King of the Hill: A Novel Cybersecurity Competition for Teaching Penetration Testing

Kevin Bock George Hughey Dave Levin
University of Maryland

Abstract

Cybersecurity competitions are an effective and engaging way to provide students with hands-on experience with real-world security practices. Unfortunately, existing competitions are ill-suited in giving students experience in penetration testing, because they tend to lack three key aspects: (1) pivoting across multiple machines, (2) developing or implanting custom software, and (3) giving students enough time to prepare for a lively in-class competition. In this paper, we present the design, implementation, and an initial run of *King of the Hill (KotH)*, an active learning cybersecurity competition designed to give students experience performing and defending against penetration testing. KotH competitions involve a sophisticated network topology that students must pivot through in order to reach high-value targets. When teams take control of a machine, they also take on the responsibility of running its critical services and defending it against other teams. Our preliminary results indicate that KotH gives students valuable and effective first-hand experience with problems that professional penetration testers and network administrators face in real environments.

1 Introduction

Performing and defending against penetration testing (pentesting) are important and exciting topics for students to learn. Through teaching these concepts, we find that students learn fundamental aspects of security, such as maintaining critical services, identifying weak links in a network, and composing multiple, smaller attacks to achieve a larger, more difficult goal.

Cybersecurity competitions—such as Capture-the-Flag (CTF) [1, 2, 7, 6] and Build-it/Break-it/Fix-it [18, 19]—have been demonstrated to have positive impact on cybersecurity education [4, 8, 12]. However, we have found prior designs to be a poor fit for pentesting. In particular, we identify three key aspects to pentesting that, to our knowledge, have not been widely reflected in prior cybersecurity competitions (we describe related work more fully in Section 2):

- **Pivoting**: In real-world networks, rarely are all machines (and all vulnerabilities) directly accessible from all other machines. Real pentesters must *pivot* from one compromised machine to another in order to gain access to more machines.
- **Implants**: After gaining access to a machine, pentesters often have the opportunity to leave behind *implants*: their own software that, for instance, opens a backdoor, closes vulnerabilities from other attackers, and so on.
- **Preparation**: Professional pentesters often have time to prepare by performing reconnaissance, scanning a network, and writing their implant code.

We introduce *King of the Hill (KotH)*, an in-class cybersecurity competition designed to give students hands-on experience performing—and defending against—penetration testing. KotH is distinguished from prior competitions in that it supports the aspects we believe to be key to teaching pentesting. *Pivot*: Whereas most competitions like Attack-Defense CTFs tend to use a fully connected mesh network, KotH competitions involve a more sophisticated network topology. Students must therefore employ a strategy of what *sequence* of machines to compromise and defend. *Implant*: Once one team takes over a machine, another team of students may subsequently take it over. Students therefore have incentive to leave *implants* that patch running services and that introduce their own backdoors. *Prepare*: KotH competitions are held during class; in order to prepare their strategies and implants, we give students copies of the network ahead of time.

As with all offensive competitions, care must be taken to ensure this offensive exercise does not end up teaching students the "wrong" behavior. To this end, KotH is both offensive and defensive: when a team takes over a machine, they do not simply collect a flag, but rather they take over the *responsibility* of that machine. In particular, every machine in a KotH network runs one or more critical services; when a team has control over a machine, the team gains points for as long as the critical service remains running. Students therefore gain experience having to decide between turning off a service that is known to be vulnerable (and thus not gain as many points), running the vulnerable service (and thus risk losing control of the machine altogether), or patching the vulnerability (and spending valuable time in the process). In this way, we believe KotH exposes students to many of the trade-offs inherent in network administration, and teams are rewarded if they can defend a machine successfully. Furthermore, exploiting these vulnerabilities actually gives students more experience and a deeper knowledge of how the vulnerabilities occurred, so that they can protect themselves in the future. We aim to define a game that reinforces best practices in security, and punishes objectively incorrect behavior (destroying computers, etc.).

This paper makes the following contributions:

- We present the design and implementation of KotH, a security competition for teaching students how to launch and defend against penetration testing.
- We present preliminary results from an in-class run of KotH, showing that KotH is an effective way to teach students penetration testing skills.
- Our run of KotH's network layout and scorebot source code are publicly available at:

  `https://koth.cs.umd.edu`

The rest of this paper is organized as follows. In Section 2, we review related work in the areas of cybersecurity competitions. We describe KotH's design goals in Section 3, its design in Section 4, and our implementation in Section 5. In Section 6, we present the results from a sample run of KotH. We discuss potential variations of KotH in Section 7, and conclude in Section 8.

2 Related Work

As cybersecurity competitions have become remarkably popular in recent years, a wide range of variants have been introduced. In Capture the Flag (CTF) competitions, participants act as attackers in various challenges, including web exploits, cryptography, and reverse engineering—these give students experience with launching attacks. In Attack/Defense CTFs, participants are provided a network they must solely attack or defend [14, 20]—these were designed to give students experience with both offensive- and defensive-related skills. Build-it/Break-it/Fix-it competitions ask build-it teams to write software, which is subsequently attacked by break-it teams [18, 19]—these competitions give students experience with writing and attacking secure software. Throughout various competitions such as these, teams tend to take a purely offensive or purely defensive stance. In contrast, KotH teams must simultaneously exploit a wide range of vulnerabilities while defending against other attackers seeking to take over their compromised machines. This "tug-of-war" over individual machines leads students to consider trade-offs between both offensive and defensive stances.

Conklin described a cybersecurity curriculum involving penetration testing [3], but chose not to incorporate offensive games into his capstone competition. There is empirical evidence for a shortage of competitions suited to teach penetration testing. Woszczynski and Green [21] surveyed educators, employers, and judges from cyber defense competitions (CDCs) to evaluate the extent to which different competitions prepared students for various skills, including penetration testing, use of Linux, information assurance, and so on. They found that, while universities and employers placed high importance on penetration testing (2.88 and 3.14 on a 4-point scale respectively), existing CDCs and CTFs did a poor job at preparing students for them (1.88 on a 4-point scale). We seek to fill this gap with a competition centered on penetration testing.

3 Goals

In order to teach penetration testing, we wanted an exercise that was engaging and allowed students to practice and apply skills specific to penetration testing. Attack/Defense CTFs lend themselves closely to the offensive nature of such an exercise, and have been used for students to practice both offensive and defensive skills alike. However, there are a number of critical aspects of penetration testing specifically that students do not get to practice in most Attack/Defense CTFs:

Goal #1: Pivoting The first of these is the concept of pivoting. For professional penetration testers, pivoting (sometimes also known as lateral movement) is the process of using a compromised machine to further access and launch attacks deeper into otherwise inaccessible networks or machines. For example, if an attacker has a foothold in a machine in subnet A and can compromise a machine connected to both subnet A and subnet

B, the attacker could use the second machine to pivot and extend their reach into subnet *B*. Pivoting is a critical skill for professional hackers and penetration testers to master, but most Attack-Defense CTFs generally connect all teams to a small set of computers on one fully interconnected or mesh network. In these network configurations, no pivoting is required to attack other computers. Therefore, the first goal in the design of our competition was to require students to pivot, so as to more closely mimic real penetration testing engagements.

Goal #2: Implants Another important aspect to penetration testing engagements is the development of custom scripts, tools, and implants. For professional hackers and penetration testers, an implant is a binary that is deployed on a target network to accomplish some task or offer a persistence mechanism to the attacker. The development of custom implants or tools has become increasingly critical for successful penetration testing engagements. Unfortunately, some competitions directly prohibit the development or inclusion of custom tools or scripts [20]. Therefore, the second goal of our competition was to encourage custom implant development.

Goal #3: Reconnaissance In most CTFs, the competitors are not given access to or even detailed information about the target environment in advance of the competition. Therefore, students are limited to the competition timeframe to discover vulnerabilities, develop and test exploits, write any required tooling or implants, and plan a strategy for usage. This tight timeline often forces students to sacrifice deeper analyses of targets or the development of scripts, tools, or other implants that could be useful during the competition. Therefore, the third goal of our competition was to design the exercise timetable in such a way that permits deeper vulnerability analysis, while still maintaining the excitement and engagement of a live exercise.

Goal #4: Defensive trade-offs By its nature, an exercise to teach penetration testing must be offensive, but we want students to learn critical defensive skills as well. Many decisions faced daily by Network Security Operation Centers daily are important for students to learn, such as weighing the security risk of assets, monitoring the security of a network, and most importantly, being able to defend against the penetration testing techniques they are learning. Therefore, our fourth goal was to encourage defensive operations and force students to make defensive trade-offs: weighing the relative risk of keeping an important (but potentially vulnerable) asset on a network, watching their network for potential insider threats, and more.

Goal #5: Encourage Good Habits Lastly, it is important to ensure that students do not take away the "wrong" lessons from this exercise. Although the competition is offensive in nature, a critical goal was that students take away best practices for both offense and defense, and keep ethics in mind.

4 Design

To fulfill these goals, we introduce King of the Hill (KotH). KotH is a novel active learning cybersecurity competition designed to let students practice skills specific to penetration testing in a safe, isolated environment. KotH requires students to pivot extensively, develop custom implants, perform advanced reconnaissance before the competition, make live defensive tradeoffs, and addresses other shortcomings. We begin by describing the high-level gameplay.

4.1 Gameplay

Students are divided into teams, and each team must work together to attack, control, and defend as many computers on a target network as they can.

Gameplay occurs over a large, complex, isolated virtual environment, comprised of vulnerable Linux and Windows virtual machines of various builds that are spread across multiple partially interconnected subnetworks. Each team has an entry point into the network at points distant to one another: an offensive Kali image preloaded with many offensive tools that students SSH into to access the competition environment. By design, no other team can access a teams entry machine to ensure that every team can always access the competition environment, and Kali machines are out of scope for the duration of the competition. We also introduce a network-wide *scorebot* which tracks scores throughout the competition. Each machine (other than the entry points and the scorebot) has a number of critical services that must be maintained and protected, and has a number of pre-seeded vulnerabilities. At the beginning of the competition, every vulnerable machine in the network is considered unclaimed.

Each team's goal is to exploit as many vulnerable machines on the network as they can, claim them by calling out to the global scorebot, defend them from other teams, and protect their critical services. We will refer to the collection of machines that have already been claimed by a team as that team's *territory*. Teams are awarded points for each machine they control and, if critical services are functioning on each machine, for each scoring interval throughout the competition. The winner is the team with the most points at the end of the competition.

4.2 Non-trivial network topologies

To achieve our first goal of encouraging and requiring pivoting, in King of the Hill, we design a non-trivial network topology with multiple distinct local area networks, requiring students to pivot repeatedly in order to reach many of the machines in the virtual environment. As in real networks, controlling machines attached to multiple networks is inherently more valuable, as they can be used as a pivot to breach other teams' territories. Students therefore have incentive to control and defend machines attached to multiple networks these machines. KotH's heavy encouragement of pivoting is a significant difference from previous competitions, in which all teams can directly attack all others at any time. Conversely, based on a sample run of KotH (Section 6), we observe that the competing incentives to control well-connected machines lead to interesting and dynamic battlegrounds.

King of the Hill also opens up a number of new teaching opportunities for educators. Instructors can design the environment in order to encourage certain behavior or force students to use specific knowledge, or tailor the environment to their students skillsets and levels. We discuss this in greater depth in Section 7.

4.3 Defense

When gameplay begins, teams initiate offensive operations on the machines in the subnet they can access, and attempt to expand their territory of control. As students break into and claim these machines, they inherit the responsibility for running the computer, and must ensure that the critical services on the machines stay running and secure from other teams. Students must therefore protect both their services and their access to the machines. Competitors are incentivized to patch the vulnerability they used to gain access and any other vulnerabilities they may be aware of to prevent other teams from taking the machine from them. This gives students valuable defensive practice, as they must be able to both exploit a vulnerability and defend against it to succeed.

Through this process of exploitation, patching, and re-exploitation, we expect the machines in the network to get increasingly secure as the competition goes on. Many machines therefore are likely to change hands between teams a number of times until finally they become too secure to be taken over again.

Importantly, many of the services for each computer may contain vulnerabilities. This forces students to make difficult decisions about which services are worth keeping alive—each additional service is worth many points, but could potentially be an avenue for other teams to compromise their machine. Such cost-benefit analyses are common in the real world.

4.4 Student preparation

Introducing a larger, more complicated, diverse network topology, however, exacerbates the issue of limited competition time to fully investigate and explore the environment. Therefore, to our third goal, each KotH team is given a full clone of the environment two weeks in advance of the live competition to allow them to penetration test the network. Every teams copy of the network was completely isolated from the other clones, allowing each team to find vulnerabilities and test exploits without risking other teams discovering what they found.

We encouraged all teams to find multiple vulnerabilities before the start of the competition. In so doing, each team could enter the competition with multiple ways to access (and sometimes escalate privileges on) and defend multiple target machines. Note that teams can find overlapping but different sets of vulnerabilities about a target system, making for an interesting and exciting dynamic during the live exercise as teams can have different methods of access to the same valuable machines.

4.5 Implants

Towards our goal of implant development, using the environment clone teams were given in advance, we require each team member to develop an implant to be deployed on a machine of their choice in the competition environment. Each team member develops a backdoor or implant, and we deploy them on a specific target machine of their choice before the competition begins. Therefore, no team knows what implants were developed by other teams, or on which machines they were deployed. These implants are designed to seed access to specific machines and offer a persistence mechanism for teams. In this manner, at the start of the competition, each team has multiple additional means of access or persistence to specific machines on the network that no other team is initially aware of. Teams are offered bonus points during the live competition for finding other teams implants, giving teams incentive to hunt and root out other teams' implants or backdoors.

4.6 Continual scanning

During the competition, a few additional, highly vulnerable, unscored, hidden machines are secretly added to the network that do not appear in the teams initial network copies or the scorebots top-down map of the network. These machines are easy to breach compared to the rest of the network, and can pose a threat to teams if other teams can attack them through previously unseen vectors. It is important for students to frequently scan their networks and territory to avoid a surprise attack from a

previously-unknown pivot, or to take advantage of an unknown pivot to launch attacks at other machines. This is done to mimic the threat faced often by real Network Operation Centers of new vulnerable or compromised machines being connected to networks by unknowing employees, insider threats, or malicious actors.

4.7 Scoring

Scorebot We run a global scorebot central to the network, reachable from all other machines. Students can trigger a "phone-home" at the scorebot to claim a machine. Students prove access to the scorebot by issuing a crafted GET request from the claimed machine. Students cannot spoof the scorebot from other IP addresses.

The scorebot has a web interface accessible to all teams with a top-down, live graphical map of the network, so students can immediately see which teams have claimed which machines and when machines are claimed. The scorebot displays a graph of the score for all the teams, so students can watch the overall progression of the game. The overall accessibility of each service is also displayed in the scorebots web interface, so students are given near immediate feedback of the integrity of the services they are defending. This dashboard helps create a more lively battleground, as it allows teams to stay abreast of where in the network the attacks are taking place, and where to target.

The scorebot is considered out of scope for the offensive actions for the students on the network.

The scorebot also periodically checks which services are up, and uses this information in its scoring. Every two minutes, the scorebot checks the integrity of the critical services on each machine, and teams are awarded points for each machine they control and all of the functioning services they have kept alive on that machine.

Points Points were automatically awarded by the scorebot as follows:

- 1 point if the machine is up/responding to pings
- 1 point for each critical service required by the machine (often HTTP, SSH, FTP, etc, which are usually also vulnerable, therefore requiring teams to keep these services up rather than just firewalling everything off)
- The total points for each round is multiplied with a (slowly) exponentially growing multiplier so that later rounds are worth more than earlier rounds, as machines are more secure by the end of the competition

5 Implementation

The competition backend was designed and run in Cypherpath, a virtual Software Defined Infrastructure (SDI) management program [5], allowing us to virtualize the entire network and all machines on it. The scorebot was deployed on an out-of-scope machine central to the network, and was written in Node JS. The network layout, machine information, vulnerabilities we seeded, and scorebot implementation are publicly available at `https://koth.cs.umd.edu`.

6 Sample Run

We deployed a King of the Hill exercise for our course on Introduction to Penetration Testing, and ran the live exercise for 3 hours. The class was split into 4 teams of 4–5 people, each labeled by color (Blue, Green, Orange, and Red). We designed a large vulnerable network, depicted in Figure 1. Each team was given an entry Kali machine connected to a different network, and no team at the onset of the competition could reach any other teams Kali machine directly. Each network included a set of machines that were connected either only to their network, or to other networks as well (pivots). Machines that do not act as pivots were worth more points, as they are harder for teams to reach initially (as you must pivot through another machine to reach them). One pivot exists between every pair of teams; these are depicted as the two-colored machines in Figure 1. A fifth central network was introduced, so that all pivot machines can communicate directly with all other pivots. We created six unique vulnerable images (4 Linux, 2 Windows machines), and duplicated them across the networks so that, from each teams perspective, the network was symmetric and no two duplicate images were connected to any one team's network.

Consider the network diagram in Figure 1. All machines that share the same image name are the same base image. The set of machines that each team can reach through one hop from their Kali machine is unique, and it is apparent that the network is symmetrical from each team's perspective. If teams develop an exploit for one of the pivots, they can use that pivot to launch that attack at its clone and increase their network control. The network diagram also shows which machines are inherently more valuable as pivots. From red's Kali machine, if the red team compromises the Linux FTP machine, the red team will be able to pivot through this machine to launch attacks at the machines that are connected only to the green teams initial network (Asset 1 and Asset 2).

Each Kali machine and one Windows machine were also given access to a second network switch that provided Internet access. In this manner, teams could down-

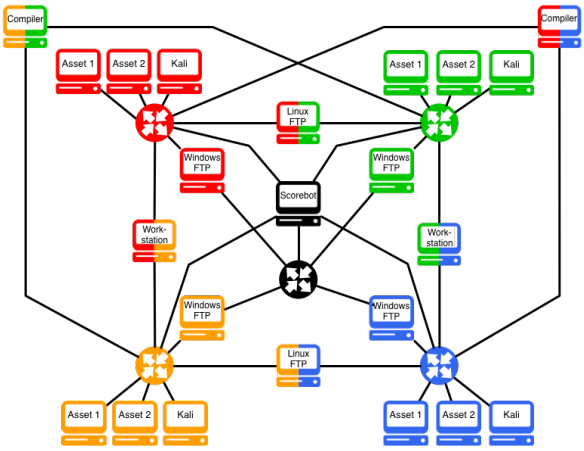

Figure 1: Network topology of our run of KotH. Four teams (Blue, Green, Orange, and Red) compete across this non-trivial network topology (mixed colors denote a pivot between those teams) a pivot between those teams). A central scorebot verifies that all critical services are running.

load patches, tools, or other software during the competition and move it around the network from these machines. Note that the rest of the machines in the network were isolated from the Internet.

Table 1 describes the critical services present on each of the six images, and their relative difficulty to compromise.

6.1 Results

For our course, students were given access to an isolated copy of the full competition environment two weeks in advance of the live competition, and were given an assignment to find two vulnerabilities on a machine on the target network. Every student completed the assignment, and many went well above and beyond—over half the class spent extra time to find at least four vulnerabilities, and some students penetration tested almost every image on the network. Note that we can only know that a team discovered a vulnerability if they submitted a report about it or exploited it during the live competition—we cannot measure if a student stumbled across a vulnerability but did not submit or use it.

Vulnerabilities Every vulnerability we seeded was discovered by at least one team, but no one team found every vulnerability in the environment. The set of vulnerabilities discovered by each team overlapped, and was unique between all four teams. We divide the vulnerabilities into two classes: access and local privilege escalation (LPE). Across all teams, students discovered every access vulnerability and most LPE vulnerabilities.

Prior to the competition, students attempted a wide range of attacks against the machines on the network. Teams performed MITM attacks between machines to capture network credential exchanges, and every team cracked many local password hashes for the local accounts to have more credentials for use during the live competition.

Students were quick to patch vulnerable services during the competition after gaining access. Competitors worked to enable and configure strong firewall policies to block traffic on unwanted ports, and then combed carefully through running services and processes to find malicious or vulnerable code. By the end of the competition, most computers were significantly more secure than at the beginning. For some vulnerabilities that were more time-consuming to patch (or would require a computer reboot, such as EternalBlue [9, 10, 17], students weighed the cost of lost points during patching against the risk of another team exploiting them—they usually chose to *leave them unpatched* (multiple teams specifically reported this for EternalBlue). Unsurprisingly, although students worked hard to patch the computers, some were hesitant about installing AntiVirus on the computers they had taken over for fear of losing their own access methods. Most students trusted their implants as a backdoor over the access methods they knew other teams could be aware of, and students usually accessed machines through their implants over other access vulnerabilities, if given the choice between them. Implants will be discussed in greater depth below.

Although students identified most access vulnerabilities, LPE vulnerabilities were most often missed by students during their initial penetration test. Only unprivileged access is required to trigger a phone-home to the scorebot from a given machine to claim a machine, so privileged access is primarily useful for bolstering access and acting defensively. Before the competition, students seemed to value unprivileged access to machines more highly than a full-chain of exploits (access + LPE) to gain fully privileged access, but this dynamic changed during the competition. Many machines had multiple teams simultaneously accessing them with unprivileged access, and it became a "race to root" of which team could escalate their privileges to kick out the other teams first and defend the machine. Students who did not find sufficient LPEs beforehand spent time identifying and enumerating new ones during the competition to protect their assets and prevent another team from taking down their services, and most were successful in identifying LPEs their team had not previously discovered. Through this process, students learned how to identify vulnerabilities and how they could be leveraged by an attacker.

Image Name	OS	Critical Services	Difficulty	Pivot?
Linux FTP	Linux	FTP, SSH, HTTP	Easy	✓
Windows FTP	Windows	FTP, IIS, RDP	Difficult	✓
Workstation	Windows	RDP, SMB, Msft. Net Discovery	Medium	✓
Compiler	Linux	SSH, Apache, Compiler	Easy	✓
Asset 1	Linux	FTP, SSH, HTTP	Medium	✗
Asset 2	Linux	SSH, HTTP, HTTP-ALT	Medium	✗

Table 1: Critical services across the images in our deployed competition. "Difficulty" refers to the difficulty to initially infect the critical service(s) or machine. Our "Compiler" application takes an input program and runs `gcc` on it.

Implants Students put a great deal of effort in developing their implants, and the quality of submitted implants was high. Although there were some simpler implants, such as custom written unauthenticated bind shells, some implants were significantly more complex. Multiple students developed various userland rootkits for Linux using LD_PRELOAD, and some were fully self-protecting or self-hiding. Even multiple students with root access to the machine could not delete one rootkit, as it had hooked functions used by *rm*, *ls*, and many more binaries. Multiple students built off open source rootkit projects, such as the Jynx Kit, to protect their implants [13]. One student, aware that other teams were working on userland LD_PRELOAD solutions, developed a small kernel module. Two students developed rootkits that infected the Linux PAM (Pluggable Authentication Module) to always allow their team's password to access the machine. Two students independently recompiled Bash from source to introduce hidden backdoors or enable restricted shells for users without their teams' passwords. One student recompiled a modified SSH from source so that it logged all password attempts in plaintext to a hidden file. Multiple students deployed custom encoded Metasploit backdoors for both Linux and Windows. Most teams trusted that their implants were secure from other teams, and multiple teams manually installed their implants on additional machines during the competition to bolster their access to machines.

Most teams were able to find and remove at least one implant from another team, but some implants were very well self-protected, and could not be removed. Some students noticed different behavior on certain machines caused by implants, and applied forensics to identify what type of implant was running and how it could be safely removed. Interestingly, as teams began increasingly relying on their implants, students began changing tactics to attack other teams' implants. One team began fuzzing a team's custom shell mid-competition, and succeeded in finding edge cases that caused unexpected behavior (although they did not have time to develop this into a functioning exploit). One team successfully found a hard-coded key in another team's implant, and began using that as their primary method of access to all machines that implant had been installed on.

Gameplay Each team pursued a different strategy at the beginning of the competition. The green team worked quickly at the onset to claim as many machines as possible, and get an early foothold in most of the network. However, although they took a large early lead, they overextended themselves, and could not effectively defend all of their machines simultaneously. The blue team began very defensively, and began slowly and carefully hardening both asset machines on their local subnet before working on attacking the pivots. The red team did not focus on their local network at all, and instead immediately tried to take over the pivots that could reach their network, reasoning that if they could defend the network border to their internal starting network, they could prevent teams from breaching their assets. The orange team took a metered approach between blue and green, and tried to take advantage of teams over-extending themselves while simultaneously hardening.

Halfway into the competition, two additional unscored Metasploitable 3 images were added to the network as pivots [15]. Multiple teams found the machines by to scanning the subnets they had access to and identified them as an opportunity and a potential threat. Although some teams left the machines alone for fear of spreading themselves to thin, one team successfully compromised and used them to access other team's networks.

As expected, during the competition, the pivots were highly contested territory, and every pivot changed hands at least twice during the competition. Few teams were successful in holding their pivots for long from other teams at the onset of the competition. On multiple pivots, multiple teams had manually installed their difficult-to-remove implants, so multiple teams had a simultaneous reliable method of access to these machines. For long stretches of time, no team could effectively or permanently evict the other teams; machines were reclaimed very frequently. Most teams did a good job defending the computers on their initial local subnet; very few of those machines changed hands during the competition—in fact, both the blue and orange teams never lost an asset once during the competition.

The red team was able to generate a Golden Ticket for the Domain Controller, and was able to use it during the competition to access the Domain Controller [11, 16]. They used it towards the end of the competition to push Group Policy to the Windows machines to give themselves access. Although extensive fighting over the Windows computers ensued between the green team (who

had control of the computers) and red team (who had control of the Domain Controller), having the Domain Controller on their side was too much for the green team to overcome, and most of the Windows pivots fell to the red team towards the end of the competition. This, combined with strong self-protecting implants, led the red team to win the competition.

7 Discussion

The King of the Hill exercise went well, and students enjoyed getting to practice with both offensive and defensive cybersecurity skills. Students were highly engaged during the competition, and students reported having learned a great deal from the experience. KotH also gave students the opportunity to explore novel attack vectors. Many students got to work with large open-source projects during their implant development, such as Bash, PAM, and SSH, and these students reported that they learned a lot from their implant development.

Overall, KotH gives students exposure to not just how to exploit vulnerabilities, but also how to patch, fix, or mitigate the damage from vulnerabilities. Since students are scored by the integrity of their services, merely turning off a service or computer is not always a viable option. Instead, students are faced with real-life cost-benefit analysis decisions between leaving a service unpatched, patching the vulnerable service, or disabling it.

A common problem with previous CTFs is that a go-to strategy for most defensive teams is to install patches as quickly as possible. Although this is good security practice, it removes the students' incentive to more deeply explore and understand *how* the vulnerabilities occur, how to exploit them, or even how to properly defend against them. Conversely, in KotH, students are asked to patch systems they control and later exploit them on other machines. This leads to an intimate understanding of both the offensive and defensive side of each of the vulnerabilities.

Finally, while KotH is not a real-world environment, it gives students real-world experiences and training in penetration testing. Students during the exercise discovered vulnerabilities on services not discussed in class and developed implants with technologies that we had not taught. Students pivoted effectively, and each team was able to leverage its arsenal of exploits during the competition against other teams.

Tailoring KotH There are many ways that KotH could be tailored or changed based on the technical level of the students and what the instructors wish to teach. To modulate the competition for less experienced students, the relative ease to breach a machine can be reduced. By introducing specific vulnerabilities to the environment, instructors can focus what students practice with. Instructors can also introduce lesser-known services, configurations, or operating systems if they want students to get more knowledge or experience using those systems. We have observed students researching exploits, defenses, and gaining a deep knowledge of various services or operating systems that were never covered in class, completely of their own volition.

We envision many other alternative designs for scoring that could be explored as well. Instructors could give incentives for defensive action by awarding points based on repelled attacks. A system-administration focused course could limit the weaknesses present in the machines themselves, and focus the competition on protecting service integrity. Forensic activity could be encouraged if the competition more heavily rewarded finding other teams' implants or discovering what offensive actions other teams took against their network. To train more purely offensive skills, the defensive requirements could be removed entirely.

Perhaps most importantly, pivoting presents a unique opportunity for instructors to encourage the usage of specific skills. By designing different paths in the network of how to get from a starting computer to a distant computer, instructors can build dependency chains of exploits that must be achieved to navigate from one target to another. It becomes easy for educators to customize and decide where they want the competition to be - the network topology makes it easy for instructors to express it, and easy for students to see and immediately start working on it. Tailoring implant requirements as well is a way for instructors to give students practice with specific technologies and skillsets in a creative manner.

8 Conclusion

We have introduced King of the Hill (KotH), a novel cybersecurity competition that provides students with hands-on experience with real-world penetration testing practices. KotH is distinguished from prior competitions in that it combines (1) network pivoting, (2) custom implant development, and (3) advanced preparation. An initial in-class run of KotH indicate that it creates an exciting environment in which students gain critical practice with valuable penetration testing skills. To facilitate adoption and adaptation, we have made our code and network maps available at `https://koth.cs.umd.edu`

Acknowledgments

We thank the anonymous ASE reviewers for their helpful feedback. This work was supported in part by NSF grants CNS-1409249 and CNS-1564143.

References

[1] P. Chapman, J. Burket, and D. Brumley. PicoCTF: A game-based computer security competition for high school students. In *USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE)*, 2014.

[2] N. Childers, B. Boe, L. Cavallaro, L. Cavedon, M. Cova, M. Egele, and G. Vigna. Organizing large scale hacking competitions. In *DIMVA*, 2010.

[3] A. Conklin. Cyber defense competitions and information security education: An active learning solution for a capstone course. 2006.

[4] G. Conti, T. Babbitt, and J. Nelson. Hacking competitions and their untapped potential for security education. *Security & Privacy*, 9(3):56–59, 2011.

[5] Cypherpath - software defined infrastructure. `https://www.cypherpath.com/`.

[6] DEF CON Communications Inc. Def con hacking conference. `http://www.defcon.org`.

[7] A. Doupé, M. Egele, B. Caillat, G. Stringhini, G. Yakin, A. Zand, L. Cavedon, and G. Vigna. Hit 'em where it hurts: A live security exercise on cyber situational awareness. In *Annual Computer Security Applications Conference (ACSAC)*, 2011.

[8] C. Eagle. Computer security competitions: Expanding educational outcomes. *IEEE Security & Privacy*, 11(4):69–71, 2013.

[9] Gtic monthly threat report, may 2017. `https://www.nttsecurity.com/docs/librariesprovider3/resources/gtic-monthly-threat-report-may-2017`.

[10] How to exploit eternalblue and doublepulsar. `https://dl.packetstormsecurity.net/papers/attack/exploiting-ebdp-en.pdf`.

[11] Complete domain compromise with golden tickets. `https://blog.stealthbits.com/complete-domain-compromise-with-golden-tickets/`.

[12] L. J. Hoffman, T. Rosenberg, and R. Dodge. Exploring a national cybersecurity exercise for universities. *IEEE Security & Privacy*, 3(5):27–33, 2005.

[13] Jynx kit userland ld_preload rootkit. `https://github.com/chokepoint/jynxkit`.

[14] Maryland cyber defense competition. `https://www.cybercompex.org/topic/mdc3`.

[15] Metasploitable3: An intentionally vulnerable machine for exploit testing. `https://blog.rapid7.com/2016/11/15/test-your-might-with-the-shiny-new-metasploitable3/`.

[16] Kerberos golden ticket protection - cert-eu security whitepaper. `https://cert.europa.eu/static/WhitePapers/UPDATED%20-%20CERT-EU_Security_Whitepaper_2014-007_Kerberos_Golden_Ticket_Protection_v1_4.pdf`.

[17] Eternalblue: Exploit analysis and port to windows 10. `https://www.risksense.com/_api/filesystem/466/EternalBlue_RiskSense-Exploit-Analysis-and-Port-to-Microsoft-Windows-10_v1_2.pdf`.

[18] A. Ruef, M. Hicks, J. Parker, D. Levin, M. L. Mazurek, and P. Mardziel. Build it, break it, fix it: Contesting secure development. In *ACM Conference on Computer and Communications Security (CCS)*, 2016.

[19] A. Ruef, M. Hicks, J. Parker, D. Levin, A. Memon, J. Plane, and P. Mardziel. Build it break it: Measuring and comparing development security. In *Workshop on Cyber Security Experimentation and Test (CSET)*, 2015.

[20] P. e. a. Sroufe. Experiences during a collegiate cyber defense competition. *Journal of Applied Security Research*, 5(3):382–396, 2010.

[21] A. B. Woszczynski and A. Green. Learning outcomes for cyber defense competitions. *Journal of Information Systems Education*, 28(1):21–38, 2017.