

Observations From an Online Security Competition and Its Implications on Crowdsourced Security

Alejandro Cuevas
Carnegie Mellon University

Emma Hogan
University of California, San Diego

Hanan Hibshi
Carnegie Mellon University

Nicolas Christin
Carnegie Mellon University

Abstract

The crowd sourced security industry, particularly bug bounty programs, has grown dramatically over the past years and has become the main source of software security reviews for many companies. However, the academic literature has largely omitted security teams, particularly in crowd work contexts. As such, we know very little about how distributed security teams organize, collaborate, and what technology needs they have. We fill this gap by conducting focus groups with the top five teams (out of 18,201 participating teams) of a computer security Capture-the-Flag (CTF) competition. We find that these teams adopted a set of strategies centered on specialties, which allowed them to reduce issues relating to dispersion, double work, and lack of previous collaboration. Observing the current issues of a model centered on individual workers in security crowd work platforms, our study cases that scaling security work to teams is feasible and beneficial. Finally, we identify various areas which warrant future work, such as issues of social identity in high-skilled crowd work environments.

1 Introduction

In the 2019 iteration of picoCTF, a two-week computer security competition with over 90,000 participants, the top three ranking teams presented a set of counter-intuitive properties. Members were all geographically distributed, their collaboration was fully online, and some team members did not even know each other, let alone had collaborated before. Yet, these teams adopted a set of strategies that allowed them to surpass

thousands of other competing teams. While the research literature on distributed teams may provide hints on the factors that contributed to these successful collaborations [18, 27, 48], little attention has been paid to how *security engineering* teams collaborate. In particular, there is scant research on the specific practices of teams that join together online, without much/any prior interaction, to complete short-term security tasks. What can we learn from the practices of these “short-term security teams,” particularly in the context of security crowdsourcing?

Security engineering manifests in a variety of forms across industry teams and may cover a wide variety of functions. In the context of this study, the functions that we consider are evaluation, vulnerability assessment, exploitation analysis, and digital forensics, as categorized by the NIST [47]. These skill sets directly overlap with those applied across many capture-the-flag competitions (CTFs), which have made CTFs a popular tool for many ends: screening potential employees (similar to coding interviews for software engineering roles) [8, 20], teaching in corporate, academic, and military settings [5, 30, 46, 62], and training future bug bounty hunters [1, 10]. For these reasons, we argue that CTF teams are a naturally good proxy to study security engineering teams, especially in distributed contexts and for short-term security tasks, such as screening code for bugs.

Today, many companies have opted to offload or complement the functions described above through crowdsourced security tasks. Bug bounty programs are among the most popular offering, although penetration testing is also increasingly gaining popularity [24]. Companies, through bug bounty programs, invite crowds of security researchers to review their software, and issue payments to researchers who report bugs. Due to the cost efficiency of such programs [21], the crowd sourced security industry has been dramatically increasing over the past couple of years. In 2019, HackerOne reported that hacker-powered security programs increased by at least 30% across each region of the world [24], while BugCrowd reported a 92% increase in reported vulnerabilities [11]. However, current crowdsourced security offerings focus only on

Copyright is held by the author/owner. Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee.

USENIX Symposium on Usable Privacy and Security (SOUPS) 2021.
August 8–10, 2021, Vancouver, B.C., Canada.

harnessing work at the individual level (i.e., slicing and distributing work to individuals), which introduces issues of work replication, reduced report quality due to competition, and attrition [66].

Scholars focusing on security crowd work have so far primarily focused on rigorous, quantitative modeling, mainly from an economics perspective [36,66], to tackle this problem. While economics-inspired approaches are relevant, we argue that the problem is ripe for investigation from a collaborative-work perspective as well, particularly given the prevalence of CTF teams who work together to efficiently solve similar problems to those found in BBPs. In discussing the future of crowd work, Kittur et al. propose a framework to scale work to support more complex tasks and increase efficiency, among other gains [33]. However, can we begin to conceptualize the future of security crowd work under this framework knowing next to nothing about how security teams operate and what their needs are? We attempt to fill this gap by focusing on security teams which embody the characteristics that teams in security crowd work would have.

To understand the dynamics, practices, and needs of high-performing short-term security teams, we conducted focus groups with the top five ranking teams from the picoCTF 2019 competition. Our study is exploratory and seeks to get a wide view of the practices of short-term security teams, from their formation to their motivations, with a focus on the practices that enabled their high performance. Our research questions are as follows:

- **RQ1:** *How, why, and through which assembly mechanisms do these teams form?*
- **RQ2:** *What technological needs do teams have and how do they address them?*
- **RQ3:** *How do teams scope and distribute tasks, roles, and responsibilities?*
- **RQ4:** *What motivations and limitations do teams have?*
- **RQ5:** *What factors contribute to performance according to the teams?*

Our results offer a first view on the practices of security teams in a distributed collaboration context. At a high level, we note how teams organically have adopted various strategies and processes which readily fit the more advanced framework of crowd work proposed by Kittur et al. [33], particularly in terms of task decomposition, hierarchical/reputational organization, task assignment, and collaboration. For instance, we find that teams adopted a role-based approach based on specialty areas, to split and assign tasks. We also find that assembly and recruitment was heavily-based on role and reputation. Looking at the current bug-bounty program models and their limitations, our observations indicate that scaling security crowd work to teams is feasible and potentially beneficial in various aspects, both for the individuals, as well as for the work platform. We conclude our study discussing other

salient observations and identifying various areas which merit future work.

2 Related Work

In this section, we review research that investigated geographically-distributed software engineering teams, security teams, and crowdsourced security.

2.1 Distributed Teams

The challenges of virtual or geographically-distributed teams have been well studied and reviewed, particularly for software engineering teams [4, 22, 51]. Since we are not aware of prior research on security teams, we survey the literature for constructs that can guide our exploration on the factors that affect virtual teams' performance. Abarca et al. conducted a systematic literature review of 2 354 studies on virtual teams between 2015 and 2019 [4], following the review by Gilson et al. which covered work from 2004 to 2014 [22], as well as the review by Powell et al. in 2004 [51]. We describe the main constructs that resulted from these reviews.

Geographic Distribution and Communication: Dispersion reduces the amount of communication across teams, increasing the difficulty in coordination and awareness, increasing delays [48]. Beyond decreased communication, however, greater spatial distribution also introduces temporal challenges. Less time overlap makes it harder for teams to adjust when unexpected problems arise and makes communication more prone to breakdown [18, 19]. In the case of software engineering teams, distribution leads to more software failures [27]. Herbsleb and Mockus posit that increased awareness, communication, and better work distribution can reduce cross-site delays [27]. Newer studies have looked at how technologies that facilitate coordination and resource sharing can help distributed teams cope with dispersion [65].

Trust: Trust is defined as “as an individual’s willingness to become vulnerable to the actions of others with the expectations that others will follow through on their commitments [43, 53].” In a study of globally distributed development teams, Al-Ani et al. found that participants described trust as expectations from their colleagues, such as in terms of technical competency [7]. Trust is in particular cultivated when teammates fulfill the positive expectations of their team members [50, 54]. Increased trust fosters knowledge sharing and coordination, and leads to greater performance [31, 44, 58].

Types of Tasks and Interdependence: The nature of tasks and their assignment impacts team performance. When members work on tasks with more interdependence, more communication and coordination is needed, particularly if these are unplanned. In software engineering teams, the high amount of interdependence between tasks is among the variables that most influence team performance [27, 35].

Cohesion and Familiarity: Cohesion relates to the sense of unity in a team [4, 14]. Physical distance can make it harder for teammates to get to know (i.e., increase familiarity with) each other [57], but given enough time distributed teams may still become cohesive through online communications [14]. Past studies have found that cohesion leads to better performance [40]. Similarly, knowing of how other members work or who knows (“team awareness”) can also increase the team’s performance [17]. As such, prior collaboration, such as “team dating”(i.e., interacting in brief tasks before choosing teams) can help members do better, particularly in ad hoc scenarios [41].

Leadership and Motivations: Team leadership can be an effective tool for increasing motivation and coordination in a team [9]. Past research, however, has found that the lack of face-to-face contact attenuates the positive effects of hierarchical leadership [9, 52]. Some scholars suggest that structural supports (such as fair and reliable reward systems) and shared leadership can complement hierarchical leadership in virtual settings to increase performance [9, 28]. Particularly, the performance gain of sharing leadership across members is based on the premise that it empowers individual members [32]. Hoch and Kozlowski found empirical support for the performance advantages of sharing leadership [29], while Zhu et al. found evidence that leadership behavior by all members in online communities increased members’ motivations [67].

2.2 Security Teams

In a 2017 NIST special publication on creating a cybersecurity workforce framework, computer security work was split in 7 categories: securely provision (SP), operate and maintain (OM), oversee and govern (OV), protect and defend (PR), analyze (AN), collect and operate (CO), investigate (IN) [47]. Each of these categories is then split into specialty areas [47]. For the purposes of this paper, the specialty areas we consider when discussing security teams in the context of CTFs and bug-bounty programs are the following: test and evaluation (TST, part of SP), vulnerability assessment and management (VAM, part of PR), exploitation analysis (EXP, part of AN), and digital forensics (FOR, part of IN).

Most of the academic work on security teams has focused on computer security incident response teams (CSIRTs). CSIRTs are part of the PR category, and the incident response (CIR) specialty area. Work CSIRTs has focused on: exploring it from an organizational psychology perspective [13], formalizing management capabilities [55], and strategies to develop shared knowledge and increase performance [60]. Beyond CSIRTs, Henshel et al. identify the need for and propose an assessment model for quantifying cyber defense teams’ proficiency [26]. Kokulu et al. studied Security Operation Centers (teams of security analysts who monitor, prevent, report, and respond to security attacks) and found that there are disagreements between managers and analysts which threaten

the efficiency and effectiveness of these security teams [34].

2.3 Bug Bounties and Crowdsourced Security

Bug bounty programs are programs offered by organizations through which external security professionals are given the opportunity to look for security bugs across the organizations’ software. Security professionals who find a bug can then submit a report to the organization and depending on the criticality of the bug, receive commensurate compensation (i.e., a bounty). The idea is that harnessing the wisdom of the crowd is more economically efficient and also results in more discovered bugs. Both claims have been validated empirically, the former by Finifter et al [21] and Walshe and Simpson [63]. and the latter by Maillart et al. [42].

Bug bounties are possibly the most popular crowdsourced security offering, with over 1400 participating organizations, 450,000 registered hackers, and more than US\$62M paid in bounties in the HackerOne platform alone [24]. However, BBPs suffer from a high number of duplicate submissions (30-40%) due to the competition between researchers. Researchers may devote weeks to the same security bug only for the first submission to be rewarded [66]. This has caused researchers to quickly flock to new programs, prioritizing higher payouts and easier bugs [42].

3 Background

We provide an overview of the nature and motivation of capture-the-flag competitions, as well as their current usage for recruitment and education. Following, we describe the picoCTF competition and its participants. We provide a sample challenge from the competition in Figure 3.2 and describe its solution to provide context on players’ skill sets and expertise.

3.1 Capture-the-flag Competitions

Capture-the-flag (CTF) competitions are events where teams of computer security experts and/or students compete to showcase their skills across various areas of security, such as memory corruption, web security, and cryptography. In jeopardy-style CTFs, puzzles typically involve finding and exploiting vulnerabilities, reverse engineering binaries, or finding information through digital forensics tools. CTFs have been steadily gaining popularity, with over 170 competitions planned for 2020 and more being announced every week [16]. While originally conceptualized as an entertaining battle of wits between talented hackers, CTFs have become much more. The NSA, along with companies like Google and Facebook, organize CTF competitions to identify and recruit security talent [2, 8, 20]. Across schools and colleges, CTF problems have been gaining popularity as a way to teach security concepts through hands-on exercises [46]. And competitions are organized to foster interest in a field that is seeing a shortage

in security professionals [30]. Finally, bug bounty platforms have also shifted to CTFs to provide education to bug bounty hunters [1, 10]. With CTFs having such a quintessential role in the education and recruitment of security professionals, we argue they are a suitable environment to study the collaboration of security teams.

3.2 The picoCTF Competition

picoCTF is a competition organized and run by Carnegie Mellon University (CMU). The organizers provided the research team with competition data and facilitated the recruitment of the top teams. While the \$14,000 prize pool is restricted to middle/high school students in the US, the competition is open and free of cost to participants of all ages from around the world. In 2019, 46,052 registered teams, of which 18,201 had at least one successful submission. The competition was open for 2 weeks, from September 26th to October 11th, 2019. Teams earned points by completing challenges. In case several teams completed all challenges, their final ranking was determined by the speed at which they solved these challenges. The competition consisted of 124 problems, broken down across 6 categories: general, cryptography, forensics, reverse engineering, binary exploitation, and web exploitation. In the the Global rankings (which includes all registered teams), only 13 teams finished all challenges in the allotted time. While picoCTF provides beginner-friendly introductory challenges, its toughest challenges require deep technical knowledge similar to the knowledge needed to work on real systems. For example, Figure 3.2 is an example of a technical problem lifted from the competition. To solve this problem, participants have to be comfortable using professional reverse engineering tools, such as Ghidra ¹, and be familiar with the GNU C Library (glibc) ², heap management techniques (in this case, `tcache` [39]), and vulnerability exploitation techniques (such as, `tcache` poisoning [59], double free errors [49], or null-byte overflows [3]).

4 Methodology

Our data analysis was based on five focus groups conducted with each of the five top-scoring CTF teams from the picoCTF 2019 competition. We invited all members (a total of 25 individuals) to participate in the focus groups; 17 members participated. We followed a semi-structured interview format, and each focus group was scheduled for 1 hour and 30 minutes. Two researchers were part of each focus group: one researcher asked questions, while the other researcher took notes. Our questions were organized across eight themes, as described in 4.1. The focus groups were held online and the meetings were recorded and then transcribed. The transcriptions were then coded under a grounded theory approach [12],

¹<https://ghidra-sre.org/>

²<https://www.gnu.org/software/libc/>

Challenge Name: “zero_to_hero” **Points:** 500 / 34201
Description: “Now you’re really cooking. Can you pwn this service?”

Goal: Participants are given access to a binary and two libraries: `libc.so.6` and `ld-2.29.so`. To complete the challenge they must exploit the binary.

Solution: To solve this problem, a participant must bypass the 2.29 `glibc` patch which fixes a double free vulnerability. To achieve this, they need to employ a null byte overflow to change a chunk’s size, use this as a double free, and then conduct a `tcache` poisoning attack to overwrite `__free_hook` and succeed in the exploit.

Figure 1: A sample security challenge from the picoCTF 2019 competition and a summarized description on how to solve it.

following open and axial procedures by three independent coders, as described in 4.2. Not all teams had full representation in the focus groups, which we discuss in 6.1, among other limitations.

4.1 Focus Group Script

Our goal was to uncover factors (e.g., practices, technologies, strategies) used by the top-ranking teams to achieve a successful collaboration. Due to the scarce literature, our interview script was built for breadth: covering a wide range of possible topics. Within these topics, we explored in more depth a particular topic if it seemed to be more salient (e.g., the team recurrently mentioned it) or if the team explicitly emphasized its importance. The list of topics we explored are as follows:

- **Formation:** composition, recruitment, assembly mechanisms, as discussed by Harris et al. [25].
- **Technology:** team’s use of current tools, their purpose, and desired/missing functionality.
- **Communication:** the media and purpose of members’ communications.
- **Collaboration:** how, when, and why team members chose to work together.
- **Dynamics:** roles, responsibilities, and decision processes (e.g., leadership).
- **Coordination:** task definition, creation, assignment, and review and feedback processes.
- **Motivation:** which incentives motivated individuals to join and stay in their teams.
- **Limitations:** what limiting factors prevented teams/individuals from succeeding.

4.2 Data Analysis

Our analysis was based on five focus groups (one per team) with a total of 17 participants. Three independent coders first listened to all focus group recordings and read each transcript. The lead author (who also led the focus groups) then grouped question/answer pairs across the 8 topics used in the interview script, keeping follow-up questions and answers together as appropriate. Transcriptions were then coded by following an open coding approach to refine and reorganize our initial themes [12]. We allowed these codes to deviate from the original question categories, creating new categories as appropriate. Our approach follows Strauss’ view [15] of grounded analysis where researchers can start from an initial coding frame and continue to refine the code book as they proceed. After a first round of coding, the researchers compared codes and discussed similarities and discrepancies across their codes and interpretations. The codes were then refined based on the discussions. We did not calculate inter-coder reliability as the goal was to identify emergent themes [45]. Following, the coders jointly worked to cluster categories hierarchically (using axial coding), seeking to more narrowly categorize and sub-categorize codes. Through this process, we focused our discussion on action strategies and consequences. For example, why did a team make a certain decision, does this decision affect communication or dynamics? Additionally, we used the subcategories to enable the creation of a comparison table as a mechanism to laterally compare teams’ practices and strategies. A result of this analysis can be observed in Table 1. Lastly, we discussed teams’ similarities and differences across each category. We describe our observations in the upcoming results section below.

4.3 Ethics

Our study was reviewed and approved by the Institutional Review Board (IRB) of our institution before any research activities began. As our research involved minors and voice recordings, we contacted the parents/guardians of the participants to obtain their written consent in addition to that of each participant. Prior to beginning the interviews, participants were once again briefed on the study, the data collection and retention policies, and ways for withdrawing from the study. Participation was voluntary and no financial incentives were provided for participation.

5 Results

We present the results from our analysis framed in the context of our research questions. With regards to performance (RQ5), we observed that many factors seemed to contribute to performance differently as described by the teams. As such, we discuss performance across each subsection, where pertinent. Commonalities across task coordination, dynamics,

Placement	Location & Collaboration	Prior Collaboration	Leadership	Task Distribution
1 st	Distributed: fully remote	Some	Informal	All specialists with sub-specialists
2 nd	Distributed: fully remote	None	Shared	All specialists with sub-specialists
3 rd	Distributed: fully remote	None	Shared	All specialists & basic generalists
4 th	Mixed: mostly remote	Some	Hierarchical	All specialists and one generalist
5 th	Mixed: mostly remote	Some	Informal	Mix of specialists and generalists

Table 1: Performance and collaborative features of the top five ranking teams

and collaboration, seem to be among the most influential factors in performance. Formation, motivation, and limitations were more heterogeneous across teams and individuals, but seemed to have a varying impacts on performance across individuals and teams. We assessed whether a certain factor had positive/negative impact on performance based on participants explicit descriptions (e.g., “this strategy helped us work faster”) and through observation of implicit themes, such as teams describing how they spent too much time looking for information on their chat logs.

5.1 Roles, Task Distribution, and Modularity

Clearly defined roles based on specialties were the cornerstone of each team. Specialties were divided based on broad categories of problems (e.g., a reverse engineering specialist), which addresses our RQ3. Formation and recruitment centered around members’ specialties. Because teams adopted a role-based task distribution approach, they recruited members based on specialty to maximize the coverage of tasks; these observations also shed light on our RQ1. Each member took ownership of a category of problems based on their specialty. This approach had multiple benefits. First, it increased efficiency by reducing redundant work (i.e., two members working on the same problem). Doing so, members were able to tackle more problems simultaneously.

T1–“When we were organizing the team each person kind of has a specific category they’re going to do. So by keeping the work to each category, we kind of just eliminated the problem of double work.”

Second, each specialist was able to lead their section and determine how and when they needed help from other team members. This is important, participants said, because problems across categories may need different types or levels of involvement from other members that may be hard to define by a non-specialist. For example, a member working on a cryptography problem may benefit most from a teammate finding information on cryptosystems that might relate to the problem at hand. Whereas a member looking for a web security vulnerability might benefit more from somebody actively

looking for potential vulnerabilities in the code. Third, it created a sense of accountability to the team. Members described not wanting to fall behind and drag their team down. Fourth, it allowed teams to be more modular, and thus if a member was unavailable for a competition, a similar specialist could more easily substitute for the missing member.

T1—*“We have a poll for who wants to participate in the CTF, and then we try to organize teams based on different categories. So each person is better at certain category, and then we make sure there’s one of each category on the team so they have everything covered.”*

Fifth, it facilitated training of new recruits. A specialist would become the go-to expert for a category, and thus new recruits who are interested in that particular specialty can be trained by the more senior members.

While the specialist model carries several benefits, two scenarios may disrupt teams. If the competition or problems involve more categories than team members, members may have to take ownership of more than one category, which may become a heavy burden depending on the quantity and difficulty of the problems across the categories they own. Furthermore, if a specialist gets stuck in their category and no one else has any expertise in the area, progress becomes stunted because other members may not be able to offer enough help or the necessary help. A few teams mitigated these downsides by having members sub-specializing in other categories and/or by each member developing a general understanding of all categories (i.e., basic generalists). That way, to mitigate the first scenario, the team can assign two sub-specialists to a category. When they are done with their main commitments, they can direct their attention to their secondary category.

T4—*“If there’s a five member limit, I always want two reversers slash binary exploitators on [the team], and then two web slash forensics people on [the team]. And then the last one is just everything else. Because if you have one person and one of those bigger categories, like web or reversing, it just gets troublesome, and they might get overwhelmed with the amount of stuff they have to do.”*

In the second scenario (i.e., a member getting stuck), the specialist would first collaborate with the sub-specialist. If they are unable to solve the problem together, they would then involve the other members. Since the other members are generalists, then they can offer additional insights and also know how to better support (e.g., whether to write code, or what online articles may be related). Thus, while having highly talented individuals in a team is important, having the right team composition and task distribution makes the team significantly more efficient.

T2—*“We didn’t really have anyone that specialized in reversal problems. We all worked together on that. We all bounced ideas around and checked. Because we do have an overlap in skills. That helps a lot, if we’re stuck.”*

Observation #1: Teams organize and distribute tasks across roles based on computer security specialties (e.g., web security, cryptography) to reduce redundant work, increase team modularity, distribute accountability, and to allow each member to define how they want their teammates to provide help when needed.

5.2 Performance, Trust, and Leadership

Performance in previous competitions was the most important signal that members use to determine the credibility of potential recruits. Within the team, this indicator may also be used implicitly to determine leadership and as a proxy for trust as defined in Section 2 (i.e., “expectation that others will follow through with their commitment”). These results shed more light on our **RQ3**.

T5—*“The best way to see how someone would do on a CTF is to just look at their performance on previous ones.”*

T1—*“Competing in past CTFs before would be a really good indicator, I think. And of course, specialization in a category that we need more skill in for the team.”*

This signal is easily quantified by three factors: the number of problems an individual has solved, the difficulty of the problems solved, and the speed to solve these problems. This information is typically available at the team level. However, it also informally transmitted by word-of-mouth or inferred by an individual’s participation in previous CTFs or their contribution of write-ups (tutorials on how to solve specific problems). This metric, because it is seen as an objective measure of individual abilities, is used when recruiting new members as a predictor of their performance. Additionally, it seems as if performance can be used as a mediator for trust between members. That is, because members take ownership of task portions, other members may feel more at ease if they perceive their teammate(s) as more capable, based on their past performance. Similarly, we observed that informal leadership across teams seemed to gravitate to members who had higher performance signals. On the other hand, we also observed that leadership was more distributed across teams were all members seemed to have similar performance signals. For instance, members of T1 mentioned one teammate as being the “unspoken leader.” This member also had substantial experience participating in other competitions and doing bug bounties. Similarly, T5 also had two members who

did not denominate themselves as leaders, but made a lot of the decision in terms of organization and task distribution. T4 had a similar member as T1, but in their case they decided to denominate this member the team captain. In the case of T2 and T3, all members rated their abilities at the same level. At the same time, both of these teams described fully democratic decision-making processes: “someone has an idea and we all agree or we disagree.”

Observation #2: Past CTF performance is used as a strong determinant of future performance and thus, is an important metric used during member recruitment, along with role. An individual’s past performance seemingly correlates with their perceived capability. Consequently, members with higher perceived performance or knowledge, may informally raise as leaders unless the perceived performance/abilities is balanced across members.

5.3 Technology Usage and Needs

Apart from traditional voice and text-based communication needs, all teams mentioned the need for a mechanism to keep track of task ownership and progress, which addresses our **RQ2**. All teams used Discord as their primary communication platform. Discord is a communication software which features text and VoIP channels, with rich integration capabilities and fine-grained access control.³ Apart from text and voice, users can share images, files, and stream their screen with other members of the server. Teams used text channels for asynchronous communications and disjointed conversations on various topics, with the option to directly talk about something using the voice channel.

Task visibility and tracking was a harder problem, and teams experimented with different approaches. One team used Trello, a Kanban-style web-based application where users can make lists, assign roles, and specify deadlines⁴. The team mainly used it to leave notes under each problem, but seemed to not find it very valuable.

T3– “[Trello] was okay I think. We didn’t use Trello for [a subsequent CTF]. So I don’t think it was completely necessary [for picoCTF]. But it was sort of helpful to keep track of which problems people are working on and also for keeping track of notes so we don’t have to keep searching through the Discord history.”

Another team used Google Sheets for the same purposes. The other teams mostly relied on Discord, but often griped about conversations being mixed up, or retrieving information from the chat history. Alternative strategies involved creating a channel for each problem (the team mentioned it did not work well) and creating a channel for each category, which

³<https://discord.com/>

⁴<https://trello.com>

had better results, but was still prone to information retrieval issues. Furthermore, teams also mentioned they lacked a good way of sharing larger files or code snippets.

T5– “Okay, so one annoying thing about Discord [for sharing code] is that there’s a message limit of 2000 characters... it’ll just complain and say you have to upload a file instead. [Another problem is that] I don’t think any other platform even has like code highlighting, you can’t even put code blocks with highlighting although maybe Slack can do that?”

Observation #3: Text and voice communication were adequately addressed by existing software. On the other hand, teams struggled with task visibility (i.e., awareness of what each member is working on), tracking (i.e., tracking progress and resources on each task), and resource sharing and retrieval (e.g., code and files). Teams experimented with various software but no tool seemed to suit all their needs.

5.4 Formation, Motivations, and Limitations

Assembly mechanisms, reasons to join together, and familiarity between members were different across all teams: from friends who went to the same school, to strangers who found each other through CTF-oriented Discord channels (**RQ1**). On the other hand, motivations (why members sought to do well) across all teams and members were mostly similar (although ranked differently by each member), which addresses our **RQ4**. Members often cited a combination of incentives: educational or professional development, entertainment, socialization, prizes, and measuring performance. These factors (including assembly) seemed to have little impact on performance and collaboration as expressed by the teams, except for one team (T3). One team mentioned that they formed the team and recruited members with the goal of earning a top score in the competition. When screening recruits, the founding members as well as the prospective recruits explicitly stated that they wanted to “place” in the competition (i.e., enter the top ranking in the score ladder). This common motivation, as described by participants, played a role when preparing and during the competition.

T3– I had been growing my skills from last year. And so I thought if I joined a team, I’d have, like they said, a chance of getting up there in the top five. And so when I saw that message reaching out from a strong team, it sounded like a good opportunity.

T3– “Yeah, for me, it was mostly the same reasons. I wanted to place and I also saw that they were running [their own CTF]. It showed that they were good. So that also influenced me. I think it has to do with sort of the goal. We were trying to place really well, as a competitive team. ”

Setting the common goal of doing well in the competition seemingly increased their motivation, and potentially their performance as well, as it increased the accountability of each member in achieving that goal. Furthermore, since the team was fully democratic, we hypothesize that having an aligned goal facilitated decision-making since the interests of one member would not conflict with another.

Observation #4: While motivation across team members was heterogeneous across most teams, it reportedly played a big role in the performance of one team (T3). This suggests that while common motivation might not be a requirement for a successful collaboration, it may become an enhancer.

While most discussions on limitations were straightforward (e.g., lack of technology functionality, lack of expertise/time for certain problems), some members discussed their gripes with diversity and their social identity self-presentation, which adds another dimension to our **RQ4**. CTF competitions, because they are conducted online and communication in larger groups is typically text-based, often allow participants to limit their self-presentation to a mere online handle. Members from two teams mentioned that this anonymity was relieving, since their abilities were not subject to presumptions based on things like their gender or race. However, when this anonymity was no longer in place, they felt pressured to compensate based on their social identity. Across the five teams (25 participants), only three participants were female (two of which were present in the focus groups) and only one participant was from an under-represented group.

P0⁵—*“I guess being aware that I’m the only female in a group, might add a bit of pressure I suppose. Like sometimes I think that if I say something strange or if I mess up somewhere, I might be confirming the assumption that females aren’t as good at CS, or maybe I might be confirming certain stereotypes about my gender.”*

P1—*“I think the online aspect where we’re just all like, you know, playing together, nobody knows where you’re from definitely helps or at least makes it ([issues of diversity]) less of a problem.”*

Observation #5: Members may self-censor their self-presentation (particularly their social identity) across online platforms to avoid presumptions of their abilities.

Observation #6: Members may feel pressured to perform to a certain level to prove that their social identity does not inhibit their abilities.

⁵We omit mentioning the team of the above two members (P0 and P1) to preserve their privacy.

6 Discussion, Recommendations, and Future Work

The security teams in this study developed a variety of strategies to cope with the multiple known challenges related to dispersed collaboration. At the core of these strategies was the usage of roles to determine coordination (**Obs. 1**), communication (**Obs. 1**), recruitment (**Obs. 2**), and technology usage (**Obs. 3**). This role-based approach attenuated many of the known downsides of dispersed teams, such as by reducing interdependencies and the amount of coordination and communication needed. We also found that teams assembled in a variety of ways, through various channels, and for various reasons (**Obs. 4**), but that quantified performance and roles heavily affected the assembly process (**Obs. 2**). Finally, we found that participants from under-represented groups faced a set of unique challenges that ought to be accounted for in the design of online communities (**Obs. 5-6**). Our observations shed light on all of our proposed research questions (**RQ1-5**). We next discuss the implications of our results across various aspects, make suggestions, and identify areas which merit future exploration.

Security and Software Engineering Teams. While many of the problems caused by dispersion documented in the context of software engineering might readily explain the issues that security engineering teams may face, it does not seem like the same remedies extend. We find that security teams cope with dispersion challenges differently than their software engineering counterparts. In our study, teams employed a role-based approach based on separate specialties to determine to determine work distribution and collaboration interactions (**Obs. 1**). Such an approach is likely not possible for software engineers without a prior planning due to the interdependencies that arise when building software and the lack of clear cut roles.

Thus, because the nature of the work and its requirements are different, treatments conceptualized in the context of software engineering teams (such as radical collocation [61]) may have a lesser impact on security teams. In fact, two teams in our panel had the possibility to work in the same physical space, yet still opted for asynchronous remote work.

We believe that team-oriented research for security teams is imperative given the continuous growing need for security services, professionals, and teams. While our study focused on teams that encompasses various capabilities, we note that there exist various other types of security teams which remain unexplored, such as red/blue teams and penetration testers, both in industry as well as in crowd work contexts.

Technology Needs. Different practices tend to imply different needs, and consequently call for different supporting tools. Given the lack of academic work on security teams, it is unsurprising that there are no tailored tools for the collaboration needs these teams described. This is evidenced in our study, whereby teams juggled a variety of tools to cope

with their task visibility and knowledge sharing needs (**Obs. 3**). In software engineering, real-time shared environments have been a popular approach to improving visibility and mitigating coordination costs [37, 38, 56]. For security teams which mostly focus on reviewing code, existing issue tracking systems (such as GitHub’s) might be adequate if the project allows for a priori task segmentation. However, for faster paced environments, such as a newly launched bug-bounty program, non-real-time issue tracking systems may fall short. Because there may not be sufficient information (e.g., which parts of the codebase contain cryptography bugs) nor time for a priori coordination, short-term security teams would likely benefit from real-time systems which provide information on the portions (e.g., code segments) that other analysts might be working on and the types of tasks or bugs they are testing for (e.g., web bugs). Additionally, it is unclear what tools readily and acceptably extend to other tasks beyond code reviews, such as testing remote server deployments. How should tests of applications or configurations be tracked and made visible? Or how should analysts share helpful resources and strategies?

Teams in Security Crowd Work. In the context of security crowd work, and in particularly bug-bounty programs, the current paradigm is focused on harnessing the expertise of various individuals. However, the current system is inefficient, and invites a lot of duplicate and invalid reports caused by competition among individuals [66]. Our study cases that short-term security teams are viable and could help mitigate efficiency issues, as evidenced by team strategies to reduce duplicate and redundant work. Participants in our study were able to efficiently self-organize relying only on roles and performance (**Obs. 2**), two signals that are already available across bug bounty platforms. Allowing users in bug-bounty programs to self-assemble may allow for a more efficient allocation of users’ abilities within a program. Thus, rather than all participants analyzing everything at the same time, crowd workers can spend more dedicated effort in specific sections. Furthermore, the effectiveness of the program is also poised to increase, as crowd workers can now collaborate and share knowledge, potentially increasing the complexity of bugs they are able to catch while also speeding up the process. Finally, we also envision a benefit across crowdsourced security communities, as team assembly might motivate and allow for more opportunities for professional development and education of newer crowd workers, which have been motivating factors both for our participants (**Obs. 4**), as well as for joining BBPs according to bug-bounty hunters [6]. Naturally, questions of resource pooling or team-level competition may arise in this new paradigm. To further explore this proposition, a field study with bug-bounty hunters in teams should be conducted.

Diversity and Self-Censoring. Women and participants from under-represented groups in our study saw the anonymity shield granted by online participation as a beneficial feature. This anonymity allowed them to participate in online CTF communities without having their gender or

race being subject to the judgements or presumptions that other members in the community may have. To achieve this, however, members may self-censor their presentation (**Obs. 5**) across these communities or even within *ad-hoc* teams they become part of. Within teams, they may also feel pressured to perform to a certain standard (**Obs. 6**)—which is basically relatively close to being an impostor syndrome of sorts. With regards to majority members in the CTF community, perhaps there is also an issue of abstaining from diversity such as the one described by Gómez-Zára et al. [23], which under-represented participants may be experiencing. Past research has shown that brief social-belonging interventions can improve academic and health outcomes of under-represented students [64]. These observations could raise questions on how to design online communities, particularly in specialized crowd work settings, to mitigate feelings of exclusion based on social identity. For example, sites like HackerOne or Bugcrowd may consider always keeping personal profiles optional. We recognize that our sample is already small and under-represented participants were also a minority in our sample (3 out of 17 participants). While we do not claim that this is a phenomenon which affects all CTF players from under-represented groups, a previous experiential paper on using CTF problems with high school students found a significant difference between male and female students’ experiences. The researchers note that “the male students thought the [CTF] activities were more enjoyable and interesting than female students” [30]. Given our results, we believe that this issue is not due to lack of interest, but rather related to the pressure to perform our participants describe to this issue. Future studies ought to explore this issue to begin addressing issues of diversity in the education and profession of computer security.

6.1 Limitations

We identify two main limitations with regards to our study and the implications of our findings. The first is that the teams we observed (teams formed for a CTF competition) are not a strict subset of neither industry security engineering teams nor of teams of bug-bounty hunters, and as such, may affect the generalizability of our results. Because bug-bounty teams remain, at the time of writing, quite rare compared to solo operations, we argue that CTF teams are a good proxy to study the feasibility of implementing team-based bug-bounty programs. We posit that CTF teams are a better proxy than industry computer security teams because the CTF teams we studied mostly self-assembled online, had no prior collaboration, and worked on a time-constrained task (two weeks). Additionally, bug-bounty hunters are often CTF players as well. For instance, members across the teams we studied had submitted more than 50 bug reports at the time of interview for organizations such as the Department of Defense. Furthermore, CTF challenges are increasingly being used to train

practitioners in computer security across classrooms [46] and even by bug-bounty programs on their educational platforms [1, 10].

Secondly, not all members were able to participate in the focus groups. In particular, one team was only represented by one member. We tried to account for this by framing the focus group around collective group practices, i.e., practices that all members took part of so that each member was able to talk about them. For example, a single member was still fully capable answering questions about the technology needs of their team, their self-assembly, their communication practices, etc. While we believe this approach allowed us to have a good picture of each team, we note that in some cases having additional responses from each member added nuance to the team dynamics.

7 Conclusion

We presented a first view on the practices and needs of the security teams who participated in an online security competition (picoCTF 2019). The teams we studied organically developed many organizational algorithms to maximize their efficiency and reach a top spot among a very large number of competitors. Surprisingly to us, the top three teams seemed to be unhindered by the many challenges posed by dispersed collaboration. Equally surprising is that the fourth and fifth place teams, despite being in the same geographic area (students at the same school), opted to conduct most of their work online, collaborating asynchronously. These teams placed member specialization at the core of their decision-making, from formation to task distribution. This approach seemed to yield performance gains in various aspects, such as the ability to get unstuck from areas in which the team lacked expertise, or the ability to minimize redundant work. Beyond providing a first glance at the practices and needs of successful security teams, our findings shed light on a possible path forward for the future of the emerging field of crowdsourced security. Scaling crowdsourced security offerings beyond individuals not only has the potential to increase the efficiency of tasks, but can likely offer many benefits to the overall community. For instance, more opportunities for crowdworkers' development may arise, through knowledge sharing. At the same time, some of our findings—such as the poignant examples of members preferring not to disclose their gender lest their competences were called into question—make resoundingly clear the importance to assess issues of exclusion based on social identity, particularly in collaborative crowd work environments. Our work also sheds light on the large research gaps that exist in studying security teams: the area is ripe with opportunities for team-oriented scholarship and technology development. From the teams' technology usage—adapting and reworking tools which just do not seem to fit the bill—to better understanding to which extent our findings carry over to professional bug-hunters, to corporate security teams, there

seems to be plenty of room for improvement, particularly as we continue to learn more about the practices of more types of security teams.

References

- [1] Hacker101 CTF, 2020. <https://ctf.hacker101.com/>.
- [2] NSA Codebreaker Challenge, 2020. <https://codebreaker.ltsnet.net/resources>.
- [3] 0x00sec The Home of the Hacker. Null byte poisoning - the magic byte, 2017. <https://0x00sec.org/t/null-byte-poisoning-the-magic-byte/3874>.
- [4] Víctor M Garro Abarca, Pedro R Palos-Sanchez, and Enrique Rus-Arias. Working in virtual teams: A systematic literature review and a bibliometric analysis. *IEEE Access*, 8:168923–168940, 2020.
- [5] William J Adams, Efstratios Gavas, Timothy H Lacey, and Sylvain P Leblanc. Collective views of the nsa/css cyber defense exercise on curricula and learning objectives. In *CSET*, 2009.
- [6] Omer Akgul, Taha Eghtesad, Amit Elazari, Omprakash Gnawali, Jens Grossklags, Daniel Votipka, and Aron Laszka. The hackers' viewpoint: Exploring challenges and benefits of bug-bounty programs. 2020.
- [7] Ban Al-Ani, Matthew J Bietz, Yi Wang, Erik Trainer, Benjamin Koehne, Sabrina Marczak, David Redmiles, and Rafael Prikladnicki. Globally distributed system developers: their trust expectations and processes. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work, CSCW '13*, page 563–574. Association for Computing Machinery, 2013.
- [8] Josh Armour. Announcing google capture the flag 2017, 2017.
- [9] Bradford S Bell and Steve WJ Kozlowski. A typology of virtual teams : Implications for effective leadership. *Group & Organization Management*, 27(1):14–49, 2002.
- [10] Bugcrowd. Levelup0x07 - hack another day bugcrowd. (<https://levelup.bugcrowd.com/>).
- [11] Bugcrowd. Priority one: The state of crowd-sourced security in 2019. Technical report, 2019. <https://www.bugcrowd.com/resources/reports/priority-one-report/>.
- [12] Kathy Charmaz. *Constructing grounded theory*. SAGE Publications, 2 edition, 2014.

- [13] Tiffani R Chen, Daniel B Shore, Stephen J Zaccaro, Reeshad S Dalal, Lois E Tetrick, and Aiva K Gorab. An organizational psychology perspective to examining computer security incident response teams. *IEEE Security & Privacy*, 12(5):61–67, 2014.
- [14] Laku Chidambaram. Relational development in computer-supported groups. *MIS Q.*, 20:143–165, 1996.
- [15] Juliet Corbin and Anselm Strauss. *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage, 2007.
- [16] CTFTIME. CtfTime.org / all about ctf (capture the flag), 2020. <https://ctftime.org/event/list/>.
- [17] Paul Dourish and Victoria Bellotti. Awareness and coordination in shared workspaces. In *Proceedings of the 1992 ACM Conference on Computer Supported Cooperative Work, CSCW '92*, page 107–114. Association for Computing Machinery, 1992.
- [18] J Alberto Espinosa and Erran Carmel. The impact of time separation on coordination in global software teams: a conceptual foundation. *Software Process: Improvement and Practice*, 8(4):249–266, 2003.
- [19] J Alberto Espinosa and Cynthia Pickering. The effect of time separation on coordination processes and outcomes: A case study. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences, HICSS'06*. IEEE, 2006.
- [20] Facebook. Facebook ctf is now open source! Facebook, 2016. <https://www.facebook.com/notes/facebook-ctf/facebook-ctf-is-now-open-source/525464774322241/>.
- [21] Matthew Finifter, Devdatta Akhawe, and David Wagner. An empirical study of vulnerability rewards programs. In *Proceedings of the 22nd USENIX Conference on Security, SEC'13*, page 273–288. USENIX Association, 2013.
- [22] Lucy L Gilson, M Travis Maynard, Nicole C Jones Young, Matti Vartiainen, and Marko Hakonen. Virtual teams research: 10 years, 10 themes, and 10 opportunities. *Journal of Management*, 41(5):1313–1337, 2015.
- [23] Diego Gómez-Zarà, Mengzi Guo, Leslie A. DeChurch, and Noshir Contractor. The impact of displaying diversity information on the formation of self-assembling teams. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, CHI '20*, page 1–15, New York, NY, USA, 2020. Association for Computing Machinery.
- [24] HackerOne. The hacker-powered security report 2019. Technical report, 2019. <https://www.hackerone.com/resources/reporting/the-hacker-powered-security-report-2019>.
- [25] Alexa M Harris, Diego Gómez-Zarà, Leslie A DeChurch, and Noshir S Contractor. Joining together online: The trajectory of csw scholarship on group formation. *Proceedings of the ACM on Human-Computer Interaction*, 3, 2019.
- [26] Diane S Henshel, Gary M Deckard, Brad Lufkin, Norbou Buchler, Blaine Hoffman, Prashanth Rajivan, and Steve Collman. Predicting proficiency in cyber defense team exercises. In *IEEE Military Communications Conference, (MILCOM '16)*, pages 776–781. IEEE, 2016.
- [27] James D. Herbsleb and Audris Mockus. An empirical study of speed and communication in globally distributed software development. *IEEE Transactions on Software Engineering*, 29(6):481–494, 2003.
- [28] Pamela J. Hinds and Sara Kiesler. *Distributed Work*. MIT Press, 2002.
- [29] Julia E Hoch and Steve WJ Kozlowski. Leading virtual teams: Hierarchical leadership, structural supports, and shared team leadership. *Journal of Applied Psychology*, 99(3):390, 2014.
- [30] Ge Jin, Manghui Tu, Tae-Hoon Kim, Justin Heffron, and Jonathan White. Game based cybersecurity training for high school students. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education, SIGCSE '18*, page 68–73. Association for Computing Machinery, 2018.
- [31] Gareth R Jones and Jennifer M George. The experience and evolution of trust: Implications for cooperation and teamwork. *The Academy of Management Review*, 23(3):531–546, 1998.
- [32] Bradley L. Kirkman, Benson Rosen, Paul E. Tesluk, and Cristina B. Gibson. The impact of team empowerment on virtual team performance: The moderating role of face-to-face interaction. *Academy of Management Journal*, 47(2):175–192, 2004.
- [33] Aniket Kittur, Jeffrey V. Nickerson, Michael Bernstein, Elizabeth Gerber, Aaron Shaw, John Zimmerman, Matt Lease, and John Horton. The future of crowd work. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work, CSCW '13*, page 1301–1318. Association for Computing Machinery, 2013.
- [34] Faris Bugra Kokulu, Ananta Soneji, Tiffany Bao, Yan Shoshitaishvili, Ziming Zhao, Adam Doupé, and Gail-Joon Ahn. Matched and mismatched socs: A qualitative study on security operations center issues. In

Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS '19, page 1955–1970. Association for Computing Machinery, 2019.

- [35] Claus W Langfred. Autonomy and performance in teams: The multilevel moderating effect of task interdependence. *Journal of Management*, 31(4):513–529, 2005.
- [36] Aron Laszka, Mingyi Zhao, and Jens Grossklags. Banning misaligned incentives for validating reports in bug-bounty platforms. In *European Symposium on Research in Computer Security, ESORICS '16*, pages 161–178. Springer, Cham, 2016.
- [37] Thomas D. LaToza, W. Ben Towne, Christian M. Adriano, and André van der Hoek. Microtask programming: Building software with a crowd. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology, UIST '14*, page 43–54. Association for Computing Machinery, 2014.
- [38] Sang Won Lee, Yan Chen, Noah Klugman, Sai R. Gouravajhala, Angela Chen, and Walter S. Lasecki. Exploring coordination models for ad hoc programming teams. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems, CHI EA '17*, page 2738–2745. Association for Computing Machinery, 2017.
- [39] The GNU C Library. Memory allocation tunables (the gnu c library). https://www.gnu.org/software/libc/manual/html_node/Memory-Allocation-Tunables.html.
- [40] Jeremy S Lurey and Mahesh S Raisinghani. An empirical study of best practices in virtual teams. *Information & Management*, 38(8):523–544, 2001.
- [41] Ioanna Lykourantzou, Robert E. Kraut, and Steven P. Dow. Team dating leads to better online ad hoc collaborations. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work, CSCW '17*, pages 2330–2343. Association for Computing Machinery, 2017.
- [42] Thomas Maillart, Mingyi Zhao, Jens Grossklags, and John Chuang. Given enough eyeballs, all bugs are shallow? Revisiting Eric Raymond with bug bounty programs. *Journal of Cybersecurity*, 3(2):81–90, 2017.
- [43] Roger C Mayer, James H Davis, and F David Schoorman. An integrative model of organizational trust. *The Academy of Management Review*, 20(3):709–734, 1995.
- [44] Roger C Mayer and Mark B Gavin. Trust in management and performance: Who minds the shop while the employees watch the boss? *Academy of Management Journal*, 48(5):874–888, 2005.
- [45] Nora McDonald, Sarita Schoenebeck, and Andrea Forte. Reliability and inter-rater reliability in qualitative research: Norms and guidelines for cscw and hci practice. volume 3 of *CSCW '19*, New York, NY, USA, November 2019. Association for Computing Machinery.
- [46] Jelena Mirkovic and Peter AH Peterson. Class capture-the-flag exercises. In *USENIX Summit on Gaming, Games, and Gamification in Security Education, 3GSE '14*. USENIX Association, 2014.
- [47] William Newhouse, Stephanie Keith, Benjamin Scribner, and Greg Witte. National initiative for cybersecurity education (NICE) cybersecurity workforce framework. Technical Report 2017, National Institute of Standards and Technology, 2017.
- [48] Gary M Olson and Judith S Olson. Distance matters. *Human-Computer Interaction*, 15(2):139–178, 2000.
- [49] OWASP. Doubly freeing memory. https://owasp.org/www-community/vulnerabilities/Doubly_freeing_memory.
- [50] Gabriele Piccoli, Anne Powell, and Blake Ives. Virtual teams: team control structure, work processes, and team effectiveness. *Information Technology & People*, 17, 2004.
- [51] Anne Powell, Gabriele Piccoli, and Blake Ives. Virtual teams: A review of current literature and directions for future research. *SIGMIS Database*, 35(1):6–36, 2004.
- [52] Radostina K Purvanova and Joyce E Bono. Transformational leadership in context: Face-to-face and virtual teams. *The Leadership Quarterly*, 20(3):343–357, 2009.
- [53] Lionel P. Robert. Monitoring and trust in virtual teams. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work, CSCW '16*, page 245–259. Association for Computing Machinery, 2016.
- [54] Lionel P Robert, Alan R Denis, and Yu-Ting Caisy Hung. Individual swift trust and knowledge-based trust in face-to-face and virtual team members. *Journal of Management Information Systems*, 26(2):241–279, 2009.
- [55] Robin Ruefle, Audrey Dorofee, David Mundie, Allen D Householder, Michael Murray, and Samuel J Perl. Computer security incident response team development and evolution. *IEEE Security & Privacy*, 12(5):16–26, 2014.

- [56] Stephan Salinger, Christopher Oezbek, Karl Beecher, and Julia Schenk. Saros: An eclipse plug-in for distributed party programming. In *Proceedings of the ICSE Workshop on Cooperative and Human Aspects of Software Engineering*, CHASE '10), page 48–55. Association for Computing Machinery, 2010.
- [57] Wm David Salisbury, Traci A Carte, and Laku Chidambaram. Cohesion in virtual teams: validating the perceived cohesion scale in a distributed setting. *SIGMIS Database*, 37(2-3):147–155, 2006.
- [58] John Schaubroeck, Simon SK Lam, and Ann Chunyan Peng. Cognition-based and affect-based trust as mediators of leader behavior influences on team performance. *Journal of Applied Psychology*, 96(4):863, 2011.
- [59] Cesare Silvio. Linux heap tcache poisoning, 2019.
- [60] Julie Steinke, Balca Bolunmez, Laura Fletcher, Vicki Wang, Alan J Tomassetti, Kristin M Repchick, Stephen J Zaccaro, Reeshad S Dalal, and Lois E Tetrick. Improving cybersecurity incident response team effectiveness using teams-based research. *IEEE Security & Privacy*, 13(4):20–29, 2015.
- [61] S. D. Teasley, L. A. Covi, M. S. Krishnan, and J. S. Olson. Rapid software development through team collocation. *IEEE Transactions on Software Engineering*, 28(7):671–683, 2002.
- [62] Daniel Votipka, Michelle L Mazurek, Hongyi Hu, and Bryan Eastes. Toward a field study on the impact of hacking competitions on secure development. In *Workshop on Security Information Workers (WSIW)*, 2018.
- [63] Thomas Walshe and Andrew Simpson. An empirical study of bug bounty programs. In *IEEE 2nd International Workshop on Intelligent Bug Fixing*, IBF '20, pages 35–44. IEEE, 2020.
- [64] Gregory M Walton and Geoffrey L Cohen. A brief social-belonging intervention improves academic and health outcomes of minority students. *Science*, 331(6023):1447–1451, 2011.
- [65] Jeffrey Warshaw, Steve Whittaker, Tara Matthews, and Barton A. Smith. When distance doesn't really matter: Effects of geographic dispersion on participation in online enterprise communities. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work*, CSCW '16, page 335–345. Association for Computing Machinery, 2016.
- [66] Mingyi Zhao, Aron Laszka, Thomas Maillart, and Jens Grossklags. Crowdsourced security vulnerability discovery: Modeling and organizing bug-bounty programs. In *The HCOMP Workshop on Mathematical Foundations of Human Computation*, 2016.
- [67] Haiyi Zhu, Robert Kraut, and Aniket Kittur. Effectiveness of shared leadership in online communities. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 407–416. Association for Computing Machinery, 2012.