Netsiege: A Long-Form, Club-Led King-of-the-Hill Competition for Accessible Cybersecurity Training

Steven Ngo
sngo1@uci.edu
University of California, Irvine
Irvine, California, USA

Akshay Rohatgi
arohatg2@uci.edu
University of California, Irvine
Irvine, California, USA

Joshua Garcia
joshug4@uci.edu
University of California, Irvine
Irvine, California, USA

Abstract

Collegiate cybersecurity competitions are an established modality for hands-on cybersecurity education, but their dominant short-form, weekend-long format favors experienced teams, leaves little time to learn within the environment, and is prone to a snowball effect that drives less competitive teams to disengage. We present *Netsiege*, a long-form King-of-the-Hill competition designed to couple offense and defense and to be run by a student cybersecurity club without formal instructors or a dedicated budget, providing accessible training for newcomers. Netsiege is designed to unfold over several weeks of club meetings and expand in scale across phases, deliberately leveraging distributed practice. Teams begin with no access and must break into a role-based enterprise network, then defend the access they gain. We describe the competition design, an open-source scoring toolchain with privilege-tiered control checks, and a reproducible provisioning-and-validation pipeline intended to lower the development burden for adopters. We report on a pilot with a novice-leaning cohort of 12 participants, drawing on shell-level command telemetry and surveys. We reflect on what aspects of Netsiege were effective or ineffective; in particular, coupling offense and defense through scoring incentives alone proved insufficient to balance the two, leaving defense under-practiced. We translate these observations into revisions for practitioners.

CCS Concepts

• **Social and professional topics** → **Informal education**; • **Security and privacy** → **Software and application security**; • **Applied computing** → **Collaborative learning**.

Keywords

Cybersecurity education, Experiential learning, Cybersecurity competitions, Student cyber clubs, Extracurricular learning, Software and application security

ACM Reference Format:
Steven Ngo, Akshay Rohatgi, and Joshua Garcia. 2027. Netsiege: A Long-Form, Club-Led King-of-the-Hill Competition for Accessible Cybersecurity Training. In *Proceedings of the 58th ACM Technical Symposium on Computer Science Education (SIGCSE TS 2027).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

SIGCSE TS 2027, Sacramento, CA

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

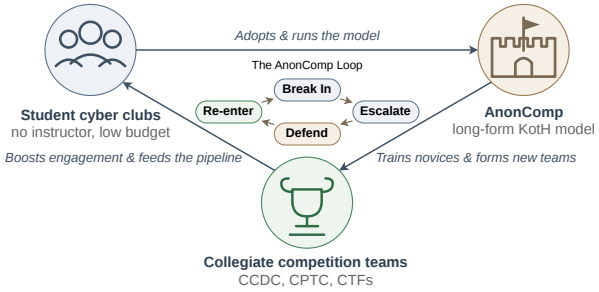

Figure 1: Netsiege situated in the cyber-club ecosystem.

1 Introduction

Experiential learning is widely regarded as essential for developing cybersecurity competencies, yet computing curricula often abstract security away or offer little hands-on experience outside the classroom [9, 17, 29]. Collegiate cybersecurity competitions have emerged as one of the most effective modalities for filling this gap, exposing students to adversarial reasoning, realistic systems, and practice that transfers to industry [4, 23]. Among colleges and universities, student-run cybersecurity clubs (i.e., cyber clubs) are frequently the primary facilitators of collegiate competitions, organizing teams, running training, and facilitating peer-led learning [20, 27, 30]. Clubs are a venue where informal security education already happens across many colleges and universities.

Despite their central role in security education, a majority of clubs struggle to benefit from the dominant competition format. Established and prestigious competitions—such as the Collegiate Cyber Defense Competition (CCDC) [12], the Collegiate Penetration Testing Competition [24], and Capture the Flag (CTF) contests (e.g., PlaidCTF [21], HITCON CTF [13])—compress activity into 48-to-72 hour windows, so the skills exercised are largely acquired in preparation rather than during the event itself [28]. Attack-defense and defense-focused competitions such as CCDC also tend to favor experienced teams and are prone to a "snowball effect", where an early lead compounds into an insurmountable advantage and discourages less competitive teams from continuing [23]. Newer teams face a high barrier to entering and succeeding in more difficult competitions [4, 22] and often lack the vocabulary and context to train themselves [14]. Newer members rarely have time in these compressed time windows to learn while also competing, and clubs typically operate with no formal course of record (e.g., a for-credit "Introduction to Cybersecurity" course) and minimal budget for training infrastructure [30].

To address the gap in novice-focused competitions and training mechanisms for cyber clubs aiming to compete in established events, we present **Netsiege**, a long-form, club-led King-of-the-Hill (KotH) competition designed for accessible cybersecurity training.

In a KotH competition, teams compete to gain and then retain control of shared machines, earning points for as long as they hold a target against other teams. Rather than a single timed session, Netsiege is designed to run across several weeks of club meetings and expands across phases, leveraging distributed practice so that participants revisit techniques over spaced sessions [7]. Teams begin with no access and must enumerate and break into a mock role-based enterprise network under black-box conditions. Teams then have to defend the footholds they gain in the network environment, with scoring based on the privilege level they attain on individual machines and their ability to keep services available while under attack from other teams. Unlike conventional KotH designs in which teams start from identical machines [6], Netsiege is designed to couple offense and defense on the same network throughout, and its phased expansion provides trailing teams repeated opportunities to re-enter the competition, directly countering the possible snowball effect.

We piloted Netsiege over seven weeks with a novice-leaning cohort of 12 participants in a student-run cyber club at a large R1 institution, and we report formative data from shell-level command telemetry and participant surveys alongside a candid reflection on which aspects were effective or ineffective, including an overemphasis on offense that left defensive practice underexercised.

This paper makes the following contributions: (1) the design of a long-form, phased, black-box KotH competition with a re-entry mechanism intended to sustain novice engagement; (2) an open-source, privilege-tiered scoring toolchain (Scorekeeper, Tally, and Perspective) and a reproducible provisioning-and-validation pipeline; (3) a reusable educational roadmap that maps instruction to competition phases under a club-mentor delivery model; and (4) formative observations from a seven-week pilot, framed throughout for adoption and extension by other resource-constrained clubs and their faculty advisors. To our knowledge, no prior competition combines black-box initial access, multi-week distributed practice, and a club-mentor delivery model. We release tooling and example scenarios as open source to support cross-club replication [1].

2 Related Work

We situate Netsiege at the intersection of three lines of work: (1) cybersecurity competitions and their accessibility, (2) the KotH format, and (3) club-led security education.

Cybersecurity competitions and accessibility. Balon & Baggili's 2023 survey of the cyber competition landscape catalogs 9 competition types, including KotH-style events, alongside numerous tools (e.g., for challenge creation and configuration management). They note that competitions cultivate student interest in the field but identify a high knowledge barrier, a lack of diversity in competition types (CTFs dominating), and a need for competition hosts to share resources and publish works on their events [4]. Vigna et al. discuss the difficulty of running attack-defense CTFs in their iCTF framework [28], while Pusey et al. observe that competition training occurs through extracurriculars (e.g., clubs) with room to support self-efficacy in informal settings [23].

The KotH format. The KotH model most directly informs our design of Netsiege. Bock et al. [6] introduced an in-class KotH competition for teaching penetration testing in which teams receive an isolated clone of the network weeks in advance, allowing participants to identify vulnerabilities and write *implants* (i.e., automation that supports persistence on a compromised machine) before competing in a single live session of a few hours. We adapt the KotH model because it supports both attack and defense, allowing adopters to emphasize either side to match their desired specialty (e.g., a CCDC-focused club can prioritize defense). In comparison to Bock et al.'s focus on advanced preparation under white-box conditions, Netsiege differentiates from this model by being designed to have teams begin with no access and break in under black-box conditions (i.e., the competition environment is introduced only at the event start), to be long-form and run across weeks, and to assume the event is facilitated by students with no faculty instructor.

Club-led security education. Recent studies stress the need for more experiential learning opportunities [5, 9, 17], reinforced by the hands-on skills gap between new graduates and employer expectations [2, 3]. Student-run cyber clubs have emerged as facilitators of experiential, peer-led learning that contributes to addressing the skills gap [18, 20, 27, 30]. Straub claims that peer-learning activities engage students in ways that instructional staff cannot and offer a path to scaling hands-on instruction when qualified personnel may be scarce [27]. From a survey of 7 cybersecurity clubs across high schools and universities, Yerby & Coburn discuss how clubs operate with minimal budgets and formal support [30]. Netsiege is designed and implemented to support resource-constrained clubs and provide avenues for clubs to collaborate with their university and faculty in offering hands-on security education opportunities.

3 Netsiege Design

3.1 Competition Model and Lifecycle

An Netsiege competition runs on a shared network of virtual servers hosting varied services (e.g., databases, RESTful APIs, admin consoles) woven into a coherent scenario. Each machine exposes multiple vulnerabilities that teams are incentivized to find, exploit, and escalate in order to obtain superuser (i.e., root) access. Control is contested rather than assigned: A team that gains access must remediate the same attack vectors they used, denying them to others and preserving their scoring position. The competition lifecycle exercises offensive and defensive activities on the same network throughout the event.

Two properties distinguish this model from conventional KotH competitions and attack-defense CTFs, in which teams begin with identical machines and defend from a white-box perspective [6, 11]. First, teams start with no control and must enumerate and break in, emulating the black-box conditions of a real-world penetration-testing engagement rather than probing their own machines to discover shared weaknesses. Second, the competition is persistent and long-form, running across weeks of dedicated club meetings rather than a single timed session.

To support transfer to real-world practice, each machine is provisioned with a specific operational role (e.g., database server, web application host, authentication), and its vulnerabilities are derived from that role (e.g., weak password access into a database). This design keeps the environment from reading as disconnected CTF challenges and prompts participants to hypothesize plausible attack vectors (i.e., threat model) within a realistic threat landscape [19].

3.2 Phased Progression and Re-entry

Netsiege is structured into phases that grow more difficult as participants' skills develop, primarily by adding new machines. Every machine introduced in a new phase is integrated into the prior phase's environment through interdependent lateral-movement vectors, rewarding teams and participants who conduct thorough network enumeration. An asset that seems irrelevant early (e.g., leaked credentials for a non-existent account) may become a critical pivot once later machines are introduced, reinforcing methodical practice over opportunistic vulnerability hunting.

We propose four per-machine design requirements to keep environments consistent across scenarios and facilitators: (1) **Initial-vector redundancy:** Every machine provides at least two distinct paths to initial access, and at least one must be reachable without exploit chaining or lateral movement, so a foothold is always directly attainable. (2) **Privilege-escalation redundancy:** Every machine provides at least two distinct vectors for vertical privilege escalation from an unprivileged user to root. (3) **Lateral movement:** Every machine has at least one outgoing lateral-movement mechanism, encouraging post-exploitation enumeration rather than treating hosts as isolated. (4) **Contextualization:** Every machine fulfills an operational role from which all its vulnerabilities plausibly derive.

Phased progression also addresses the snowball effect (i.e., an early lead compounding into an advantage that trailing teams cannot overcome). Netsiege counters this effect by expanding the environment in each phase, introducing a new segment that gives disadvantaged teams a partial reset and a renewed opportunity to compete. While existing lateral-movement vectors do confer an advantage on leading teams, the requirement that every server retain an independent initial-access vector (i.e., one not contingent on lateral movement) ensures that trailing teams can always regain a foothold. We refer to this property as *re-entry*.

3.3 Scoring Model

Scoring relies on three attributes: the access levels a team achieves (i.e., regular user versus superuser), a team's ability to keep the host's services functional while under attack, and a team's continued defense of the control on machines. Each machine is evaluated through two complementary kinds of scoring checks.

Control checks. A control check determines which team, if any, holds a given privilege level on a server. Every machine has at least one control check, each of which verifies the successful execution of a single exploit or a chain of exploits. Organizers may choose the granularity. For instance, a check may validate a single application-layer vulnerability (e.g., arbitrary file write) or require a complete multi-stage chain of exploits (e.g., converting an arbitrary file write into SSH access via a backdoor user) before awarding points. Control is claimed through a pre-placed `control.txt` file on the host, polled periodically by a scoring agent. A team writes its unique token into that file to be credited with control.

Service-uptime checks. A service-uptime check verifies that a team's remediation or patching has not broken the machine's core functions. An external service exercises each monitored service via protocol- or application-specific interactions (e.g., authenticating via SSH, uploading documents to a file share). Uptime is a prerequisite for scoring rather than a source of points; if a machine's services are found to be inoperative, no team controlling that machine accrues points for a given round's checks.

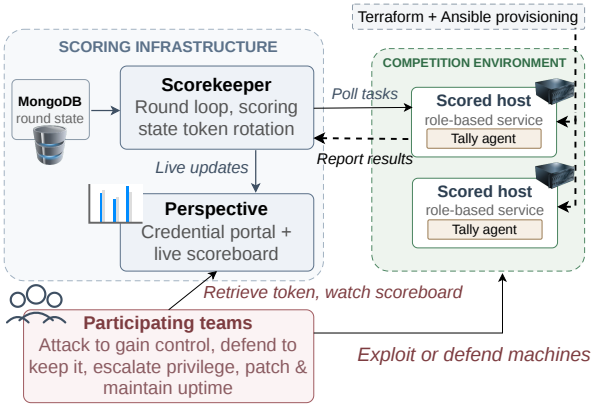

Figure 2: The Netsiege toolchain and its interaction with the competition environment and teams.

Incentive structure. Control checks are weighted in proportion to the privilege level attained, meaning that superuser access is worth more than an unprivileged foothold. A failed service-uptime check nullifies the control points for the affected machine; thus, teams are simultaneously pushed to escalate to the highest privilege available and to patch the vulnerabilities they exploited without disrupting the services that depend on them. This design mirrors the real availability/security trade-off while discouraging destructive "scorched-earth" tactics (e.g., removing vulnerable functionality that users need) that a purely offense-driven scoring scheme would reward. Whether these incentives suffice to elicit balanced defensive practice is a question we examine in our pilot (Section 6.2).

4 Implementation and Tooling

A central design goal of Netsiege is adoptability by resource-constrained student organizations. Rather than adapt a scoring engine built for other goals and audiences, we developed a lightweight, declaratively configured toolchain that a club can deploy, inspect, and extend with modest infrastructure. This section describes the scoring architecture, the deployment and validation pipeline, and the telemetry used to study participant behavior.

4.1 Scoring Architecture

The scoring system comprises three connected applications: *Scorekeeper*, a central orchestration engine; *Tally*, a per-host scoring agent; and *Perspective*, a public-facing interface.

Scorekeeper is a Python service that oversees all scoring state and drives the competition's round loop. On each round, Scorekeeper runs every host's checks, updates team scores, rotates per-team tokens, persists round state to a MongoDB database, and pushes results to the front end. Round timing is randomized (a configurable interval plus jitter), so participants cannot predict when checks will occur and stage their defenses accordingly. A competition is described declaratively: Organizers advance the competition or develop a new scenario by editing per-phase configuration files rather than code, lowering the barrier for adopters to build and run their own events.

Table 1: Existing scoring engines against Netsiege's requirements.

Engine	Style	Uptime	Tiered Control
CyberSeer [16]	general skills	–	–
Quotient [10]	defense-only	✓	–
RootTheBox [15]	CTF	–	partial
ScoreStack [25]	defense-only	✓	partial
Scorebot [6]	attack-defense	✓	–
ScoringEngine [26]	attack-defense	✓	–
Netsiege	attack-defense	✓	✓

Tally is a lightweight agent that runs on every scored machine, reporting check results back to Scorekeeper. Its capabilities are deliberately minimal, so that a team compromising a scored host gains no administrative channel into other machines through the agent itself.

Perspective is the participant- and spectator-facing interface, serving as both a credential portal and a real-time scoreboard broken down by the privilege tier at which each machine is held. Surfacing the privilege tier rather than a binary "owned" state lets participants see partial progress at a glance.

4.2 Rationale for a Custom Scoring Engine

We surveyed existing open-source scoring engines before building our own and found that, while several met individual requirements, none met all our requirements at the needed granularity (Table 1). In particular, scoring which privilege level a team holds, rather than a binary 'owned/not-owned' state, was absent from every engine surveyed. RootTheBox and ScoreStack can model ownership [15, 25], but not at the per-privilege granularity central to our partial-credit design; others focus on service-uptime scoring [10, 25] or CTF-style flag submissions [15]. Building a purpose-made scoring engine also lets us prioritize operational simplicity for organizers and understandability for novice participants.

4.3 Deployment, Validation, and Telemetry

Redeploying an environment is a recurring organizer burden; thus, we treat the environment as versioned, reproducible infrastructure. A scenario is a self-contained bundle of (1) per-host provisioning scripts, (2) the vulnerable applications defining each host's role, (3) machine-readable documentation of every host's intended attack vectors, and (4) the scoring configuration. Deployment proceeds in two stages: infrastructure-as-code (Terraform) provisions the virtual machines, and configuration management (Ansible) installs the Tally agent, creates the scoring account, and collects logs from every machine with a single command.

To study participant behavior with minimal interference, we instrumented each participant's provisioned workstation with a lightweight shell hook that appended every command, alongside a corresponding timestamp, to a local CSV log. Sitting at the shell level, the hook captures all terminal activity across tools without per-tool integration and with negligible overhead. After the event, the same Ansible-based log-collection step used for machine logs retrieved these workstation logs into a single dataset.

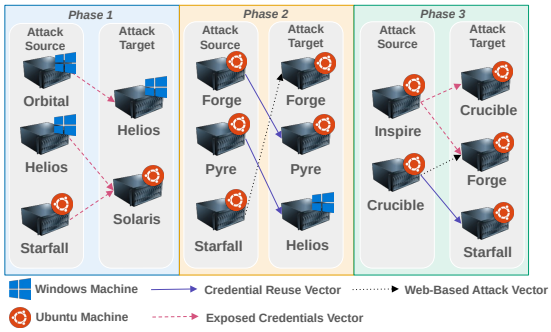

Figure 3: Lateral-movement dependencies across Sunspire's three phases. In each phase, arrows connect attack-source hosts (left) to attack-target hosts (right) by vector type.

5 Deployment: The Sunspire Pilot

5.1 Example Scenario: Sunspire

To make Netsiege concrete and show the Section 3.2 requirements are satisfiable in practice, we describe *Sunspire*, the scenario we built for our pilot. Sunspire is framed as the municipal computing network of a fictional futuristic city, where each machine corresponds to a civic system (e.g., scheduling, thermal monitoring, weather forecasting). The lightweight narrative supports our role-based design by assigning a plausible role to every machine.

The Sunspire network comprises 9 machines spanning Ubuntu and Windows servers, introduced across three phases: Phase 1 starts with four machines, and Phases 2 and 3 add three and two machines, respectively. Figure 3 shows the full topology and its inter-machine dependencies: Every machine provides at least two privilege-escalation paths; an edge (m_{src}, m_{tgt}) in such a path, as depicted in Figure 3, represents m_{src} as the machine from which an attack can be successfully launched and m_{tgt} as the machine that would be controlled if that attack succeeds. This design satisfies our redundancy requirements and guarantees that trailing teams can always gain re-entry.

5.2 Example Curriculum

Netsiege's design targets clubs without an instructor of record; thus, we paired Sunspire with a practitioner-oriented curriculum that club mentors can deliver. The curriculum distills techniques from Netsiege's developers (i.e., the authors) who have accumulated over a combined decade of cyber-competition experience and is organized so that instruction precedes the capability each phase unlocks. Table 2 maps each session to its competition phase: Each session pairs a short mentor-led lecture with open competition time—time during the same meeting in which participants play in the live environment and apply the technique(s) just taught. We release the full Sunspire scenario alongside this curriculum and its session timeline so that adopting clubs can run Sunspire directly or adapt its mapping to their own competition specialty (e.g., a club preparing for CPTC can focus on offensive security sessions) [1].

5.3 Pilot Context and Cohort

We piloted Netsiege with a student-run cyber club of roughly 45-50 active members at a large U.S. R1 university (30,000+ students) that needed hands-on training for newer members interested in

Table 2: The Sunspire example curriculum, mapping mentor-led instructional sessions to competition phases. Each session pairs a short lecture with open competition time.

Phase	Session	Content
1	Intro to Netsiege	Competition scenario, structure, scoring, access
	OS fundamentals	Linux/Windows CLI, SSH, RDP
	Network enumeration	`nmap`, `enum4linux`, password reuse
	File shares	SMB, FTP
2	CVEs	Using and adapting CVE proof-of-concepts
	Basic privilege esc.	SUID, writable files, scheduled jobs, leaked credentials/keys
	Web exploitation I	Command injection, LFI, file upload
	Defense	Log review, credential rotation, configuration hardening
3	Review and compete	No new technique
	Web exploitation II	SQLi, log poisoning, SSTI
	Wrap-up & open session	Exploit paths released

joining their competition teams. Instruction was delivered by 5 club members with at least 1 year of experience on a collegiate cyber competition team (e.g., CCDC, CPTC), serving as mentors rather than as an instructor of record, reflecting the club-led model the design targets. Our institution's Institutional Review Board (IRB) reviewed our study protocol and granted us exempt self-determination status.

The pilot ran over seven weeks of a 10-week quarter, from early January to mid-February 2026, structured as a sequence of dedicated club meetings twice a week. We had three meetings canceled due to holidays and conflicting club events, leaving 11 two-hour meetings. From a pool of 27 interested club members, 12 participants were randomly selected and organized into three teams of four; team size ensured enough targets per team, given the 9-machine network. The cohort was novice-leaning and drawn primarily from undergraduate computing-adjacent majors, with Computer Science most represented. One was enrolled in a non-computing major (Social Ecology), suggesting the format may have lowered the barrier to engage a student outside the traditional CS pipeline. Table 3 summarizes the cohort.

Each participant worked from a provisioned Kali Linux workstation, which let us avoid per-participant setup and gave us a uniform platform for telemetry collection. We administered a free-response entry assessment of 17 questions covering expected network and curriculum content (e.g., command-line knowledge, common vulnerability classes), scored out of a possible 17 points (one point per question). Participant scores ranged from 1 to 7 ($mean$ = 3.56, $median$ = 3.5), confirming that most participants began with limited security knowledge. Teams were balanced by entry-assessment score and self-reported time commitment so that no team held a disproportionate share of experienced members.

Table 3: Pilot cohort demographics (n = 12), by declared major and academic year.

	Male	Female	Total
Major			
Computer Science	5	2	7
Software Engineering	2	0	2
Computer Engineering	1	1	2
Social Ecology	1	0	1
Academic year			
First-year	4	1	5
Second-year	1	0	1
Third-year	4	1	5
Fourth-year	0	1	1
Total	9	3	12

5.4 Data Sources

We collected two categories of formative data over the seven-week pilot: Shell-level command telemetry from participant workstations and three self-report surveys. Each data source is intended to characterize participation and surface concrete lessons for future iterations and adopters. **Command telemetry:** As described in Section 4.3, each Kali workstation logged every shell command with a timestamp to a local CSV, retrieved via Ansible at the end of the pilot. **Surveys:** We administered three surveys on a 1-5 agreement scale (1 = strongly disagree, 5 = strongly agree). Two brief mid-competition surveys, released after the first and second instructional blocks, asked whether each lecture's content was reflected in the environment and whether participants had adequate time to apply it. The concluding survey covered self-perceived gains in foundational, offensive, and defensive topics; career interests; and the educational value of the lecture, environment, and mentoring components. Each survey also included free-response prompts on coverage gaps, AI tool usage, and suggested improvements. The two mid-competition surveys received 9 and 5 responses, respectively, and the concluding survey received 7 responses; response counts declined alongside meeting attendance over the quarter.

6 Discussion

We piloted AnonComp once, with a single novice-leaning cohort, so we frame this section as a reflection for practitioners rather than as evidence of a generalizable effect. Our formative data (telemetry and three self-report surveys) characterize participation and surface lessons; survey responses are voluntary and self-reported, and telemetry captures only terminal activity, missing browser-driven exploitation and GUI tools. We report on which aspects the pilot suggests were effective, which were not, and why.

6.1 What Proved Effective

Long-form participation created time to learn inside the environment. Participants consistently agreed they had adequate time to apply what they learned (mid-competition survey means of 4.22 and 4.60 on a 1-5 scale, concluding survey mean of 4.29). Telemetry shows self-directed adoption of untaught tools, with participants looking up unfamiliar utilities and using them correctly.

The club-mentor model was viable without an instructor of record. Instruction was delivered by peer mentors, and among

the participants who completed the concluding survey ($n = 7$), the mentoring component was rated highest of all items (mean of 4.86, tied with overall enjoyment); we acknowledge this reflects the subset who remained engaged through the end of the pilot. This aspect is encouraging for the resource-constrained clubs our design targets, as it shows that the most-valued instructional element from our pilot is one provided by club members.

Phased expansion and re-entry drew trailing teams back in. Telemetry shows a trailing team re-establishing scoreboard presence in a later phase after being locked out earlier—recovery that our re-entry property is meant to enable. The redundancy requirement was also exercised, as on one machine, two teams pursued distinct documented vectors within the same session. The environment functioned as a coherent, contextualized network.

6.2 What Proved Ineffective

Offense crowded out defense. The pilot's central shortcoming is an overemphasis on offensive practice. Self-reported gains on the concluding survey were high for offensive and foundational topics but markedly lower and more dispersed for defensive topics. The scoring model does reward uptime-gated control, but in practice, the race to first compromise dominated the experience, and defensive activity (e.g., patching, credential rotation, log review) was underutilized. Our design intent of coupled offense and defense was only partially realized.

First-control advantage undermined re-entry in practice. Although the independent initial-vector requirement guarantees a path back into any machine, participants reported re-entry was hard in practice. Once a team compromised a machine and changed its credentials, others perceived the machine as effectively locked. The independent initial-vector requirement only ensures such a vector exists; it does not make that vector discoverable or usable by a novice, which is a gap between the design's property and the lived experience that adopters should anticipate.

Attendance decayed over the quarter. Participation fell from full attendance in week one to an average of 6 out of the original 12 attendees (50%) in the last four meetings, amid midterms and competing obligations. Long-form delivery allows for time for distributed practice but exposes the intervention to the ordinary attrition of a multi-week extracurricular [8].

6.3 Recommendations for Adopters

The pilot surfaced concrete changes that would improve Netsiege before another club or educator adopts it. We frame these as guidance for practitioners, tied to the shortcomings of Section 6.2.

Force defensive practice structurally, not just through incentives. Making control points conditional on service uptime (Section 3.3) did not, on its own, produce balanced defensive activity. Adopters should consider a dedicated defensive phase in which control is frozen, and points accrue only for successful hardening and service maintenance. Another option is a scoring multiplier that grows the longer a team controls a machine without losing it, rewarding defense directly rather than as a side effect of offense.

Soften first-control lockout to protect novice re-entry. In our pilot, the first team to compromise a machine typically changed its passwords and keys, disabling the credential-based entry paths that trailing teams had been taught to use and leaving those machines feeling permanently "locked" (Section 6.2). Adopters can keep a path open in two ways. First, an organizer can reset a machine's credentials to a known state at each phase boundary, restoring the taught entry path for the next phase. Second, each machine can expose at least one entry vector that does not depend on a changeable credential (e.g., an unauthenticated service vulnerability), so patching credentials cannot fully seal it. Both measures keep a beginner path open, making re-entry more possible for novices.

Provide a persistent, written reference. The most common request from the free-responses was for article-style lecture write-ups with worked examples that participants could revisit asynchronously, since not everyone attended every meeting.

Plan around attrition. Adopters should front-load foundational content before expected midterm seasons, keep later sessions self-contained, and treat declining attendance as expected.

Minimal setup. Running Netsiege requires a small set of virtual machines (e.g., nine used for our three-phase pilot), one Scorekeeper host with a Tally agent on each scored machine, and a few mentors with prior competition experience (e.g., more senior club members). The declarative scenario and the infrastructure-as-code pipeline let a club redeploy an environment without years of competitive experience or infrastructure expertise. The highest recurring cost is developing and maintaining vulnerable network environments, which demands expertise in building realistic attack surfaces; adopters without that expertise can run Sunspire directly rather than creating a new scenario.

7 Conclusion

Hands-on cybersecurity training for students is often delivered through short-form competitions that favor experienced teams and leave novices little room to learn, a gap most pronounced in resource-constrained cyber clubs. We presented Netsiege, a long-form, club-led KotH competition in which teams break into a role-based enterprise network under black-box conditions, defend what they take, and re-enter as the environment expands across phases. We contribute a four-pronged adoptable model: (1) a phased competition design, (2) an open-source privilege-tiered scoring toolchain, (3) a reproducible provisioning-and-validation pipeline, and (4) a curriculum mapping instruction to phases under a club-mentor delivery model. Our seven-week pilot with a novice-leaning cohort suggests that the format provides genuine time to learn within the environment and that peer mentors can provide instruction without a course of record, while also surfacing a weakness in the existing design of offense crowding out defense, which we have translated into revisions for adopters. We release the full Sunspire scenario, curriculum, and tooling to support cross-club replication [1].

Acknowledgments

Special thanks to Jacob Lee, Caitlyn Oda, and Ashton Yoshino for their contributions to Netsiege during its development, and to Sterling Gip, Alex Loan, Drew Levy, and Dhruv Kandula for serving as student mentors during the competition pilot. We greatly appreciate the feedback from the reviewers of this paper.

References

[1] 2026. Anonymized Artifact for AnonComp: A Long-Form, Club-Led King-of-the-Hill Competition for Accessible Cybersecurity Training. https://zenodo.org/

records/21188261.
[2] Sam Attwood and Ashley Williams. 2023. Exploring the UK Cyber Skills Gap through a mapping of active job listings to the Cyber Security Body of Knowledge (CyBOK). In *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering* (Oulu, Finland) *(EASE '23)*. Association for Computing Machinery, New York, NY, USA, 273–278. doi:10.1145/3593434.3593459
[3] Louise Axon, Katherine Fletcher, Arianna Schuler Scott, Marcel Stolz, Robert Hannigan, Ali El Kaafarani, Michael Goldsmith, and Sadie Creese. 2022. Emerging Cybersecurity Capability Gaps in the Industrial Internet of Things: Overview and Research Agenda. *Digital Threats* 3, 4, Article 34 (Dec. 2022), 27 pages. doi:10.1145/3503920
[4] Tyler Balon and Ibrahim (Abe) Baggili. 2023. Cybercompetitions: A survey of competitions, tools, and systems to support cybersecurity education. *Education and Information Technologies* 28, 9 (Feb. 2023), 11759–11791. doi:10.1007/s10639-022-11451-4
[5] Brendan Bertone, Paul Wagner, and Joshua Pauli. 2025. Experiential Learning: Innovative Approaches to Post-Secondary Cybersecurity Education. *Journal of Cybersecurity Education, Research and Practice* 2025 (09 2025). doi:10.62915/2472-2707.1254
[6] Kevin Bock, George Hughey, and Dave Levin. 2018. King of the Hill: A Novel Cybersecurity Competition for Teaching Penetration Testing. In *2018 USENIX Workshop on Advances in Security Education (ASE 18)*. USENIX Association, Baltimore, MD. https://www.usenix.org/conference/ase18/presentation/bock
[7] Nicholas J. Cepeda, Harold Pashler, Edward Vul, John T. Wixted, and Doug Rohrer. 2006. Distributed practice in verbal recall tasks: A review and quantitative synthesis. *Psychological Bulletin* 132, 3 (2006), 354–380. doi:10.1037/0033-2909.132.3.354
[8] Ronald S Cheung, Joseph Paul Cohen, Henry Z Lo, Fabio Elia, and Veronica Carrillo-Marquez. 2012. Effectiveness of cybersecurity competitions. In *Proceedings of the International Conference on Security and Management (SAM)*. 1.
[9] James Crabb, Christopher Hundhausen, and Assefaw Gebremedhin. 2024. A Critical Review of Cybersecurity Education in the United States. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (Portland, OR, USA) *(SIGCSE 2024)*. Association for Computing Machinery, New York, NY, USA, 241–247. doi:10.1145/3626252.3630757
[10] dbaseqp. 2026. Quotient. https://github.com/dbaseqp/Quotient.
[11] DEF CON. 2026. CTF Archive. https://defcon.org/html/links/dc-ctf.html.
[12] Center for Infrastructure Assurance and Security. 2025. National Collegiate Cyber Defense Competition. https://www.nationalccdc.org/.
[13] HITCON CTF Team 2025. 2025. HITCON CTF 2025. https://ctf2025.hitcon.org/dashboard/.
[14] James Mattei, Christopher Pellegrini, Matthew Soto, Marina Sanusi Bohuk, and Daniel Votipka. 2025. "I'm trying to learn... and I'm shooting myself in the foot": beginners' struggles when solving binary exploitation exercises. In *Proceedings of the 34th USENIX Conference on Security Symposium* (Seattle, WA, USA) *(SEC '25)*. USENIX Association, USA, Article 148, 20 pages.
[15] moloch--. 2026. RootTheBox. https://github.com/moloch--/RootTheBox.
[16] Christopher Morales-Gonzalez, Matthew Harper, Pranathi Rayavaram, Sashank Narain, and Xinwen Fu. 2025. Enhancing Cybersecurity Education using Scoring Engines: A Practical Approach to Hands-On Learning and Feedback *(SIGCSETS 2025)*. Association for Computing Machinery, New York, NY, USA, 784–790. doi:10.1145/3641554.3701905
[17] Sashank Narain, Pranathi Rayavaram, Christopher Morales-Gonzalez, Matthew Harper, Maryam Abbasalizadeh, Krishnaa Vellamchety, and Xinwen Fu. 2025. Practical Cybersecurity Education: A Course Model Using Experiential Learning Theory. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1* (Pittsburgh, PA, USA) *(SIGCSETS 2025)*. Association for Computing Machinery, New York, NY, USA, 819–825. doi:10.1145/3641554.3701922
[18] Brooke Nelson, Amanpreet Kapoor, and Christina Gardner-McCune. 2024. Understanding Undergraduate Students' Participation in Computing Clubs. In *ACM Technical Symposium on Computer Science Education (SIGCSE) — ACM Student Research Competition*. Portland, OR. https://sigcse2024.sigcse.org/details/sigcse-ts-2024-acm-student-research-competition/2/Understanding-Undergraduate-Students-Participation-in-Computing-Clubs
[19] TJ OConnor. 2022. HELO DarkSide: Breaking Free From Katas and Embracing the Adversarial Mindset in Cybersecurity Education. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1* (Providence, RI, USA) *(SIGCSE 2022)*. Association for Computing Machinery, New York, NY, USA, 710–716. doi:10.1145/3478431.3499404
[20] Matt Piazza and Aspen Olmsted. 2015. Founding a cybersecurity club in a higher education environment: A case study. In *2015 World Congress on Internet Security (WorldCIS)*. 139–143.
[21] Plaid Parliament of Pwning. 2025. Plaid CTF 2025. https://plaidctf.com/.
[22] Portia Pusey, Sr. David Tobey, and Ralph Soule. 2014. 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)*. USENIX Association, San Diego, CA. https://www.usenix.org/conference/3gse14/summit-program/presentation/pusey
[23] Portia Pusey, Mark Gondree, and Zachary Peterson. 2016. The Outcomes of Cybersecurity Competitions and Implications for Underrepresented Populations. *IEEE Security & Privacy* 14, 6 (2016), 90–95. doi:10.1109/MSP.2016.119
[24] RIT ESL Global Cybersecurity Institute. 2025. Collegiate Penetration Testing Competition - Education Through Competition. https://cp.tc/.
[25] scorestack. 2025. Scorestack. https://github.com/scorestack/scorestack.
[26] scoringengine. 2026. Scoring Engine. https://github.com/scoringengine/scoringengine.
[27] Jeremy Straub. 2019. Assessment of the Educational Benefits Produced by Peer Learning Activities in Cybersecurity. In *2019 ASEE Annual Conference & Exposition*. ASEE Conferences, Tampa, Florida. https://peer.asee.org/32131.
[28] Giovanni Vigna, Kevin Borgolte, Jacopo Corbetta, Adam Doupé, Yanick Fratantonio, Luca Invernizzi, Dhilung Kirat, and Yan Shoshitaishvili. 2014. Ten Years of iCTF: The Good, The Bad, and The Ugly. In *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*. USENIX Association, San Diego, CA. https://www.usenix.org/conference/3gse14/summit-program/presentation/vigna
[29] Jan Vykopal, Valdemar Švábenský, Michael Tuscano Lopez II, and Pavel Čeleda. 2025. Cybersecurity Study Programs: What's in a Name?. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1* (Pittsburgh, PA, USA) *(SIGCSETS 2025)*. Association for Computing Machinery, New York, NY, USA, 1169–1175. doi:10.1145/3641554.3701976
[30] Johnathan Yerby and Matthew Coburn. 2025. Cybersecurity Club and Team Practices: A Qualitative Multisite Study of Engagement and Success Strategies. In *2025 Cyber Awareness and Research Symposium (CARS)*. 1–6. doi:10.1109/CARS67163.2025.11337861