Shell We Play A Game? CTF-as-a-service for Security Education

Erik Trickel[†], Francesco Disperati[‡], Eric Gustafson[‡], Faezeh Kalantari[†], Mike Mabey[†],
Naveen Tiwari[†], Yeganeh Safaei[†], Adam Doupé[†], and Giovanni Vigna[‡]
[†]*Arizona State University* [‡]*University of California, Santa Barbara*
[†]{etrickel, fkalentari, mmabey, nktiwar1, ysafaei, doupe}@asu.edu
[‡]{francesco, edg, vigna}@cs.ucsb.edu

Abstract

Although we are facing a shortage of cybersecurity professionals, the shortage can be reduced by using technology to empower all security educators to efficiently and effectively educate the professionals of tomorrow. One powerful tool in some educators' toolboxes are Capture the Flag (CTF) competitions. Although participants in all the different types of CTF competitions learn and grow their security skills, Attack/Defense CTF competitions offer a more engaging and interactive environment where participants learn both offensive and defensive skills, and, as a result, they develop their skills even faster. However, the substantial time and skills required to host a CTF, especially an Attack/Defense CTF, is a huge barrier for anyone wanting to organize one. Therefore, we created an on-demand Attack/Defense tool via an easy-to-use website that makes the creation of an Attack/Defense CTF as simple as clicking a few buttons. In this paper, we describe the design and implementation of our system, along with lessons learned from using the system to host a 24-hour 317 team Attack/Defense CTF.

1 Introduction

We are facing a cybersecurity crisis because the demand for cybersecurity professionals is growing exponentially and the supply-side is unable to create enough qualified professionals [30]. In fact, the 2015 $(ISC)^2$ Global Information Security Workforce Study predicts a shortfall of 1.5 million global information security jobs by 2020 [32]. The lack of a qualified cybersecurity workforce gives rise to high-profile security incidents, such as the Office of Personal Management data breach, where hackers stole 21 million files containing sensitive background check information [23]. In addition, attacks against the nation's critical infrastructure could have devastating effects that go well beyond the financial losses we are witnessing today.

Over the last thirty years, the world of hacking has morphed from an idealistic place populated by young and curious explorers who only wanted to hack the planet and keep the world of electrons free for all[1] into a world-wide battlefield in which nation-states and criminal organizations launch attacks to access and control the flow of information. As a result, modern attackers have gone from a chaotic group seeking to explore their digital world to highly sophisticated and coordinated actors that patiently wait for the optimal moment to engage their enemies. Modern hackers leverage vulnerabilities on one system to compromise another, and then, as stealthily as possible, they slowly exfiltrate the data they seek and eliminate any evidence of their activity. This metamorphosis of the hacking world demands an equal rise in the skills of security professionals and security-minded developers. Therefore, we must train the next generation of security professionals who will secure the software systems that run companies, organizations, and the nation's critical infrastructure.

Growing the cybersecurity workforce is a challenging problem because the knowledge and skills are a complex and constantly moving target. Solving security problems requires strong objective critical thinking skills [14, 25, 39]. In other words, developers must learn to think like the attackers and then learn to defend against those attacks and exploits. Although studying vulnerabilities and their abstract patterns provides a theoretical start, it is not enough—hands-on practice is crucial for mastering the highly-complex theoretical concepts involved in cybersecurity [37]. Solving real problems in small cooperative groups can greatly improve knowledge acquisition [28]. In addition, gamification of the learning experience also produces several positive effects [19]. Thus, it is not surprising that live cybersecurity competitions, which take advantage of those ideas, are on the rise.

Attack/Defense Capture the Flag events (ADCTFs) are a type of live cybersecurity competition that attempts to maximize the learning for the competitors. In an ADCTF, the participants practice finding vulnerabilities, developing exploits, and defending against exploits. Additionally, the

[1] See Loyd Blankenship's essay *The Conscience of a Hacker*, aka *The Hacker's Manifesto*, which he said he wrote to describe the essence of "what [hackers] were doing and why we were doing it" [10, 38].

competitors are often in teams, which further increases their learning [24, 28]. Beyond the learning that takes place during the competition, many competitors also experience significant learning in preparing for the competition and creating write-ups after it concludes [12].

Although organizing and running challenge-based competitions is relatively simple[2], organizing and running an ADCTF competition requires a significant amount of time and a broad range of skills. An ADCTF organizer must spend a large amount of time to meticulously build a secure infrastructure for hosting the game. Moreover, the organizer must develop some type of application that securely controls the game and scores the participants' activities. All of this means that an organizer must be an expert in operating systems, networking, application development, and server administration to successfully organize and host an ADCTF.

To address this pressing need, we relied on our experience gained over the last fourteen years hosting ADCTFs and created a CTF-as-a-Service platform, which is now available at `https://ShellWePlayAGame.org` (SWPAG) and the source code is available on GitHub [22]. On SWPAG, anyone can organize and host their own ADCTF. After filling out the proper information, a completely configured and stable ADCTF is created in a cloud environment, thus relieving the organizer of the operating system, networking, and server administration burden. Our goal is that SWPAG will empower all security educators, even those with limited network or administrative skills, to easily host their own ADCTF for educational purposes.

2 Background and Motivation

Live cybersecurity exercises benefit the security community in several ways. First, the exercises allow the participants to practice the theory and concepts they have acquired from books and articles [35]. Second, the real-time aspect of a finite event that occurs for a limited amount of time and the competitive-drive of the participants improves learning [19]. Third, live cybersecurity exercises provide a deeper engagement and increase academic learning time, which results in faster learning and mastery of the concepts [16, 17, 33]. Fourth, the participants learn how to operate in a dynamic setting, having to react to attacks by developing, on the spot, defenses and countermeasures. Last, the events allow participants to showcase their skills.

Collectively, we have been organizing, running, and competing in cybersecurity competitions for many years. From this unique vantage point, we have seen first-hand the effect that live cybersecurity competitions have on the participants, who are driven to invest a substantial amount of resources in preparing, executing, and post-evaluating. Preparation includes classroom learning, peer teaching, independent study, and the creation of novel tools. Execution requires them to think critically and generalize their theoretical knowledge while having to react, in real time, to unexpected circumstances. The post-evaluating entails objectively evaluating their performance, discussing the effectiveness of their attack and defense mechanisms, and studying solutions to the problems they could not solve. Many participants learn even more by taking the time to write blog posts that discuss their lessons learned, the details of how they found and eventually exploited the vulnerability, and their strategy and approach to the competition. As a result, these participants have a tendency to grow and improve after every event.

The first cybersecurity competition was held in 1996 at DEF CON [20]. The early DEF CON competitions were in what is now considered a challenge style using a single host with custom-written vulnerable services. The participants would discover vulnerabilities in each service and then prove it by crafting an exploit. Even though this style of event focuses purely on offensive skills, it is still an excellent way for participants to practice and refine their security skills.

The next iteration of CTF competitions allowed participants to refine *both* their offensive and defensive skills: Attack/Defense Capture the Flag events. This type of event is an interactive competition in which each team receives an identical machine that is running vulnerable services. The competitors then use their security skills to protect their own services while simultaneously trying to break into the same services on their opponents' machine. Once successful, the competitors must obtain proof that they succeeded at exploiting an opponent's service by gaining access to a unique piece of data referred to as a *flag*. With this flag in their virtual hand, they must then turn it in to a scorekeeper for points. ADCTFs are a fun and exciting way for security researchers to showcase, enhance, and refine their security skills while also competing with one another for fame and glory.

Since 2003, we have hosted the international Capture the Flag (iCTF) competition, which was not only one of the first ADCTFs but is now one of the largest [37]. We have continued to host the iCTF every year since then (the most recent edition was in March of 2017). Each year, we experiment with various designs and approaches to the game [12, 15, 31, 34, 36].

After running the competition for fourteen years, we recognized that many of the game infrastructure components were reused year after year [37]. Therefore, in August 2014, the UCSB SecLab released an open-source framework for hosting interactive CTF competitions with the hopes of easing the burden on other ADCTF organizers and to give educators access to an ADCTF competition for their classroom [22]. By abstracting the common infrastructure (starting services, scoring, service checking, VM creation) and by defining a common interface to create services, the authors enabled anyone, with significant manual effort, to create and host an

[2] In fact, there are only a handful of ADCTFs, while most available competitions are challenge-based [8].

ADCTF-like competition. Even though the iCTF framework provides the components necessary to run an ADCTF event, their setup and configuration is far from trivial. An organizer must still spend a significant amount of time understanding how the components work. After that, she must create the network, take the time to deploy the machines, and create and install vulnerable services. In addition, an organizer must debug any components that fail to work properly, which can involve investigating the database, finding the various log files on each of the machines, and even patching bugs. Thus, the technical barrier to adoption is still substantial because the would-be organizers must understand a great deal about networking, server administration, network security, application development, and application security.

This led us to realize that the community would benefit greatly from a turn-key solution. To validate this conclusion, we surveyed the teams that participated in our competition in 2015. We asked them "If you could press a button on a website to automatically host your own CTF competition, with no technical setup on your part, would you or your group use it?" 31 out of the 36 responders answered that they would.

All of this pushed us toward taking the open source platform to the next level and offer it as an easy-to-use service; thus, now we are proud to present our CTF-as-a-Service solution, which is available at `https://ShellWePlayAGame.org` (SWPAG). SWPAG offers the capability to launch an ADCTF that leverages the computing resources of the cloud. After entering some information and clicking a few buttons, an organizer launches a CTF instance, which is created and configured within a few minutes on Amazon's Web Services platform (AWS).

3 Design of the CTF-as-a-Service

While the SWPAG website acts as the front-end for users to configure an ADCTF, behind the SWPAG website the CTF-as-a-Service platform has one master controller called the Games Controller (GC), which is responsible for managing all of the CTF instances (see Figure 1). A CTF instance is the logical space containing all the virtual machines (VMs) necessary to run a single ADCTF event. In this section, we describe the GC and the CTF instance components.

3.1 The Games Controller

The SWPAG website is the web front-end that organizers use to manage events. An organizer can be anyone—including students, educators, CTF teams, or organizations. Initially, an organizer must create an account. After creating an account, she may create a new CTF instance and modify any of the settings. The organizer may choose to select intentionally-vulnerable services from a library of existing services and eventually may write and upload her own vulnerable services, which will then become a part of the library

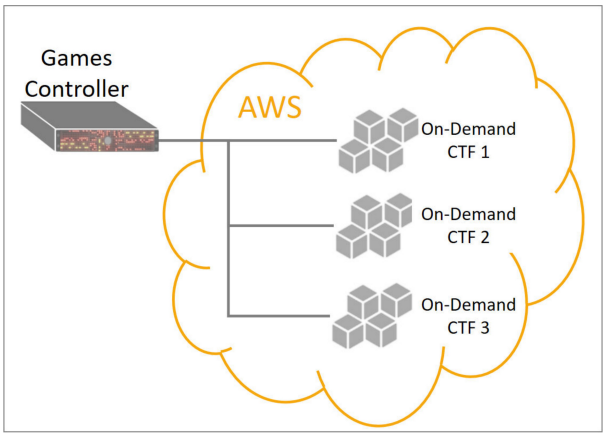

Figure 1: CTF-as-a-Service Overview.

and available to other educators[3]. In addition, the organizer chooses various parameters for the competition, such as the number of teams, the members of the teams, the game start time and duration, and so on. Next, the organizer provides credentials for a valid account on Amazon Web Services (AWS), which is currently the only cloud service provider that is compatible with the platform. Once setup is complete, the teams wishing to participate in an event can register for it. When the organizer is ready, she clicks a button to launch the CTF instance and the GC takes care of the rest.

The GC is responsible for creating, managing, and terminating all the CTF instances. Using the supplied credentials, the GC accesses the AWS account and creates a Virtual Private Cloud (VPC) for each CTF instance. VPCs are a networking feature of AWS that enable the provisioning of a logically isolated section of AWS's cloud [6]. Within the VPC, the GC has full control of the IP addresses of the servers and the network routing between them. The VPC is configured to keep all network traffic within it, and, as a result, the attacks launched during the competition cannot affect external hosts. After the GC configures the VPC, it creates the Game Master (GM).

3.2 The CTF Instance Components

The GM automates many of the difficult and time consuming tasks. It is responsible for communicating with the GC and for orchestrating the creation and management of the game. After the GM's creation is complete, it starts by configuring the CTF instance and then creating the Database, Router, Gamebot, Scriptbot, Team Interface, Scoreboard and the team VMs.

[3]Much like a riddle, the difficulty of exploiting a vulnerable service is in its novelty, as such, the services stored in the library will be easier because the prior contestants will often post their analysis and solutions. However, even with leaked solutions, it is often still difficult for less experienced students to successfully implement an exploit.

The first component the GM instantiates is the Database. The Database is the central component of the game's operation—stores all the information associated with the competition (e.g., the flags submitted, the status of the services for each round, and the team's information). Being the central component of the game, all the other components access the database, except for the vulnerable team VMs. The components access the Database over a private subnet that is different from the one used by the team VMs. As a result, the Database is inaccessible from the team VMs or from the Internet. We limited access to the Database to reduce the attack surface area of the game infrastructure. We also designed it this way so that the unencrypted database communications were protected because they only travel over the private subnet.

The second component created by the GM is the Router. The main purpose of the Router is to masquerade transmissions to the team VMs, to capture the traffic, and to act as a single entry point for the teams. First, it forwards all team-to-team transmissions and official service verification transmissions that verify each team's service is running properly. While forwarding these transmissions, it anonymizes the packets by masquerading all the traffic as itself. Thus, when a team receives a packet it has the source IP of the Router. We designed it this way to prevent teams from dropping traffic from their competitors while allowing the service checks to get through. One interesting development note is that this was not as straight forward to setup as we thought because the AWS network prevents masquerading by default [7]. To bypass this restriction, the source/destination checking must be disabled.

The Router captures, stores, and potentially limits all the traffic that it forwards (i.e., team-to-team and Scriptbot-to-team traffic). Even limiting to only team-to-team and Scriptbot-to-team traffic, the logs (in raw pcap format) grow rather quickly. For example, for our most recent competition the compressed traffic logs were more than 100GB. However, this competition was for 24 hours with 317 teams, so we expect SWPAG CTFs to have much less traffic. Next, it limits the number of connections per second each team may initiate to another team; however, it does not limit the maximum number of connections a team may have open concurrently. Once the Router creation completes, the GM instantiates the Gamebot, Scriptbot, Scoreboard, Team Interface, and the Teams' VMs in parallel.

The Router is created with a static external IP, and it serves as a single entry point for teams to access their VMs. It does this by forwarding ports 1337 and higher to each team's SSH port. We chose to design the team's access this way because we found that stopped and started, recreated, or upgraded team VMs would receive a different public IP address [2]. We could have designed it so that every team received a static IP address, however (1) Amazon limits the number that can be used per account and (2) we were unsure of how many needed to be requested. The chosen port forwarding method allows us to have 1,000 teams without needing to request an increase in the number of static IP addresses.

The Gamebot is the heartbeat of the game. The game duration is divided in ticks. A tick does not occur after a fixed and constant amount of time, instead, it occurs after a fixed amount of time plus a random adjustment. After each tick, the prior round ends and a new round begins. At end of a round, the Gamebot calculates the score for each team based on their performance during the prior round.

The next component is Scriptbot. Prior to a game starting, Scriptbot sits and waits for Gamebot to create the first tick of the game. Once Scriptbot sees the first tick, it will tirelessly test the teams' services and update the flags on each team's VM every round. The test and update processes execute in parallel, but to obfuscate itself and to spread out the load the Scriptbot generates a randomized delay for every process it must execute in a round. The maximum delay is set so that Scriptbot will complete all the processes before the end of the current round. In addition to the randomized delay, Scriptbot accesses the team's services via the Router, which masquerades all the traffic, so that Scriptbot's requests look the same as the team-to-team traffic. After it executes each process, Scriptbot updates the database with the results.

The Team Interface is both the keymaster and gatekeeper. Using the Team Interface, the teams retrieve their private SSH keys so that they can access their team's VMs. The Team Interface also allows them to retrieve a flag identifier for the round. The flag identifier is a value that will help them find the flag on their opponent's machine. For example, it might be name of the file they must look inside once they exploit the associated service. In addition, the Team Interface will provide each team with a unique flag token so that they can submit a flag without needing to use their username and password. Next, the Team Interface accepts any flags that teams submit. To access the Team Interface, the teams must retrieve a login access token and a flag submission token from SWPAG. For more information on the flag mechanisms, see the article *Ten Years of iCTF: The Good, The Bad, and The Ugly* [37].

The last system component is the Scoreboard. The Scoreboard provides feedback to the teams on their performance. On the leaderboard section of the Scoreboard, it displays each team's score and a graph showing the historical performance of the top teams. On the service list of the Scoreboard, the status of the team services are shown, so that the teams can evaluate if their security mechanisms are affecting the functionality and availability of their services.

Next, GM creates a virtual machine on AWS for each team. Although GM creates the instances in parallel, it limits the number of concurrent requests to avoid receiving a request rate limit exceeded error from AWS [1]. To keep their system available for all their users, AWS has a

fluctuating request rate limit and if an account exceeds the limit they receive the error[4].

During the team VM creation process, the GM installs and starts the vulnerable services chosen by the organizer. It also configures the VMs with a static route that forces the team-to-team traffic through the Router. If a team decides to change or remove this static route (which they can do because each team has root access to their own VM), then they will be unable to communicate to the other teams because direct team-to-team traffic is blocked by an AWS security group (see Section 3.3 for additional details). Once the GM completes instantiating and configuring a team's VM, GM tests the VM's vulnerable services.

3.3 Network Configuration

Each CTF instance must have several network configuration steps completed before a game can start. As mentioned previously, the GC creates a VPC. The new VPC is assigned an IP address range of 172.31.0.0/16. Within the VPC, the GC creates two subnets. The first subnet is the Game Components subnet, which is limited to 172.31.64.0/20. The Game Components subnet contains all the game servers, GM, Database, Gamebot, Scriptbot, Team Interface, and Scoreboard. The second subnet created by GC is called the War Range subnet. The GC defines the War Range subnet as 172.31.128.0/17. The War Range contains the teams' VMs. However, the GM limits team IP addresses to 172.31.129.0/19, which means it can currently only handle 8,190 machines, however the largest ADCTF ever held had only 317 machines it should be sufficient for the foreseeable future.

The Scriptbot is the only machine in the network that is dual homed to both the Game Components and War Range private subnets. The Scriptbot is dual homed so that it can create a static route to the Router, which obfuscates its origin while running its service tests on the team VMs.

For a virtual machine to be accessible, it must be associated with an AWS security group. An AWS security group is a virtual firewall that permits inbound and outbound traffic based on the rules assigned to it [3]. The security groups reside on the network and are inaccessible to the VMs—in fact, unlike a firewall running on a VM, packets not permitted by a security group are dropped before ever reaching the VM.

The GM associates every virtual machine in a CTF instance with one of four AWS security groups. The first group protects the servers that are only internal. This group only permits connections on ports 80 and 22 so long as the connections originate from the Game Components subnet. Similarly, the web security group has the same restrictions except that it allows Internet traffic to connect using ports 80 and 443. The next security group protects the Router. The first rule permits access to all addresses connecting to ports 1024-2352, which

[4]Through trial and error we have found it is unlikely we will receive the error if we limit the number of concurrent requests to ten.

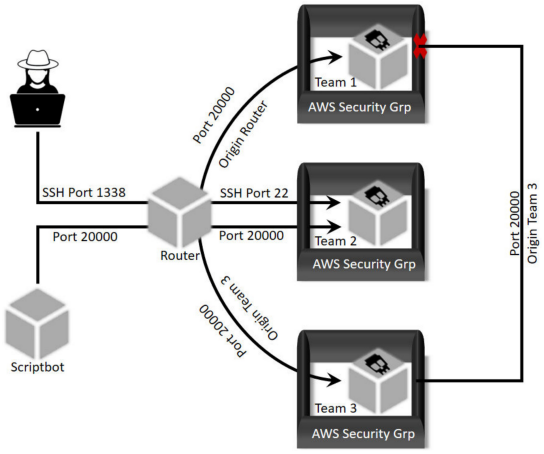

Figure 2: Scriptbot, Router, and Team VM Connectivity.

are the ports used for the SSH port forwarding (see the connection from the user to her VM in Figure 2). The Router security group also allows connections to the ports between 1024 and 65535 if the connection originates from the War Range, which is represented by an arc between the teams in Figure 2. The fourth security group is for the teams. The only rule in this group permits access to ports 1024-65535 if the connection originates from the Router, which means the only way to connect to another team is by sending packets through the Router. Referring to Figure 2, notice the arcing connection through the Router and the connection from Scriptbot are permitted on port 20000, whereas, the direct connection from team three is prevented by the AWS security group. This was designed like this because we wanted to give each team root access to their VM, and, as a result, each team's VM must be considered hostile and outside of our control. However, the VM-independent nature of the security groups provides a simple mechanism to achieve the desired effect.

3.4 Intelligent Component Recreation

As anyone who has created a machine from scratch knows, it is not uncommon for some small part of the install process to fail, and, unfortunately, this happens when creating VMs in the cloud as well. Thus, GM has a robust and extensible set of tests that it runs to verify the VMs are operating correctly. For the game components, the tests verify that the machine is accessible, the proper ports are open and responding appropriately, and the internal services are up and running. For the team VMs, it checks each of the vulnerable services by running the same scripts that will be used to verify service operation during the competition. If any of the tests fail for a machine, the GM automatically destroys the machine and recreates a new one from scratch.

4 Validation

We expected that the CTF-as-a-Service platform would scale to handle large-scale events, however we would not know for certain unless we tested it. Thus, after completing the development, we ran a load test with 250 teams to uncover any latent defects and evaluate its ability to manage a large-scale event. After fixing the issues, we torture tested it and the AWS network by hosting the 2017 international Capture the Flag event (iCTF) in which 317 teams competed for fame, glory, and an entry into the 2017 DEF CON CTF.

4.1 Load Testing, Round One

To understand the performance characteristics of the CTF-as-a-service framework in a large-scale environment, we ran a three phased load test. In phase one, we wanted to verify the components would work with a large number of teams. In the second phase, we tested the infrastructure with random team-to-team traffic and team flag submissions. In the third phase, we focused all the team-to-team traffic on one team VM. For the tests, we created 250 team VMs with ten vulnerable services running on each machine. Each team VM was configured with four processors and 16GB of RAM.

Once all the machines were running, the infrastructure reported that most of the team's services were down even though all of the machines passed the GM's verification tests. After some investigation, we realized that the Scriptbot was not able to execute all its tests within a single round. In a round, it needed to run about 8,800 scripts (each one running in a separate process), which required far more processing power and memory than we expected. After increasing it from 8-processors to 32-processors, it was completing most the time. However, we realized that the number of Scriptbots would need to be increased for large scale events to ensure that all the teams' services were properly validated each round.

Fortunately, the other infrastructure components operated as expected while the game was idle and we were able to start the second phase. For this phase, we started simulating traffic between the teams and performing flag submissions to the Team Interface by each of the teams. All the components performed exceptionally well. We had zero issues while this was running. Moreover, we ran this test for several hours and during the entire test the Scriptbot successfully verified the services on all the machines.

For the third phase, we had 249 of the team VMs connect repeatedly to a single team VM to see how the victim would handle the directed attack. During the test, the victim VM, surprisingly was able to respond to requests, and we were also able to SSH to the box, however the response time for both were exceptionally slow. During all the phases, the Router handled the load well even though it was only using a two processor instance.

4.2 The Second Load Test—iCTF 2017

We successfully tested the scalability of our CTF-as-a-service framework by using it to host the iCTF event on March 3rd, 2017. This edition of the competition was different from previous years in that it (1) was open to the public, (2) lasted 24 hours, and (3) was a DEF CON CTF 2017 qualifier; whereas, in prior years we have limited it to academic teams, only eight hours, and was not a qualifier. For this competition, we developed ten vulnerable services, and 317 teams registered.

To reduce our risk of infrastructure failure, we decided to over-provision the CTF instance. First, based on the load tests discussed in Section 4.1, we choose to create four Scriptbots that were each responsible for testing a particular subset of the teams. Next, we choose to configure the Database and all four Scriptbots with a VM that had 36 processors and 60GB of RAM. We configured the Gamebot, Scoreboard, Team Interface, and Router with a VM that had 16 processors with 64GB of RAM. Last, we configured the teams' VMs to use a 4-processor machine with 16GB of RAM.

When we started the competition, the infrastructure withstood 317 teams pounding each other and did not suffer from any infrastructure problems for the first 18 hours. However, just after the 18th hour of the competition, the infrastructure started to crumble. Specifically, the Router stopped allowing connections between the teams. During the competition, we tried to fix the issue, and, although we suspected it was a DOS attack, we could not convince the offending team to stop. So, unfortunately, we had to end the competition early.

After an in-depth forensic investigation [21], we discovered that a team cheated[5] and used their custom-developed in-game botnet (running on nearly all 317 teams VMs) to launch a DDoS attack against another team. In this attack, the bots opened a connection to the victim machine and then terminated it, however, it never sent a FIN packet. This caused the Router to hold each connection open until it timed out. As the number of connections grew, the Router reached a point where it was unable to accept new connections.

With the over-provisioned configuration, the cost of the 1,504 processor infrastructure for twenty-four hours was approximately 3,500 USD, which Amazon covered with a generous sponsorship. Fortunately, smaller and shorter competitions should cost a fraction of that amount. For example, a six-hour competition with one hour for setup and twenty teams should cost less than 50 USD[6].

Despite the challenges, the 2017 iCTF load test proved that it is possible to leverage the cloud to support large-scale ADCTF competitions.

[5]The iCTF, as a hacking competition, does not have many rules, however DoS attacks are explicitly against the iCTF rules.

[6]For example, creating a six-hour game that uses two t2.2xlarge instances for Database and Scriptbot, four t2.large instances for the remaining components, and twenty t2.medium instances for the teams, is estimated to cost 25 USD.

5 Lessons Learned

While developing and running the CTF-as-a-Service framework we ran into several issues that we will discuss in this section. Fortunately, most of the problems we uncovered are solvable, and while we have addressed many of them, we believe that they will serve as useful lessons learned to those developing complicated distributed systems.

While trying to create a large game, we found that we could not have more than ten components starting up at the same time (see Section 3.2). This limitation is from a request limit imposed by AWS, and it forced us to limit the number of simultaneous instantiations to ten. As a result, bringing up 326 VMs takes over three hours. However, we found that we could cut the process by one-third of the time by using a custom Amazon machine image [4]. To create the image, we first create a team VM with all the services installed and running properly and then have AWS make a private image of the VM.

Another interesting issue related to instantiating a large game, is that sometimes AWS may not have enough available resources to create the machines as fast as we are requesting. For example, while trying to instantiate the 317 team VMs, with four processors each, AWS stopped allowing new instances and reported that it had run out of resources for the configuration we were using within the availability zone being used. AWS has several geographically dispersed regions that are designed to be completely isolated from each other [5], called availability zones. Within each region, a VM can be assigned to a particular availability zone. However, each subnet of the VPC must reside entirely within one availability zone. As a result, the obvious work-around failed because we could not simply start bringing up machines in a different availability zone and the design would not support spanning multiple subnets. So, to bring up the boxes, we would have to wait a few minutes after receiving the error and restart the process.

As discussed in Section 4.2, we experienced a DDoS attack during the 2017 iCTF competition where nearly all the teams bombarded a single victim. Even though we load tested an attack from 249 machines to a single machine, and found that it could withstand the attack, we did not test what would happen if the attacking machines did something closer to a SYN-flood DDoS attack. In the near future, we plan to research what happened by creating a game and recreating the attack. We will use this environment to understand exactly what went wrong and to devise a solution. In addition, we will use this test to explore the possibility of using a monitoring application to notify us when the network is starting to experience connectivity issues.

While running the iCTF competition, we found that the GM's component testing and recreation process is too smart and made debugging difficult. The retry logic is absolutely necessary for the automated CTF-as-a-Service environment. In a production environment, if something fails, it is probably an issue with the instantiation so often destroying and recreating the component will solve the issue. Therefore, it is no surprise that this works great in a production environment with a stable code base. However, this approach is not applicable in an environment with unstable code, like the iCTF event. For example, if a mistake is made while making changes to a vulnerable service and it no longer works, the developer needs the machine to continue running so that she can debug the problems. However, the GM, not realizing this, will destroy the VM and attempt to recreate it until it runs out of retries and then it will simply destroy the component. As a result, with the component destroyed, the developer cannot view the logs or otherwise investigate what was actually causing the error.

Shortly after the competition began, we realized that many of the teams were given access credentials to two team VMs instead of one. To understand how this occurred it is necessary to explain how the Team Interface worked for the registration phase and the execution phase. The CTF-as-a-Service platform was designed to run as a single unit. However, for the iCTF, we needed to have a registration server up and running several weeks prior to the event starting (SWPAG will handle all registration for the CTF-as-a-Service). To do this, we created a CTF instance and left only the Database and Team Interface components running because team registration and verification was handled by those components. Once we closed registration a few day prior to the start of the competition, we exported all the teams from the Database and loaded them into the production iCTF instance. During the loading process, each team's identifier was regenerated by the process, so a team had an identifier in the old system that was different from the new one, but everything else was the same. Just before the competition started, we switched the DNS from the old Team Interface to the production instance. Within a short period of time, we realized that since the DNS was the same and the team's sessions did not expire they could see the login credentials of the team that received their old identifier (because the Team Interface stored the session information in a client-side cookie). For example, the team zero_cool with the identifier of 117 on the registration server could retrieve the private key for crash_override, which was team 117 on the production server.

The only true solution to this we could devise during the event was to create a completely new production CTF instance. Fortunately, we were able to bring up the second CTF instance in a different availability zone while the participants played on the first. So, with not much more than the push of a button, we had two CTF instances running concurrently using nearly 650 VMs. After the first eight hours of the competition, we instituted a break during the break we disabled the first production instance and pointed players to the new game, which ensured that each team only

had access to a single machine unless they found some other way to compromise them[7].

The iCTF-specific issues highlight the requirement differences between a general CTF-as-a-Service versus what is required when hosting an iCTF event. First, the iCTF is often closer to a development environment because we are constantly trying to push the envelope and find new and interesting ways to execute an ADCTF. So, for the iCTF we need less of an automated black box and more direct access to the components and configuration. Second, the iCTF competition is also the largest ADCTF, and we expect most organizers using SWPAG will host events with less than 50 teams—in fact, we will limit the size of events that SWPAG will host automatically. Third, the iCTF has open registration for its events whereas for CTF-as-a-Service SWPAG will handle registration.

6 Related Work

Although they have been around for many years the difficulty and time constraints have resulted in only a few online ADCTFs being held each year. In the United States, the two largest being our iCTF event and RuCTF.

Buena Vista University's ADCTF is a cloud-based infrastructure that is geared towards giving participants a gentle introduction to an ADCTF competition [9]. The goal of the organizers is to keep the event small so that the complexity of successfully competing is reduced. As a result, the environment relies on a single administration VM that takes care of managing the services, flags, and scoring. While some similarities exist to our CTF-as-a-Service framework, the stated goal of their system and its subsequent design are significantly different.

Another group working in a similar area is the joint team working on the Build It, Break It, Fix It (BIBIFI) contests [29]. In these contests, the participants first build a system according to the specifications published by the organizers. The teams submit their solutions and are scored based on their conformance to the specifications. Next, the teams enter the break it phase. When the breakers believe they have found a defect, they submit the flaw with an explanation. Their submission is automatically scored and more points are awarded for security vulnerabilities. In the fix-it phase, the build teams receive the bug reports and must fix the discovered flaws. This type of contest is similar to an ADCTF and provides an exciting learning opportunity for the teams. However, it is not currently offered in a framework that could be easily implemented by those that might wish to host their own BIBIFI event.

The Cyber Range Instantiation System (CyRIS) enables educators to automatically deploy and manage cyber ranges for cybersecurity education [27]. Similar to ADCTFs, a cyber range is a controlled virtual environment that is used to give participants hands-on security experience. While this work is interesting, this research takes a different approach to education and lacks the game aspect of ADCTFs, which pushes students to go beyond the call of duty. Moreover, the cyber range still requires the organizer to possess a certain level of sophistication, our expectation is that we will empower even less savvy organizers than CyRIS.

PicoCTF is designed to increase interest in computer science among high school students [11]. PicoCTF is an attack-focused style of competition. Participants interact with it using a web-based graphical user interface, which is designed as an interactive game. The game even features cut-scenes, sound effects, four levels, and 57 challenges. PicoCTF is different from our CTF-as-a-Service because its target audience is different, and it does not offer any defense exercises.

For the last ten years, the Zero Day Initiative has hosted the Pwn2Own event at CanSecWest [18]. In the Pwn2Own hacking challenge, participants try to compromise the security of various up-to-date computer devices and if they do, they win the device or money. This event differs from the CTF-as-a-Service because it is an attack-only style and its goal its goal is to help vendors find 0-day vulnerabilities and not helping to educate the participants.

Another style of competitions focus on network defense. In these competitions, participants protect their networks by reacting to intrusions from external attackers [13, 26]. This style of competition features only network defense, and, unlike ADCTFs, they do not have an attack component for the competitors.

7 Conclusion

SWPAG is a powerful educational tool that empowers anyone to launch their own ADCTF leveraging an easy-to-use interface. Although ADCTFs provide several benefits to teaching security professionals, until now, the creation of an event was a substantial undertaking that required a broad range of networking and administration skills. SWPAG leverages AWS and UCSB's open source iCTF framework to provide a secure environment for teaching the security professionals of tomorrow. While it is still in the early stages of its development, the platform has already survived a 317-competitor ADCTF event and is ready to support future online ADCTF events.

8 Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant 1623246 and 1623269 and by the generous contributions of Amazon Web Services (special thanks to the AWS Security team).

[7] For example, we do not advise participants to post the email address and password for their team to the public chat channel.

References

[1] Amazon API Error Codes. http://docs.aws.amazon.com/AWSEC2/latest/APIReference/errors-overview.html, 2017.

[2] Amazon EC2 Instance IP Addressing. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-instance-addressing.html, 2017.

[3] Amazon EC2 Security Groups for Linux Instances. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html, 2017.

[4] Amazon Machine Images (AMI). http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html, 2017.

[5] Amazon Regions and Availability Zones. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html, 2017.

[6] Amazon VPC FAQs. https://aws.amazon.com/vpc/faqs/, 2017.

[7] AWS NAT Instances. http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_NAT_Instance.html, 2017.

[8] CTF Time. https://ctftime.org, 2017.

[9] BACKMAN, N. Facilitating a battle between hackers: Computer security outside of the classroom. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (New York, NY, USA, 2016), SIGCSE '16, ACM, pp. 603–608.

[10] BLANKENSHIP, L. The Conscience of a Hacker. http://phrack.org/issues/7/3.html, 1986.

[11] CHAPMAN, P., BURKET, J., AND BRUMLEY, D. Picoctf: A game-based computer security competition for high school students. In *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)* (San Diego, CA, 2014), USENIX Association.

[12] CHILDERS, N., BOE, B., CAVALLARO, L., CAVEDON, L., COVA, M., EGELE, M., AND VIGNA, G. Organizing Large Scale Hacking Competitions. In *Proceedings of the Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA)* (Bonn, Germany, July 2010).

[13] CONKLIN, A. The use of a collegiate cyber defense competition in information security education. In *Proceedings of the 2Nd Annual Conference on Information Security Curriculum Development* (New York, NY, USA, 2005), InfoSecCD '05, ACM, pp. 16–18.

[14] DASGUPTA, D., FEREBEE, D. M., AND MICHALEWICZ, Z. Applying puzzle-based learning to cyber-security education. In *Proceedings of the 2013 on InfoSecCD '13: Information Security Curriculum Development Conference* (New York, NY, USA, 2013), InfoSecCD '13, ACM, pp. 20:20–20:26.

[15] DOUPÉ, A., EGELE, M., CAILLAT, B., STRINGHINI, G., YAKIN, G., ZAND, A., CAVEDON, L., AND VIGNA, G. Hit 'em Where it Hurts: A Live Security Exercise on Cyber Situational Awareness. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)* (Orlando, FL, December 2011).

[16] FREDRICKS, J. A., BLUMENFELD, P. C., AND PARIS, A. H. School engagement: Potential of the concept, state of the evidence. *Review of educational research 74*, 1 (2004), 59–109.

[17] GETTINGER, M., AND SEIBERT, J. K. Best practices in increasing academic learning time. *Best practices in school psychology IV 1* (2002), 773–787.

[18] GORENC, B. Pwn2own 2017 at cansecwest. https://www.zerodayinitiative.com/blog/2017/3/23/pwn2own-2017-an-event-for-the-ages, March 2017.

[19] HAMARI, J., KOIVISTO, J., AND SARSA, H. Does gamification work?–a literature review of empirical studies on gamification. In *47th Hawaii International Conference on System Sciences (HICSS)* (Hawaii, 2014).

[20] HARMON, T. Cyber Security Capture The Flag (CTF): What Is It? https://blogs.cisco.com/perspectives/cyber-security-capture-the-flag-ctf-what-is-it, 2016.

[21] The 2016-2017 iCTF DDoS. https://ictf.cs.ucsb.edu/pages/the-2016-2017-ictf-ddos.html.

[22] The iCTF Framework. https://github.com/ucsb-seclab/ictf-framework.

[23] JALABI, R. OPM hack: 21 million people's personal information stolen, federal agency says. *The Guardian* (July 2015).

[24] JARIWALA, S., CHAMPION, M., RAJIVAN, P., AND COOKE, N. J. Influence of Team Communication and Coordination on the Performance of Teams at the iCTF Competition. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (2012).

[25] MARTINI, B., AND CHOO, K.-K. R. Building the next generation of cyber security professionals. In *22nd European Conference on Information Systems (ECIS 2014)* (Tel Aviv, Israel, May 2014).

[26] MULLINS, B. E., LACEY, T. H., MILLS, R. F., TRECHTER, J. E., AND BASS, S. D. How the cyber defense exercise shaped an information-assurance curriculum. *IEEE Security Privacy 5*, 5 (Sept 2007), 40–49.

[27] PHAM, C., TANG, D., CHINEN, K.-I., AND BEURAN, R. Cyris: A cyber range instantiation system for facilitating security training. In *Proceedings of the Seventh Symposium on Information and Communication Technology* (New York, NY, USA, 2016), SoICT '16, ACM, pp. 251–258.

[28] PRINCE, M. Does active learning work? a review of the research. *Journal of engineering education 93*, 3 (2004), 223–231.

[29] RUEF, A., HICKS, M. W., PARKER, J., LEVIN, D., MAZUREK, M. L., AND MARDZIEL, P. Build It, Break It, Fix It: Contesting Secure Development. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)* (2016).

[30] 2015 Global Cybersecurity Status Report. https://www.isaca.org/cyber/documents/Cybersecurity-Status-Report_ifg_Eng_0115.pptx, 2017.

[31] SHOSHITAISHVILI, Y., INVERNIZZI, L., DOUPÉ, A., AND VIGNA, G. Do You Feel Lucky? A Large-Scale Analysis of Risk-Rewards Trade-Offs in Cyber Security. *ACM Symposium on Applied Computing* (March 2014).

[32] SUBY, M. The 2015 (isc) 2 global information security workforce study. *Frost & Sullivan in partnership with Booz Allen Hamilton for ISC2* (2015).

[33] VABLASCO-ARCAS, L., BUIL, I., HERNANDEZ-ORTEG, B., AND SESE, F. J. Using Clickers in Class. the Role of Interactivity, Active Collaborative Learning and Engagement in Learning Performance. In *Computers and Education*, vol. 62. Pergamon Press, March 2013, pp. 102–110.

[34] VAMVOUDAKIS, K., HESPANHA, J., KEMMERER, R., AND VIGNA, G. Formulating Cyber-Security as Convex Optimization Problems. In *Control of Cyber-Physical Systems*, vol. 449 of *Lecture Notes in Control and Information Sciences*. Springer, July 2013, pp. 85–100.

[35] VIGNA, G. Teaching Hands-On Network Security: Testbeds and Live Exercises. *Journal of Information Warfare 3*, 2 (February 2003), 8–25.

[36] VIGNA, G. Teaching Network Security Through Live Exercises. In *Proceedings of the Third Annual World Conference on Information Security Education (WISE)* (Monterey, CA, June 2003), C. Irvine and H. Armstrong, Eds., Kluwer Academic Publishers, pp. 3–18.

[37] VIGNA, G., BORGOLTE, K., CORBETTA, J., DOUPÉ, A., FRATANTONIO, Y., INVERNIZZI, L., KIRAT, D., AND SHOSHITAISHVILI, Y. Ten Years of iCTF: The Good, The Bad, and The Ugly. In *Proceedings of the USENIX Summit on Gaming, Games and Gamification in Security Education (3GSE)* (San Diego, CA, August 2014).

[38] The Conscience of a Hacker. `https://en.wikipedia.org/wiki/Hacker_Manifesto`, 2017.

[39] WILLINGHAM, D. T. Critical thinking: Why is it so hard to teach? *Arts Education Policy Review 109*, 4 (2008), 21–32.