the appropriate flag address at the correct offset to print the flag value	Heap, Buff, Int
Provide flag as input and print address with %s	Create a payload with the flag address, dereferencing the address with %s	FmtStr

Steps with dashed lines in between were grouped for post-hoc analysis. See explanation in section 3.6.

Table 1: Unique Challenge steps: Table of each challenge step, their definition, and applicable challenges.

CTF competition mail lists. We also worked with PicoCTF [66] and Pwncollege [76], two popular security education platforms, to send our recruitment message to students who had either previously competed in their VDEs or made an account with the platform to solve practice challenges. While recruiting on these lists resulted in some responses from more experienced individuals, we found many people who qualified according to our eligibility criteria—some people signed up for a VDE but did not solve any challenges at all. Participants with experience beyond our eligibility criteria who were not identified in the screening survey (see Section 3.5) were easily discovered during the interview. These experienced participants' responses were not included in our final dataset.

VDE Discords. Finally, we posted the recruitment message to various university student groups on VDE Discord channels. These channels offer students support and resources to help them practice their skills. As with the CTF mailing lists, these channels included beginners interested in vulnerability discovery with limited experience.

Participants were compensated with a $50 gift card for participating in the first interview. After the interview, participants were offered the opportunity to complete a second challenge for an additional $30. 14 participants completed two challenges. To avoid potential carryover effect, participants were only allowed to complete one of Int, Buff, and Heap as the solution steps for each were similar [79].

To capture various approaches to our challenges, we conducted interviews until we reached saturation of themes [29, pg. 113-115]. We observed unique themes across each vulnerability type, especially in the FmtStr challenge, which was the most different from the other three. Therefore, we considered saturation separately for each challenge type. After 12 interviews for all the challenge types, we reached saturation except FmtStr, which required 15 interviews. In total, our dataset includes interviews with 37 participants who solved 51 total challenges.

3.5 Participant screening (Figure 1.A)

We required participants to be proficient in C and have prior experience with assembly and machine architecture. Participants needed to know the basics of stack construction to understand the vulnerabilities, and we did not want participants wasting time learning C syntax. To confirm their eligibility, all participants completed the pre-screening survey in the supplemental materials [6]. We assessed participants' fluency in C using two questions adapted from Danilova et al.'s assessment of software development competency [33]. These questions showed one C function and asked participants to identify specific features. Survey respondents had to answer both correctly to be interview-eligible.

Similarly, we wanted to ensure our participants did not have too much experience with BEEs. To do this, we assessed participants' perceived vulnerability discovery skill level using the validated 9-item vulnerability discover subscale of Votipka et al.'s Secure Software Development Self-Efficacy Scale (SSD-SES), which asks participants to rate their abil-

ity to perform nine vulnerability discovery tasks on a 5-point Likert scale [87]. We asked participants to self-report their years of security experience and any security education. Survey respondents who had previously participated in more than three VDEs or solved more than three BEEs were considered too experienced and were not interviewed.

This cutoff was progressively determined as we conducted interviews with more experienced participants who could complete the challenge without referencing any resources, demonstrating clear prior knowledge of the correct solution strategy. This unique behavior was straightforward to identify and unique to participants who self-reported previously participating in several VDEs or completing multiple binary exploitation challenges. We removed their interview from our final dataset and tightened our exclusion criteria. The survey concluded with common demographic questions to assess the representativeness of our sample.

3.6 Data analysis

We used qualitative and quantitative methods for our analysis, described below. We first performed a rigorous qualitative coding of 3950 events across all interviews to label all aspects of the beginners' vulnerability identification and exploitation process, enumerating the range of their strategies, mistakes, and roadblocks. Then, we performed quantitative comparisons using this dataset to determine common struggle areas. These analyses work together to provide a rich view of beginners' experiences.

Qualitative analysis. The interviews were analyzed using iterative, open coding [78, pg. 101-122]. Two researchers jointly reviewed two videos and discussed with the research team to identify themes and generate the initial codebook. We used an event-based approach when defining the features of our codebook. We wanted to capture every action taken and roadblock encountered, so our codebook included tags to describe when participants read the source code, used various tools, visited a webpage, made a mistake, got frustrated, and received a hint, among other features. The supplemental material [6] provides the full list of codes and definitions.

Using this codebook, two researchers independently coded interviews in groups of two (switching to groups of three after the first three rounds and five for the final round to allow for higher incidents of uncommon codes), beginning with the initial codebook and allowing additional codes to emerge from the data. After each round, the researchers met to calculate Krippendorff's alpha (α), compare codes, resolve disagreements, update the codebook, and re-code interviews. Interviews were only jointly reviewed by the full research team after they were coded. We calculated α using the ReCal2 software package to measure inter-coder reliability [52]. We used α as it provides a conservative measure, accounting for chance agreements. This process was repeated seven times until α values exceeded 0.80 for each subjective variable. Final α values are reported in the supplemental material [6]. We did not calculate α for objective variables like what command participants used or the type of hints they received, as suggested by IRR best practices [60], as these can be drawn directly from the transcript. After reaching agreement, the remaining 27 interviews were divided between the two researchers and coded by a single researcher. The reported α values are for the seventh round when agreement was reached. The 27 singly coded interviews analyzed afterward did not impact reported α values.

After completing open coding, we performed axial coding to determine groups of codes for each variable. Axial coding identifies connections between codes to extract higher-level themes [78, pg. 123-142]. We specifically wanted to identify common strategies and pitfalls that impede progress.

Quantitative comparisons. To understand which steps were most challenging or required the most online searching, we conducted pairwise Chi-squared tests appropriate for categorical data [45] across participants and problem steps of the distribution of hints, mistakes, frustrations, and webpage reviews. To account for multiple testing, we applied a Bonferroni correction [39].

To compare across all challenge types, we needed to group the unique steps of Int and FmtStr to match the same six-step structure of Buff and Heap as shown in Table 1. Int had one additional step: *Determine how to get past length check*. Since leveraging the Integer Overflow and calculating the offset for the Buffer Overflow influence the construction of the final payload, we decided to merge these with the *Determine Offset* step.

FmtStr shares the same first three steps with the other challenge types but has unique final steps: *Find user input on stack* and *Provide flag as input and print address with %s*. The latter step is functionally the same as *Provide flag at correct offset*. There-

fore, these steps are grouped to *Provide payload*. *Find user input on stack* is a unique challenge step in our dataset. Still, in terms of completing FmtStr, this is the penultimate step, serving a similar function as *Determine Offset* for the other problems. Participants needed to find their offset on the stack and then incorporate that information into their payload.

These step groupings for Int and FmtStr were done for post-hoc evaluation to guide our analysis. We do not attempt to compare time-to-solve between challenges directly. These results were used to develop further questions for our dataset.

3.7 Limitations

While we attempted to replicate the experience of a real-world vulnerability discovery, there are inherent limitations with a lab setting. While we tried to emulate the structure of a standard BEE experience, we included more detailed hints than would be given in a real-world setting. Thus, our success rate may be higher than expected in practice. This was necessary to ensure we captured participants' struggles throughout the challenge but limited us from making claims about their efficiency.

We also place relatively short time constraints on participants. While competition-based VDEs often have time limits, students typically have more than one hour to solve a BEE challenge. We discuss this further in Section 4.4.3.

Participants may have also changed their behaviors because they knew they were being watched, introducing a Hawthorne effect [54]. However, the typical response to this effect is to increase productivity or focus, so our results likely provide an upper bound on student effort.

While we attempted to recruit broadly to avoid bias, we cannot claim our sample represents all beginners' experiences and may be specific to the Western-centric institutions through which we recruited. Additionally, while we have a sufficient sample to make claims about the frequency of various events, it is not large enough to make generalizable claims in all cases. For example, we included multiple types of vulnerabilities in our challenge programs but did not directly compare them, as a small number of participants were assigned to each. Throughout our results, we present statistical results for cases with sufficient data to make comparisons. Otherwise, results should be viewed as qualitative themes, which should be tested in future work.

Lastly, our study investigates beginner difficulties when solving BEEs, which may differ from their struggles when solving other VDE challenges, such as web exploitation or cryptography. Despite this, we believe that many of our results about beginners searching and using online resources can be applied broadly to similar scenarios. Additionally, BEEs are sufficiently complex to capture various roadblocks beginners face that may not be present in other easier challenge types.

4 Results

This section presents our findings. When referring to a particular participant, we use the notation "PX," where X is an anonymized number unique to the participant. When referring to many participants, we use "N="; and when referring to a number of events, we use "E=".

4.1 Participants

Table 2 gives experience metrics for all 37 participants. Only 15 (39%) had previously participated in a VDE. The majority (68%) had never solved a binary exploitation challenge. There was no clear difference in task performance between participants with or without prior experience.

Our participants came from 11 different universities across four countries. Our sample is also demographically similar to prior surveys of CS undergraduate students and participants in VDE exercises [12, 70]. Our population is mostly male (70%), Asian (42%) or White (27%), and young (63%$<$ 25).

4.2 Where Beginners Struggled (RQ1)

We first sought to understand where beginners most often struggled when attempting to find and exploit vulnerabilities. We focused on four events from our codebook which indicate participant struggles: *Mistakes, Frustration, Hints, and Web Pages*. These events encompass both when our participants got stuck and when they needed the most additional information support help them make progress.

Figure 2 shows a line graph indicating the number of participants with the code types' given count. Each line represents a challenge step. This section discusses trends for each variable across steps and issues participants struggled with at each step. We present statistically significant results where relevant. The χ^2 and corrected p-values for each com-

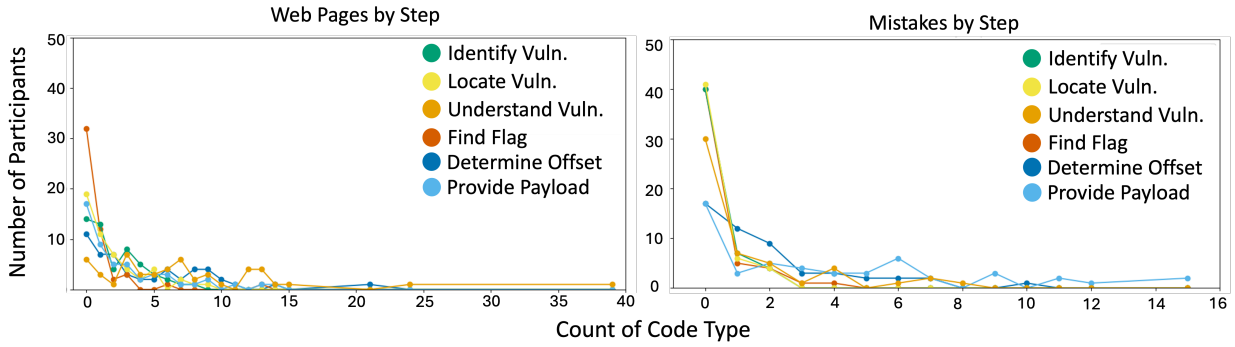

Figure 2: Line graph showing the number of participants with different counts of Mistakes and Web Pages grouped by problem step. Each line represents one of the steps for the interview challenge. The Y-axis shows the number of interviews that had the number of instances of specific codes on the X-axis.

parison are provided in the supplemental material [6].

Participants struggled to determine how to exploit vulnerabilities. Participants often struggled the most during the *Understand vulnerability* step as they figured out how to exploit it. The clearest indicator was that participants visited the web the most during this step compared to every other step (*Identify Vulnerability* $\chi^2 = 127.22$, $p < 0.001$; *Locate Vulnerability* $\chi^2 = 144.17$, $p < 0.001$; *Find Flag* $\chi^2 = 227.70$, $p < 0.001$; *Determine Offset* $\chi^2 = 23.52$, $p < 0.001$; and *Provide Payload* $\chi^2 = 12.96$, $p < 0.001$). This step naturally requires participants to search for information online, but several participants continuously updated their understanding of the vulnerability throughout the interview. P20 worked on this step for over 40 minutes, visiting seven different kinds of web pages and searching for eight different types of information—more than any other participant.

Many participants found relevant web pages about the vulnerability but struggled to implement the information. Of the 25 participants who exhibited frustration, 21 experienced it when trying to transfer exploit knowledge from web pages. Participants were more likely to be frustrated during the *Understand vulnerability* step compared to the *Identify vulnerability* ($\chi^2 = 16.94$; $p = 0.001$), *Locate vulnerability* ($\chi^2 = 15.11$; $p = 0.002$), and *Find flag* ($\chi^2 = 13.44$; $p = 0.004$) steps. P2 experienced this in practice, spending nearly ten minutes trying to get a command referred to on a web page (`cyclic`) to run. After giving up on that approach, they continued searching the web to try and find another web page.

Also, many participants (N=25) needed several hints to complete this step, suggesting participants had difficulty applying information from the web to the challenge. For FmtStr in particular, participants struggled to find their input on the stack and print out the flag variable. More participants completed this step via hint (N=11/15) than any other step for any challenge (Buff 6/12; Int 6/12; Heap 7/12).

Participants often misunderstood how to perform technical procedures. The *Provide payload* and *Determine offset* steps were particularly challenging for participants. This difficulty is reflected in the number of mistakes, which were statistically significantly more common in the *Provide payload* step compared to *Identify vulnerability* ($\chi^2 = 94.41$, $p < 0.001$), *Locate vulnerability* ($\chi^2 = 99.03$, $p < 0.001$), *Understand vulnerability* ($\chi^2 = 36.25$, $p < 0.001$), *Determine offset* ($\chi^2 = 15.22$, $p = 0.001$), and *Find flag* ($\chi^2 = 96.69$, $p < 0.001$). Similarly, frustrations were most common during the *Determine offset* step. This difference was statistically significantly different when compared to *Identify vulnerability* ($\chi^2 = 14.23$, $p = 0.002$), *Locate vulnerability* ($\chi^2 = 12.50$, $p < 0.006$), and *Find flag* ($\chi^2 = 10.94$, $p < 0.014$).

Both of these steps are relatively formulaic; there is a clear set of steps to follow once the vulnerability is identified, and they are mostly consistent across vulnerability types. However, determining these steps is difficult for those unfamiliar with binary exploitation. For example, most participants had never provided a hex address as input to a program and realized they could not just paste it as input (N=33). Many participants attempted to convert the flag address to `ascii` before providing it as input (N=15). Similarly, several participants made mistakes interpreting the hex output from GDB when

ID	VDE Skill[1]	VDE Exp[2]	BinEx Chal[3]	Vuln Type
P1	0	No	No	FmtStr/Heap
P2	19	No	No	Buff/FmtStr
P3	14	No	One	Heap
P4	0	One	No	FmtStr/Int
P5	10	No	No	Heap/FmtStr
P6	0	One	Mult. (2-3)	Int / FmtStr
P7	34	Mult. (2-3)	No	Buff/FmtStr
P8	33	One	One	Buff/FmtStr
P9	16	One	Mult. (2-3)	Int
P10	0	Mult. (2-3)	Mult. (2-3)	Buff
P11	24	No	No	Buff/FmtStr
P12	40	Mult. (2-3)	Mult. (2-3)	Int/FmtStr
P13	21	No	One	Buff
P14	23	Mult. (2-3)	One	Int/FmtStr
P15	17	Mult. (2-3)	Mult. (2-3)	Heap
P16	29	No	No	Heap
P17	18	No	No	Buff
P18	28	No	No	Int/FmtStr
P19	25	No	No	Heap/FmtStr
P20	30	No	No	Buff/FmtStr
P21	27	No	No	Heap/FmtStr
P22	29	No	No	FmtStr
P23	28	No	No	Heap
P24	30	No	No	Buff
P25	25	One	Mult. (2-3)	Int
P26	29	No	No	Heap
P27	19	No	No	Buff
P28	30	No	No	Int
P29	25	One	No	Heap
P30	31	One	No	Buff
P31	25	Mult. (2-3)	Mult. (2-3)	Int
P32	26	No	No	Heap
P33	27	No	No	Buff
P34	28	One	No	Int
P35	31	No	No	Heap
P36	26	Mult. (2-3)	One	Heap
P37	22	No	No	Int

[1]The cumulative score of 10 self-efficacy questions on a Likert scale. 0 - did not answer, 10 - least confident, 50 - most confident
[2]The number of formal VDE competitions participated in
[3]The number of binary exploitation VDEs solved.

Table 2: Participants' self-reported demographic information and assigned vulnerability types.

calculating the offset (N=16). Despite the reference sheet describing commands to complete these steps, many participants either discarded the commands as irrelevant or did not know how to use them.

Additionally, participants had trouble providing their payload to the remote location over `netcat`, despite confirming their solution was functional with the local copy of the challenge (N=8). After struggling for an hour to solve the challenge, getting robbed of the satisfaction of seeing the actual flag value left those participants even more frustrated. P34 said, "Wow, I really suck at these... This was actually, legitimately terrible."

Finally, participants neglected to convert addresses to little-endian format (N=24). While several participants recalled learning endianness, some shared the same sentiment as P10 when asked if they had previously heard about little versus big endian: "Oh, right. I remember learning about that, I just didn't really think it was important."

Many participants required a hint to find the vulnerability. The incidence of hints for the *Identify vulnerability* step was one of the lowest by counts (E=44) and had statistically significantly fewer than *Understand vulnerability* ($\chi^2 = 35.52$, $p < 0.001$), *Determine offset* ($\chi^2 = 41.40$, $p < 0.001$), and *Provide payload* ($\chi^2 = 21.23$, $p < 0.001$). Since participants only needed one hint to progress past this step, many were stuck until they received a hint (N=30). For example, P20 explicitly declared after receiving the vulnerability name hint, "OK, I guess I have something to Google now, so thank you!"

4.3 Most Common Resources (RQ2)

Our participants visited web pages 711 times. Table 3 includes the top four most common web page types and the number of times they were referenced for each step. We identified 12 different web page types. We only discuss the top page types for brevity (∼75% of viewed web pages), but we discuss the others in our supplemental material [6].

Forum pages were universally used. Forum pages were the most commonly visited (N=33, E=175)—this included StackOverflow, StackExchange, Reddit, and other similar websites. Participants most commonly used forum pages to understand the vulnerability (E=63), determine offsets (E=55), and provide the exploit payload (E=39). These pages often included short walkthroughs of VDE challenges or examples of using various tools.

Many participants also reviewed pages that included command-line tool specifications. The second-most common web pages used were sites providing resources similar to the Unix *man pages* (N=21, E=130). This included websites like GeeksforGeeks, TutorialsPoint, and DigitalOcean. Participants used these sites early in the vulnerability discovery process to try to identify (E=40), locate (E=21), and understand (E=43) the vulnerability.

Several participants used large language models to support information search and tool output interpretation. Large Language Models (LLMs) are a single location to search for new information instead of scouring multiple pages. Twelve partic-

Code Type	Sub Code Type	Identify Vuln	Locate Vuln	Understand Vuln	Find Flag	Determine Offset	Provide Payload	Totals
Mistakes	Application of info	4	1	13	2	34	63	117
	Address and memory	1	2	17	10	22	45	97
	Syntax	3	4	21	3	26	38	95
	Endian	0	0	6	0	7	36	49
	Total	8	7	57	15	89	182	358
Frustration	Lack of familiarity	4	5	20	1	19	9	58
	Tool output	0	3	11	2	6	3	25
	Web page	0	0	3	1	8	2	14
	Arbitrary roadblock	3	1	0	1	1	2	8
	Total	7	9	34	5	34	16	105
Web Pages	Forum page	17	10	63	11	55	39	175
	Man page adjacent	40	21	43	3	15	8	130
	LLM	7	12	39	11	38	12	110
	Web results only	43	15	18	14	7	7	163
	Total	107	58	163	39	115	66	578

Table 3: Top four Web pages, Mistakes, and Frustration broken down by step and challenge. Note that there are instances of double counting, as a participant can make one mistake or visit the same web page while working on several steps simultaneously. Thus the values in the table may not sum to the reported Totals.

ipants (∼32%) used LLMs (E=119) as interactive tools to search for information. P30 described the tradeoff between traditional web pages and LLMs, saying, "That's the bad thing about these static websites. If it has the info, it has it; if it doesn't, then you can't ask follow-up questions or anything."

LLMs can also act as a utility to help interpret tool output. When trying to understand the output of GDB, several participants did not know how to interpret it correctly. P25 thought to utilize an LLM saying, "So, obviously, I have no clue what's going on here…but maybe ChatGPT can help."

Participants did not always click into the search results. We observed several participants conduct a search, scan the returned results, and move on (N=32, E=163). Some participants learned the information they needed from just the returned result summaries. However, we found that of the 163 times this happened, the participant answered their question in only 59 cases (36%). As discussed in Section 4.4.1, viewing just the search results was more often an indicator the participant was struggling. Participants often exhibited this behavior when understanding the vulnerability (E=43).

4.4 Struggles Using Resources (RQ3)

Finally, we observed that participants often struggled to search for, understand, and apply relevant information to exploit their vulnerability.

4.4.1 Search paralysis

We observed several cases in which participants could not progress because they lacked any clear idea of what to search for. We define this behavior as *search paralysis*. It often presented as the participant pausing for long periods and expressing they were stuck or attempting several searches in rapid succession without a clear goal.

Many participants paused when they lacked the vocabulary to craft search queries. P33 acknowledged, "In class, they never taught me how to [exploit the `gets()` function]—just 'don't [use it]'." This was particularly common before we provided participants the vulnerability name hint. P8 described their struggles to find relevant keywords to guide their web searches, saying, "[I'm] trying to learn what I need to learn, and I'm shooting myself in the foot from that." Several participants experiencing search paralysis spent more than five minutes trying to form a search query.

Even after participants had some information to work from, their lack of experience limited their ability to develop useful search terms or understand the search results they received. This often led participants to progress through several searches quickly. For example, in six minutes, P8 made seven different web searches. They explained this behavior saying, "There isn't much logic to what I'm doing. I'm just trying to do something to get started somewhere in some kind of direction because I'm feeling

very lost as a newcomer." In many cases, these participants did not even click on any of the links for their search results, as described in Section 4.3.

4.4.2 Divergence in page utility

We observed several cases where multiple participants viewed the same webpage but diverged in whether they could use the presented information.

Prior experience impacted participants' ability to apply relevant information. Often, participants could not process webpage information due to a lack of background knowledge (E=218). In one instance, two participants working on two different challenges navigated to the same forum page with step-by-step instructions on using `cyclic` to help find the offset. P21 found the page early on and returned to the page nine times to reference the material and complete this step. Conversely, P2 recognized the page as relevant but had difficulty using it because it did not describe how to use the tool. They proceeded to perform several subsequent searches related to `cyclic`'s use, which did not yield the information they needed. Instead, P2 eventually restarted their search process and found a walkthrough using `GDB`, which they successfully replicated the steps to make progress.

Participants frequently skimmed web pages, missing key information. Participant webpage navigation ranged from quickly skimming page headings (E=472) to thoroughly reviewing each element and clicking through relevant links (E=124). Their deployment of these strategies often varied depending on the questions they were trying to answer. This is expected based on prior studies of general user web information retrieval practices [35,56]. However, these behaviors dramatically affected whether they could find relevant information. The clearest example of this was OWASP's webpage about format string attack [7]. This page was visited by almost all participants who were assigned the FmtStr challenge (N=11 of 15). Linked within that page is a longer academic writeup detailing format string exploits that was visited by only seven participants, and only three read it in enough detail to understand the vulnerability. None of the other interview challenge types had a similar concentration of participants visiting the same webpage.

Int was the most challenging problem for our participants in terms of hints and frustration. The Int challenge was designed to be the most difficult challenge in terms of the number of steps needed to solve it out of the interview challenges. This was reflected in Int having the highest count of hints and incidences of frustration compared to all other challenge types. Conversely, the Heap challenge was the simplest version of a data overflow challenge and had the fewest number of visited webpages.

Prior experience and navigation differences also impacted LLM use. Often, participants could not tell when an LLM was hallucinating. For example, P25 tried to use an LLM to convert a hexadecimal value (0x61616168) to ascii. It incorrectly asserted that the ascii value was "aahh", instead of "aaah." This resulted in P25 using the wrong offset for the problem. Conversely, familiarity with LLM hallucinations led several participants to discard valid answers for fear it could be a hallucination (N=7). P34 asked an LLM "how can attack jump to a function using a buffer overflow attack with examples?" The response contained a relevant example and described steps to progress on their challenge. However, P34 suspected the LLM's example was wrong. After briefly skimming the response, they continued their search on the web instead.

Some participants provided the entire source code (N=4), relevant addresses (N=6), and clear instructions to create a payload to solve the challenge. One participant managed to get an LLM to solve Buff. However, this approach appeared to have blunted the participant's learning process. When asked to explain why their solution worked, they could not accurately describe why the offset was significant or why the flag function was executed.

4.4.3 Other struggles engaging with websites

We also observed interesting emergent behaviors among our students that created unique struggles.

Participants were hesitant to use walkthroughs. Despite repeated encouragements to utilize any information available online, several participants had reservations about using walkthroughs for other similar VDE challenges (N=10). Challenge walkthroughs are a commonly suggested online resource for VDE beginners to reference when learning about new topics [38,77]. However, participants like P14 and P8 thought reading that information would "ruin the fun" and "wasn't in the spirit of the challenge," respectively. At least one participant did not realize walkthroughs of other VDE challenges existed. Other participants expressed similar hesitance

to rely on LLMs (N=3). For example, P28 stated, "If I just copy-paste all this code into ChatGPT, it would probably give me the answer, but I don't want to do that because I want to learn!" The observer effect may also influence this number [14].

Some participants felt the time pressure impacted learning. Another reason participants cited for their struggles was the time pressure (N=7). P25 exemplified a common sentiment, saying, "I wish I used Google more, but the time pressure really made it feel like I didn't have time to spend fully understanding and learning something and then go apply it." This time pressure dissuaded other participants from reading any web pages that were too long, even if they recognized them as potentially useful. P20 described a web page other participants used to help solve the challenge as "useful for research and understanding, but in a time-boxed scenario maybe not as much."

5 Discussion and Recommendations

Students struggled most to understand vulnerabilities, determine offsets, and provide payloads. *Search Paralysis* was a common occurrence both before participants knew the name of the vulnerability and when trying to refine their understanding of it. Many of these struggles were caused by technical difficulties not specific to a particular challenge. Participants needed help crafting relevant search queries and translating information on webpages to the challenge context. Participants could find relevant webpages but struggled to understand ones laden with unfamiliar terms. Participants who successfully utilized these pages had to exhibit patience and perseverance to implement the information. Finally, we noticed new emergent stymied behaviors among students, such as avoiding walkthroughs and managing hallucinating LLMs. Many of these challenges introduce cognitive load extraneous to learning how to exploit the vulnerabilities, limiting learning [84]. Based on these results, we provide the following recommendations for VDE organizers and anyone who creates online resources to support beginners learning binary exploitation.

5.1 Suggestions for VDE Organizers

Our results provide valuable insights into the parts of challenges where students most often struggle, indicators that students are stuck, and other common student misconceptions. VDE organizers can leverage these results in their exercise design to develop a more supportive and responsive environment.

Focus hints on resolving basic technical challenges. Many participants struggled to provide the flag address to the program and overlooked endianness as a factor that could break their solution. These mistakes were particularly time-consuming for our participants but ultimately separate from learning about the vulnerability. A well-timed reminder to consider endianness or a nudge to reference a command sheet would help students overcome these obstacles and make continual progress. Several VDE reference sheets already exist [8, 9], but more timely hints might prevent students from overlooking the resources already available.

Monitor beginners' commands and web search behavior to identify stymied students. Being able to get oneself unstuck is a critical aspect of learning. Still, the learning sciences literature suggests students need feedback before they become *overly* discouraged and potentially stop trying altogether [11, 21, 30]. Existing VDEs struggle to provide this type of timely active feedback [86], but our work offers a path forward. We observed two common behavioral patterns indicating participants were stuck. The first occurred when participants quickly attempted several commands or searches to see if any output inspired a new idea. The other occurred when participants paused for long periods and expressed no new ideas for progression. VDE organizers could track these behaviors and provide automatic hints to help students progress [57]. The former is relatively straightforward to monitor if the VDE offers an in-browser experience, as some beginner VDEs already provide [66, 76]. The latter is more complicated, as long pauses could be from the participant taking a break. To resolve this ambiguity, the VDE could ask the user if they need help and offer a hint. Future research must investigate the proper balance and specific connection between observed behaviors and frustration. Furthermore, organizers could benefit from observing participant search frequency, which can strongly indicate that participants are stuck and aimlessly searching the web.

Destigmatize walkthroughs for beginners. In Section 4.4.3, we observed several participants chose not to view walkthroughs because they believed they would ruin their learning experience. However, walkthroughs can be an excellent resource, detailing why and how a vulnerability can be ex-

ploited through step-by-step instructions. Reading them can help students establish the baseline knowledge to make sense of other webpages and solve different challenges on their own [68]. Some VDE index websites allow for uploading walkthroughs once the competition is over [4], but VDE organizers should prioritize normalizing their use [32].

When initially establishing the experience cutoff for our study, we interviewed five individuals who were so experienced they solved the interview challenge without using the web. Although we excluded their data, all five said they would go directly to a VDE walkthrough if they had to search the web for information. While this is a prohibitively small sample, it suggests walkthroughs may not hinder vulnerability discovery education.

Help students understand the strengths of LLMs. Student LLM use will likely continue to rise as LLM quality improves and their use becomes more ubiquitous. LLMs are a valuable tool, but they can cause their own set of problems. Furthermore, some students may spend more time prompt engineering an LLM to solve the challenge instead of learning the material. Organizers should provide guidance on LLM use, describe their strengths, and raise awareness of potential hallucinations. Future research should investigate how VDE organizers can implement local custom instances of LLMs designed to help guide beginners through easier challenges.

Provide common utilities internally. In the *Provide payload* step, participants frequently turned to online tools to help convert hex strings to ascii. We saw numerous mistakes both in using these tools and interpreting their results. If VDE organizers internally provide these utilities, they will be easier for students to find and for organizers to provide tailored support for students who make mistakes.

5.2 Suggestions for Resource Creators

Our results also provide insights valuable for resource creators producing content for beginners. While not all resources are developed for beginners, we provide recommendations for people targeting novices. Specifically, we identified that participants struggled to interpret webpages due to a lack of knowledge or missing context in existing resources.

Provide additional means for beginners to evaluate webpage relevance. We noticed many of our participants struggled to determine if a webpage had useful, quality information. However, websites like Stack Overflow provided participants with additional information about the relevance of a particular response with a vote count and information about the response author with their reputation score [58]. We recommend that other platforms that host resources for beginners adopt these crowd-based metrics or embed additional information to give beginners more information when they search for relevant webpages. For example, CTFTime is a platform that hosts CTF competitions and hundreds of writeups provided by competitors to previous challenges. Their writeups have tags such as "pwn" or "forensics" as well as rating scores, which can help users find relevant writeups [4]. Although these metrics exist on CTFTime, many writeup authors do not apply tags, and many writeups do not have a rating score. Further research is needed to determine why authors don't add this information and how to make it easier for them to do.

Do not assume beginners know how to use various tools. Often, participants did not realize a webpage was relevant or could not use it because it lacked details about the mentioned commands. Despite `cyclic` being a standard binary exploitation tool, our participants visited very few webpages that explained how to use the tool. Beginners need more specific details about *using* different tools and how they relate to exploiting the vulnerability.

Discuss the implementation details of the exploit design. Resources that included examples or described specific steps were helpful for participants. However, if the details of these examples were not given, beginners could not intuit them. Also, if the resource context differed from the challenge's (e.g., with endianness or offset value), students struggled to make the necessary changes on their own. However, resource creators should be wary about filling the webpage with excessive detail, as participants tend to disregard webpages with a massive wall of text. A balance in the amount of detail to ground each step without overloading beginners is essential for a high-quality resource, and how to find the sweet spot of this tradeoff needs further research. Websites like PicoCTF combat this issue by providing numerous educational resources in varying lengths and mediums [65]. Their supplied materials include in-depth tutorial videos, brief writeups and command sheets, and free online courses more curious students can participate in on their own time.

Acknowledgements

We thank the anonymous reviewers who provided helpful comments on drafts of this paper. We also thank the organizers of PicoCTF, Pwn.College, Kelsey Fulton, Hanan Hibshi, Michelle Mazurek, Kirill Levchenko, and Marshini Chetty for their help with recruitment and Yan Shoshitaishvili for providing valuable insights into the study design and beginner CTF timings. This project was supported by NSF grant CNS-2247959 and a gift from Google.

Ethics Considerations

Our study was reviewed and approved by the Tufts University Institutional Review Board. Participants were given and agreed to the full informed consent document when they initially filled out the screening survey. Once the participants joined the Zoom meeting, they were again explained the contents of the informed consent document, the data that would be collected, and their rights as participants before the recording started. Participants did not need to have their webcam on during the interview and were instructed that they could withdraw from the study at any point. Similarly, participants could request their data be withdrawn from our data set at any moment, even retroactively. Repeat participants were paid less for the second study as they did not need to do the tutorial challenge or hear the study fully described again; on average, this saved 30 minutes. Tufts University's Institutional Review Board approved our study design.

Additionally, we do not envision significant potential harm from publishing this work. In theory, after learning common beginner misunderstandings, attackers could seed intentionally misleading resources online to frustrate beginners. However, because beginners' information searches are often mediated by suggestions from the community in forums and Q&A sites, this would require co-opting a segment of the population of trained vulnerability discovery experts. Further, because these recommendations would not work properly, beginners would likely quickly stop following the suggestions from these malicious providers, limiting the impact of any attacker effort.

Open Science

To support transparency, replication, further research, and compliance with the open science policy, we will provide our interview materials, interview problem code and solutions, pre-screening questions, codebook, and timestamped segments where we applied our codebook to each interview.

We will not share the raw interview data, including interview video, audio, or transcripts, as they would de-anonymize our participants. By doing this, we minimize the risk of accidentally revealing any other information that could be used to identify our participants through contextual and meta information embedded in the raw data. Due to the extensive nature of our codebook, one can effectively reconstruct participant behaviors by analyzing the sequence of the different coded events. Therefore, we can adequately share the parts of our data that capture participant behaviors while maintaining privacy and confidentiality.

References

[1] Anonymous github of all challenge soruce code, and hint descriptions. https://github.com/Anonymous-Giraffe/Just-Google-It-Challenges.

[2] Black hat usa 2022 | briefing schedule. (Accessed 09-20-2022).

[3] Bsides las vegas 2022 | schedule. (Accessed 09-20-2022).

[4] Ctftime writeup database. https://ctftime.org/writeups.

[5] Defcon 30 hacking conference schedule. (Accessed 09-20-2022).

[6] Osf container with supplemental materials for jgi. https://osf.io/y6xew/.

[7] Owasp homepage about format string vulnerability. https://owasp.org/www-community/attacks/Format_string_attack.

[8] Picotctf binary exploitation reference sheet. https://picoctf.org/learning_guides/Book-5-Binary-Exploitation.pdf.

[9] Picotctf general reference sheet. https://picoctf.org/learning_guides/Book-1-General-Skills.pdf.

[10] Omer Akgul, Taha Eghtesad, Amit Elazari, Omprakash Gnawali, Jens Grossklags, Michelle L. Mazurek, Aron Laszka, and Daniel Votipka. Bug hunters' perspectives on the challenges and benefits of the bug bounty ecosystem. In *32nd USENIX Security Symposium*, USENIX Sec '23.

[11] Susan A Ambrose, Michael W Bridges, Michele DiPietro, Marsha C Lovett, and Marie K Norman. *How learning works: Seven research-based principles for smart teaching*. John Wiley & Sons, 2010.

[12] Computing Research Association. Generation cs: Computer science undergraduate enrollments surge since 2006. https://cra.org/data/Generation-CS/, 2017. Accessed: 2024-06-06.

[13] Nathan Backman. Facilitating a battle between hackers: Computer security outside of the classroom. In *In Proc. of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, pages 603–608, New York, NY, USA, 2016. ACM.

[14] K. Baclawski. The observer effect. In *2018 IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA)*, pages 83–89, 2018.

[15] Lisa Feldman Barrett and Daniel J. Barrett. An introduction to computerized experience sampling in psychology. *Social Science Computer Review*, 19(2):175–185, 2001.

[16] Betsy Bevilacqua. How facebook's annual "hacktober" campaign promotes cybersecurity to employees. https://hbr.org/2017/11/how-facebooks-annual-hacktober-campaign-promotes-cybersecurity-to-employees, 2017. (Accessed 05-02-2018).

[17] Raghav Bhat. Wip: Understanding vulnerability discovery in expert and novice binary analysts' behavior. In *Proceedings of the 2024 Symposium on the Science of Security*, HoTSoS '24, Virtual, 2024. National Security Agency.

[18] Joseph Biden. Executive order on improving the nation's cybersecurity, May 2021. (Accessed 07-21-2021).

[19] Jorge Blasco and Elizabeth A. Quaglia. Infosec cinema: Using films for information security teaching. In *2018 USENIX Workshop on Advances in Security Education*, ASE '18, Baltimore, MD, 2018. USENIX Association.

[20] Kevin Bock, George Hughey, and Dave Levin. King of the hill: A novel cybersecurity competition for teaching penetration testing. In *2018 USENIX Workshop on Advances in Security Education (ASE 18)*, Baltimore, MD, August 2018. USENIX Association.

[21] John D. Bransford, Ann L. Brown, and Rodney R. Cocking. *How people learn: Brain, mind, experience, and school: Expanded edition*. National Academies Press, 2000.

[22] Chuck Brooks. Alarming cyber statistics for mid-year 2022 that you need to know, June 2022. (Accessed 09-19-2022).

[23] Adam Bryant. *Understanding How Reverse Engineers Make Sense of Programs from Assembly Language Representations*. PhD thesis, US Air Force Institute of Technology, 01 2012.

[24] BugCrowd. Diversity and inclusion | bugcrowd, 2022. (Accessed 09-20-2022).

[25] Kevin Burk, Fabio Pagani, Christopher Kruegel, and Giovanni Vigna. Decomperson: How humans decompile and what we can learn from it. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 2765–2782, Boston, MA, August 2022. USENIX Association.

[26] Tanner J. Burns, Samuel C. Rios, Thomas K. Jordan, Qijun Gu, and Trevor Underwood. Analysis and exercises for engaging beginners in online CTF competitions for security education. In *2017 USENIX Workshop on Advances in Security Education (ASE 17)*, Vancouver, BC, August 2017. USENIX Association.

[27] Martin Carlisle, Michael Chiaramonte, and David Caswell. Using CTFs for an undergraduate cyber education. In *2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 15)*, Washington, D.C., August 2015. USENIX Association.

[28] Mariano Ceccato, Paolo Tonella, Cataldo Basile, Bart Coppens, Bjorn De Sutter, Paolo Falcarin,

and Marco Torchiano. How professional hackers understand protected code while performing attack tasks. In *2017 International Conference on Program Comprehension*, ICPC '17, pages 154–164, Piscataway, NJ, USA, 2017. IEEE Press.

[29] Kathy Charmaz. *Constructing Grounded Theory: A Practical Guide Through Qualitative Analysis*. SagePublication Ltd, London, 2006.

[30] William G Chase and Herbert A Simon. Perception in chess. *Cognitive psychology*, 4(1):55–81, 1973.

[31] Kevin Chung and Julian Cohen. Learning obstacles in the capture the flag model. In *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, San Diego, CA, August 2014. USENIX Association.

[32] Gregory Conti and James Caroland. Embracing the kobayashi maru: Why you should teach your students to cheat. *IEEE Security & Privacy*, 9(4):48–51, 2011.

[33] Anastasia Danilova, Alena Naiakshina, Stefan Horstmann, and Matthew Smith. Do you really code? designing and evaluating screening questions for online surveys with programmers. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 537–548, 2021.

[34] Tamara Denning, Adam Shostack, and Tadayoshi Kohno. Practical lessons from creating the control-alt-hack card game and research challenges for games in education and research. In *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education*, 3GSE '14.

[35] Debora Di Caprio, Francisco J Santos-Arteaga, and Madjid Tavana. An information retrieval benchmarking model of satisficing and impatient users' behavior in online search environments. *Expert Systems with Applications*, 191:116352, 2022.

[36] Adam Doupé, Manuel Egele, Benjamin Caillat, Gianluca Stringhini, Gorkem Yakin, Ali Zand, Ludovico Cavedon, and Giovanni Vigna. Hit 'em where it hurts: A live security exercise on cyber situational awareness. In *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC '11, page 51?61, New York, NY, USA, 2011. Association for Computing Machinery.

[37] W. Du. Seed: Hands-on lab exercises for computer security education. *IEEE Security Privacy*, 9(5):70–73, 2011.

[38] Wenliang Du. Seed: Hands-on lab exercises for computer security education. *IEEE Security & Privacy*, 9(5):70–73, 2011.

[39] Olive Jean Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64, 1961.

[40] Eldad Eilam. *Reversing: secrets of reverse engineering*. John Wiley & Sons, 2011.

[41] Margaret Ellis, Liesl Baum, Kimberly Filer, and Stephen H. Edwards. Experience report: Exploring the use of ctf-based co-curricular instruction to increase student comfort and success in computing. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, ITiCSE '21, page 303–309, New York, NY, USA, 2021. Association for Computing Machinery.

[42] Ryan Ellis and Yuan Stevens. Bounty everything: Hackers and the making of the global bug marketplace. *Available at SSRN 4009275*, 2022.

[43] Irina Ford, Ananta Soneji, Faris Bugra Kokulu, Jayakrishna Vadayath, Zion Leonahenahe Basque, Gaurav Vipat, Adam Doupé, Ruoyu Wang, Gail-Joon Ahn, Tiffany Bao, and Yan Shoshitaishvili. "Watching over the shoulder of a professional": Why Hackers Make Mistakes and How They Fix Them. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2024.

[44] Sylvain Frey, Awais Rashid, Pauline Anthonysamy, Maria Pinto-Albuquerque, and Syed Asad Naqvi. The good, the bad and the ugly: A study of security decisions in a cyber-physical systems game. *IEEE Transactions on Software Engineering*, 45(5):521–536, 2019.

[45] Karl Pearson F.R.S. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to

have arisen from random sampling. *Philosophical Magazine*, 50(302):157–175, 1900.

[46] Kelsey R Fulton, Samantha Katcher, Kevin Song, Marshini Chetty, Michelle L Mazurek, Chloé Messdaghi, and Daniel Votipka. Vulnerability discovery for all: Experiences of marginalization in vulnerability discovery. *Proc. of the IEEE*, 2023.

[47] Kelsey R. Fulton, Samantha Katcher, Kevin Song, Marshini Chetty, Michelle L. Mazurek, Chloé Messdaghi, and Daniel Votipka. Vulnerability discovery for all: Experiences of marginalization in vulnerability discovery. In *Proceedings of the 44th IEEE Symposium on Security and Privacy*, IEEE S&P '23, 2023.

[48] GDB: The gnu project debugger. https://www.gnu.org/software/gdb/, 2020.

[49] Google. Learn cybersecurity. https://learncybersecurity.withgoogle.com/. (Accessed 06-05-2024).

[50] Google. Google ctf 2019. https://g.co/ctf, 2019. (Accessed 05-27-2020).

[51] HackerOne. Home | hacker 101. (Accessed 05-21-2020).

[52] Andrew F Hayes and Klaus Krippendorff. Answering the call for a standard reliability measure for coding data. *Communication methods and measures*, 1(1):77–89, 2007.

[53] Samantha Katcher, Liana Wang, Caroline Yang, Chloé Messdaghi, Michelle L. Mazurek, Marshini Chetty, Kelsey R. Fulton, and Daniel Votipka. A survey of cybersecurity Professionals' perceptions and experiences of safety and belonging in the community. In *Twentieth Symposium on Usable Privacy and Security (SOUPS 2024)*, pages 1–20, Philadelphia, PA, August 2024. USENIX Association.

[54] Henry A Landsberger. Hawthorne revisited: Management and the worker, its critics, and developments in human relations in industry. 1958.

[55] Kees Leune and Salvatore J. Petrilli. Using capture-the-flag to enhance the effectiveness of cybersecurity education. *Proceedings of the 18th Annual Conference on Information Technology Education*, 2017.

[56] Shu-Sheng Liaw and Hsiu-Mei Huang. Information retrieval from the world wide web: a user-focused approach based on individual experience with search engines. *Computers in human behavior*, 22(3):501–517, 2006.

[57] Kuang-Chen Lu and Shriram Krishnamurthi. Identifying and correcting programming language behavior misconceptions. *Proc. ACM Program. Lang.*, 8(OOPSLA1), apr 2024.

[58] Lena Mamykina, Bella Manoim, Manas Mittal, George Hripcsak, and Björn Hartmann. Design lessons from the fastest q and a site in the west. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, page 2857–2866, New York, NY, USA, 2011. Association for Computing Machinery.

[59] Alessandro Mantovani, Simone Aonzo, Yanick Fratantonio, and Davide Balzarotti. RE-Mind: a first look inside the mind of a reverse engineer. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 2727–2745, Boston, MA, August 2022. USENIX Association.

[60] Nora McDonald, Sarita Schoenebeck, and Andrea Forte. Reliability and inter-rater reliability in qualitative research: Norms and guidelines for cscw and hci practice. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW):1–23, 2019.

[61] Jelena Mirkovic and Peter A. H. Peterson. Class capture-the-flag exercises. In *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, San Diego, CA, August 2014. USENIX Association.

[62] Jelena Mirkovic, Aimee Tabor, Simon Woo, and Portia Pusey. Engaging novices in cybersecurity competitions: A vision and lessons learned at ACM tapia 2015. In *2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 15)*, Washington, D.C., August 2015. USENIX Association.

[63] NetDiligence. Netdiligence ransomware 2022 spotlight report. Technical report, NetDiligence, Gladwyne, Pennsylvania, 2022.

[64] Government of Canada. National cyber security strategy: Canada's vision for security and prosperity in the digital age. https:

//www.publicsafety.gc.ca/cnt/rsrcs/pblctns/ntnl-cbr-scrt-strtg/index-en.aspx, 2022. (Accessed 06-05-2024).

[65] Plaid Parliament of Pwning. picoctf. (Accessed 05-27-2020).

[66] Plaid Parliament of Pwning. picoctfresources. (Accessed 01-05-2025).

[67] Kentrell Owens, Alexander Fulton, Luke Jones, and Martin Carlisle. pico-boo!: How to avoid scaring students away in a ctf competition. 2019.

[68] Annemarie Sullivan Palincsar and Ann L Brown. Reciprocal teaching of comprehension-monitoring activities. *Center for the Study of Reading Technical Report; no. 269*, 1983.

[69] Andreas Poller, Laura Kocksch, Sven Türpe, Felix Anand Epp, and Katharina Kinder-Kurlanda. Can security become a routine? a study of organizational change in an agile software development group. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '17, pages 2489–2503, New York, NY, USA, 2017. Association for Computing Machinery.

[70] P. Pusey, M. Gondree, and Z. Peterson. The outcomes of cybersecurity competitions and implications for underrepresented populations. *IEEE Security Privacy*, 14(6):90–95, 2016.

[71] Portia Pusey, Sr. David Tobey, and Ralph Soule. An argument for game balance: Improving student engagement by matching difficulty level with learner readiness. In *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, San Diego, CA, August 2014. USENIX Association.

[72] pwntools. https://docs.pwntools.com/en/stable/, 2000.

[73] Aunshul Rege and Rachel Bleiman. Collegiate social engineering capture the flag competition. In *2021 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–11, 2021.

[74] Dale C. Rowe, Barry M. Lunt, and Joseph J. Ekstrom. The role of cyber-security in information technology education. In *Proceedings of the 2011 Conference on Information Technology Education*, SIGITE '11, pages 113–122, New York, NY, USA, 2011. Association for Computing Machinery.

[75] Alan T. Sherman, David DeLatte, Michael Neary, Linda Oliva, Dhananjay Phatak, Travis Scheponik, Geoffrey L. Herman, and Julia Thompson. Cybersecurity: Exploring core concepts through six scenarios. *Cryptologia*, 42(4):337–377, 2018.

[76] Yan Shoshitaishvili and Connor) Nelson. pwn.college. (Accessed 05-27-2020).

[77] Ambareen Siraj, Sheikh Ghafoor, Joshua Tower, and Ada Haynes. Empowering faculty to embed security topics into computer science courses. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*, ITiCSE '14, page 99–104, New York, NY, USA, 2014. Association for Computing Machinery.

[78] Anselm Strauss and Juliet Corbin. *Basics of qualitative research*, volume 15. Newbury Park, CA: Sage, 1990.

[79] Thomas K. Jordan Qijun Gu Trevor Underwood Tanner J. Burns, Samuel C. Rios. Analysis and exercises for engaging beginners in online CTF competitions for security education. In *2017 USENIX Workshop on Advances in Security Education*. USENIX Association, 2017.

[80] Clark Taylor, Pablo Arias, Jim Klopchic, Celeste Matarazzo, and Evi Dube. CTF: State-of-the-Art and building the next generation. In *2017 USENIX Workshop on Advances in Security Education (ASE 17)*, Vancouver, BC, August 2017. USENIX Association.

[81] David H. Tobey, Portia Pusey, and Diana L. Burley. Engaging learners in cybersecurity careers: Lessons from the launch of the national cyber league. *ACM Inroads*, 5(1):53–56, March 2014.

[82] Sherry Turkle. *The Second Self: Computers and the Human Spirit*. Mit Press, 1984.

[83] Valgrind. https://valgrind.org/, 2000.

[84] Jeroen J. G. Van Merrienboer and John Sweller. Cognitive load theory and complex learning:

Recent developments and future directions. *Educational Psychology Review*, 17:147–177, 06 2005.

[85] Sofia Villegas. Uk government ramps up efforts to bridge cybersecurity skills gap. https://www.holyrood.com/news/view/uk-government-ramps-up-efforts-to-bridge-cybersecurity-skills-gap, 2024. (Accessed 06-05-2024).

[86] D. Votipka, E. Zhang, and M. Mazurek. Hacked: A pedagogical analysis of online vulnerability discovery exercises. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1589–1606, Los Alamitos, CA, USA, may 2021. IEEE Computer Society.

[87] Daniel Votipka, Desiree Abrokwa, and Michelle L. Mazurek. Building and validating a scale for secure software development self-efficacy. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–20, New York, NY, USA, 2020. Association for Computing Machinery.

[88] Daniel Votipka, Mary Nicole Punzalan, Seth M. Rabin, Yla Tausczik, and Michelle L. Mazurek. An investigation of online reverse engineering community discussions in the context of ghidra. In *2021 IEEE European Symposium on Security and Privacy*, EuroS&P '21, 2021.

[89] Daniel Votipka, Seth Rabin, Kristopher Micinski, Jeffrey S. Foster, and Michelle L. Mazurek. An observational investigation of reverse engineers' processes. In *29th USENIX Security Symposium (USENIX Security 20)*, Boston, MA, August 2020. USENIX Association.

[90] Daniel Votipka, Rock Stevens, Elissa M Redmiles, Jeremy Hu, and Michelle L Mazurek. Hackers vs. testers: A comparison of software vulnerability discovery processes. *Proc. of the IEEE*, 2018.

[91] Jan Vykopal and Miloš Barták. On the design of security games: From frustrating to engaging learning. In *2016 USENIX Workshop on Advances in Security Education (ASE 16)*, Austin, TX, August 2016. USENIX Association.

[92] C. Weir, L. Blair, I. Becker, A. Sasse, and J. Noble. Light-touch interventions to improve software development security. In *2018 IEEE Cybersecurity Development (SecDev)*, pages 85–93, 2018.

[93] Michael Whitney, Heather Lipford-Richter, Bill Chu, and Jun Zhu. Embedding secure coding instruction into the ide: A field study in an advanced cs course. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, pages 60–65, New York, NY, USA, 2015. Association for Computing Machinery.

[94] Miuyin Yong Wong, Matthew Landen, Manos Antonakakis, Douglas M. Blough, Elissa M. Redmiles, and Mustaque Ahamad. An inside look into the practice of malware analysis. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 3053–3069, New York, NY, USA, 2021. Association for Computing Machinery.