Hit ’em Where it Hurts: A Live Security Exercise on Cyber Situational Awareness

Adam Doupé, Manuel Egele, Benjamin Caillat, Gianluca Stringhini,
Gorkem Yakin, Ali Zand, Ludovico Cavedon, and Giovanni Vigna
University of California, Santa Barbara
{adoupe, maeg, benjamin, gianluca, gyakin, zand, cavedon, vigna}@cs.ucsb.edu

ABSTRACT

Live security exercises are a powerful educational tool to motivate students to excel and foster research and development of novel security solutions. Our insight is to design a live security exercise to provide interesting datasets in a specific area of security research. In this paper we validated this insight, and we present the design of a novel kind of live security competition centered on the concept of Cyber Situational Awareness. The competition was carried out in December 2010, and involved 72 teams (900 students) spread across 16 countries, making it the largest educational live security exercise ever performed. We present both the innovative design of this competition and the novel dataset we collected. In addition, we define Cyber Situational Awareness metrics to characterize the toxicity and effectiveness of the attacks performed by the participants with respect to the missions carried out by the targets of the attack.

1. INTRODUCTION

In recent years, security attacks have become increasingly wide-spread and sophisticated. These attacks are made possible by vulnerable software, poorly configured systems, and a lack of security awareness and education of end users.

While a large portion of the security research efforts are focused on developing novel mechanisms and policies to detect, block, and/or prevent security attacks, there is also the need for the development of novel approaches to educate those who create the computer infrastructure, as well as those who use it everyday.

This is an often-overlooked aspect of computer security, but a critical one. Almost all sophisticated, widely deployed, security mechanisms can be made useless by luring an unsuspecting user (or a developer) into performing actions that, eventually, will compromise the security of their environment. A clear example of the popularity of these attacks is the proliferation of fake anti-virus scams, in which users who are not technically savvy are conned into installing a Trojan application [17].

Security education can be performed at different levels to reach different segments, from everyday Internet users, to high school students, to undergraduate and graduate students. Recently, competition-based educational tools have become popular in graduate and undergraduate education, as competition among students fosters creativity, innovation, and the desire to excel.

Previous work has described traditional “capture the flag competitions” [19, 20], and, more recently, new designs for this type of competition [2]. The development of new designs improved the competition and forced the participants to analyze and understand unfamiliar, complex sets of interdependent components, similar to those that are part of real-life networks and malware infrastructures.

Our novel insight is that these competitions, can, in addition to their educational value, provide interesting datasets that can be used in research. To validate this idea we designed and developed a novel security competition based on the concept of *Cyber Situational Awareness* (described in Section 2). The competition is called the iCTF (international Capture the Flag) and was carried out on December 3rd, 2010, involving 72 teams and 900 students, making it the largest live educational security exercise ever performed.

This paper presents the design of the competition, the data that was collected, and the lessons learned. The data is the first publicly available dataset that explicitly supports research in Cyber Situational Awareness.

In summary, this paper adds the following contributions:

- We describe the design and implementation of a novel computer security competition, whose goal is to not just foster computer security education, but to create a Cyber Situational Awareness dataset.
- We analyze the collected dataset and discuss its use in Cyber Situational Awareness research, introducing a novel metric that characterizes the effectiveness of attacks with respect to a specific mission.
- We discuss the lessons learned from the competition, and we provide suggestions to other educators that might implement similar competitions.

2. BACKGROUND AND HISTORY

In this section, we provide background on two of the most important aspects of this paper: the design and execution of live security competitions, and the concepts associated with Cyber Situational Awareness.

2.1 Live Security Competitions

Security challenges have been a way to attract the interest of security researchers, practitioners, and students. Live security challenges add a real-time factor that supports deeper involvement and introduces the "crisis factor" associated with many real-life security problems: "*something bad is happening right now and has to be taken care of.*"

There have been a number of live security challenges, but the best-known competition is DefCon's Capture The Flag (CTF). This competition started with a simple design, where a host with vulnerable services was made available to the participants, who would attack the target concurrently. Whoever was able to break a service and steal the flag first, obtained the points associated with that service. The original design was changed in 2002. In this edition of DefCon's CTF, the participating teams received an identical copy of a virtualized system containing a number of vulnerable services. Each team ran their virtual machine on a virtual private network (VPN), with the goal of maintaining the service's availability and integrity whilst concurrently compromising the other teams' services. Since each team had exactly the same copy of the services, the participants had to analyze the services, find the vulnerabilities, patch their own copies, and break into the other teams' services and steal the associated flags. Every other DefCon CTF following 2002 used more or less the same design [4].

Even though DefCon's CTF was designed to test the skills of hackers and security professionals, it was clear that the same type of competition could be used as an educational tool. One of the major differences between the iCTF and DefCon's CTF is that the iCTF involves educational institutions spread out across the world, where the DefCon CTF allows only locally-connected teams. Therefore, DefCon requires the physical co-location of the contestants thus constraining participation to a limited number of teams. By providing remote access, the iCTF allows dozens of remotely located teams to compete.

The iCTF editions from 2003 to 2007 were similar to the DefCon CTF: the participants had to protect and attack a virtualized system containing vulnerable services [20]. In 2008 and 2009, two new designs were introduced: in 2008, the competition was designed as a "treasure hunt," where the participants had to sequentially break into a series of hosts; in 2009, the competition focused on drive-by-download attacks, and the targets were a large pool of vulnerable browsers [2]. The iCTF inspired other educational hacking competitions, e.g., CIPHER [12] and RuCTF [18].

Recently, a different type of competition has received a significant amount of attention. In the Pwn2Own hacking challenge [13] participants try to compromise the security of various up-to-date computer devices such as laptops and smart phones. Whoever successfully compromises a device, wins the device itself as a prize. This competition is solely focused on attack, does not have an educational focus, and does not allow any real interaction amongst the participants who attack a single target in parallel.

Another interesting competition is the Cyber Defense Exercise (CDX) [1,10,14], in which a number of military schools compete in protecting their networks from external attackers. This competition differs from the UCSB iCTF in a number of ways. First, the competition's sole focus is on defense. Second, the competition is scored in person by human evaluators who observe the activity of the participants, and score them according to their ability to react to attacks. This evaluation method is subjective and requires a human judge for each team thus rendering it impractical in a large-scale on-line security competition.

The 2010 iCTF differed from the 2009 iCTF [2] in the following way: we realized that a live security exercise could be structured to create a dataset to enable security research. We utilized this idea in the 2010 iCTF by creating a Cyber Situational Awareness security competition that would generate a useful Cyber Situational Awareness dataset.

2.2 Cyber Situational Awareness

Cyber Situational Awareness (CSA) is an extension of traditional Situational Awareness (SA) to computer networks. The idea behind SA is that by analyzing the surrounding environment and putting perceived events into the context of the current mission, it is possible to improve decision-making. In the cyber-world, the concept of Situational Awareness includes the concept of mission awareness, which is the analysis of network events with respect to the mission being carried out by a particular organization.

One of the most important ideas behind CSA is that not all attacks have the same impact. The relevance of an attack is determined by the importance of the target with respect to a specific mission and a specific moment in time. For example, an attack against an FTP server could be harmless if the server is not a necessary component for the currently executing mission(s) in the next, say, eight hours, because within that time frame the server could be fixed/cleaned and it could be available when needed. Instead, consider an attack against a VoIP router when a strategic meeting must use that particular piece of infrastructure. The attack will directly impact the mission being carried out, and might impose delays or cause the mission to fail.

There are several challenges in CSA. First of all, it is difficult to correctly model missions. In many cases, organizations and companies are not even aware of their cyber-missions. Usually, identifying cyber-missions is easier in environments where repetitive tasks are performed cyclically. For example, banks have well-defined missions with tasks that must be carried out in specific sequences (e.g., closing balances, reconcile balance sheets) and must be performed within a certain time limit (e.g., midnight of the current day). Another example is military systems, where cycles of observation/analysis/operation phases are carefully followed, with precise time frames and clear dependencies.

In all these cases, one must choose a particular format to precisely describe a mission. A simple solution is to use Gantt charts [3], which clearly represent the duration and dependency of different tasks. For the cyber-missions described in this paper, we used *Missionary*, a Petri net [11] based formalism we created which extends the basic Petri net model with timing and code fragments associated with transitions and states. In this formalism, the tasks are represented by the states of the Petri net. A token in a state characterizes an active instance of the task. A task terminates when a token is removed from the corresponding state as a side-effect of the firing of a transition. Analogously, a task starts when a token is created in a state as the side-effect of the firing of a transition. Peterson [11] has a detailed description of Petri nets and their extensions.

Another challenge in CSA is to represent the dependency between cyber-missions and both the *human actors* and *as-*

sets involved in the missions [5]. For the sake of simplicity we do not address the former. For the latter, we used Missionary's service composition formalism, which allows the association of different types of service compositions to a task in a mission. In a nutshell, the formalism allowed us to specify services that were associated with a state in the Petri net, thus creating an association between a mission task and the services necessary to carry out the task.

3. 2010 iCTF

The iCTF competition was held on December 3rd, 2010, and lasted from 09:00 until 17:00, PST.

3.1 Pre-competition setup

Registration for the iCTF competition began a month before the start date. Attempting to alleviate the VPN connection problems that can occur on the day of the competition, we distributed a VMware [21] image along with VPN connection instructions to each team 11 days before the competition. The VMware image was meant as an example of the type of VMware image that would be used for the competition. We took particular care in making sure that the teams solved their connectivity problems well in advance, so that they could focus on the competition.

3.2 Story

The theme of the iCTF competition was "Mission awareness in state-sponsored cyberwar." The following text was given to the teams the day before the competition:

> *The country of Litya has become a major center for illegal activities of all kinds. The country is ruled by the ruthless dictator Lisvoy Bironulesk, who has pioneered the use of large malware infrastructures in order to support Litya's economy. Recently, he has claimed that Litya has "a botnet in every country."*
>
> *His complete disregard for international laws, his support of banking fraud and phishing scams, together with his well-known taste for underage girls has finally brought the attention of the international community into his shady dealings.*
>
> *Enough is enough. Now, the affected nations have decided to strike back. Spies who infiltrated Litya's corrupt administration have leaked plans of the most critical missions carried out in the country. These plans appear to describe the various activities of each mission, their ordering and timing, and their dependency on particular services.*

In this scenario, each team represented a country with the common goal of dismantling Litya's infrastructure, thus ending Bironulesk's reign. In addition to this text, the teams were given a number of images that described the various "missions" carried out by Litya. One of the missions is shown in Figure 1.

3.3 Competition Description

At a high level, the competition was designed to force the teams to exploit services at specific times, when they are most needed by Litya, thus emulating a Cyber Situational Awareness scenario. The teams had to access the services, first by bribing Litya's administrators, then by keeping their VMware image connected to a "mothership." If the teams generated an intrusion detection system alert, they were blocked from the network for a fixed amount of time.

3.3.1 Scoring

There were two types of scores: money and points. The team with the highest points won the competition, thus points were more important than money. Points were acquired by exploiting services at the correct time. However, if a team did not have any money, they would be shut off from the network and not be able to score any points. In addition to starting the competition with 1000 in money and zero points, each team earned money by solving challenges.

3.3.2 Firewall and IDS

A substantial innovation introduced in the iCTF was creating an intrusion prevention system (IPS) by connecting the Snort [16] intrusion detection system to the firewall. If Snort detected an intrusion attempt (alert) from a team on traffic directed towards Litya's services, the offending team was shut off from the network for ten minutes. The team would either have to wait until connectivity was allowed again or spend money bribing Litya's network administrators to gain access to the network for a certain amount of time. The teams had full knowledge of the Snort version and configuration, thus, they could predict if their traffic would generate an alert. Connecting Snort to the firewall forced the teams to come up with novel ways to circumvent an IDS.

3.3.3 Botnet

Bribing Litya's network administrators for access opened up the network for a limited amount of time (proportional to the amount of money used to bribe). To remain connected to the network, the teams needed to run a bot, which we provided 2 hours before the competition. This bot would connect to a mothership every 30 seconds and while the bot was connected to the mothership it would drain money from the team at a rate of 6 money per minute. As long as the bot remained connected to the mothership, the team had money, *and* the team didn't generate any Snort alerts, they could access the services. The two means of connecting to the network (bot connection or bribing) forced teams to make strategic decisions about when to connect, when to attack, how to attack, and when to bribe (spend money). These strategic decisions added another dimension to the iCTF competition: Teams had to decide the proper allocation of money (bot connection or bribing) to maximize their access to the network and thus maximize points.

Like a real-world *bot*, these machines were "compromised" and had 3vilSh3ll [15], a backdoor bind connect, running on port 8000. This allowed anyone to connect on port 8000, supply the password: `hacked`, and obtain a root shell. The idea was to encourage teams to be careful about their firewall, and force them to defensively select what traffic they allowed into their network.

3.3.4 Challenges

To gain money to bribe the Litya network administrators, as well as allow the mothership to steal money and remain connected to the network, teams needed to solve challenges. We created 33 challenges to provide multiple ways to earn money, but also to offer opportunities to test and improve

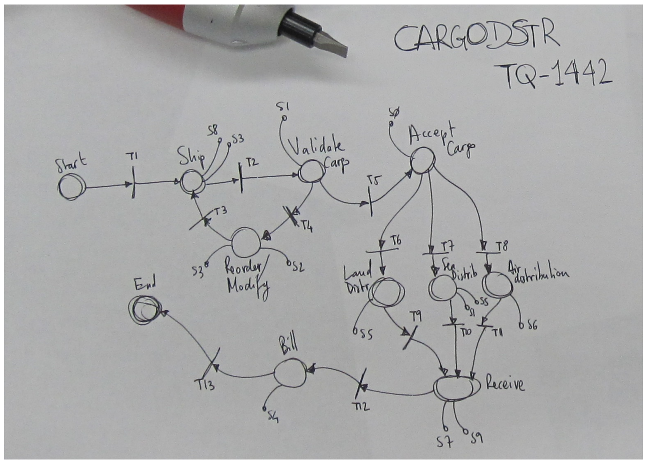


Figure 1: CARGODSTR mission that was distributed to the teams.

different skills, from cryptanalysis to forensics, program and network analysis.

3.3.5 Scoreboard

In a capture the flag competition, a scoreboard showcasing the current status and ranking of each team is vital. For the iCTF competition, we also needed to show the connection status of each team; if they were connected to the network, and why they were disconnected: Either from an IDS alert, lack of a bot connection, or lack of money. The scoreboard also showed the history of each team's money and points. The scoreboard is a very important piece of the infrastructure, because it provides immediate feedback to the teams about the success (or failure) of their attacks. Unfortunately, we had some glitches in our scoreboard that we will discuss in Section 5.

3.3.6 Missions

The day before the competition each team received an email containing a link to four pictures. Each picture contained a description of a Cyber Situational Awareness mission, in the form of a hand-drawn Petri net. Figure 1 shows one of these missions, the CARGODSTR mission. In the Petri net, all of the transitions were named (although not unique across the missions and even within some missions), as were most of the states. Some of the states were associated with one or more of the 10 services (S0-S9). For example, the "Receive" state in the lower right of Figure 1 is associated with services S7 and S9. The four Petri net missions given to the teams are graphically shown in Figure 2.

A service that we ran executed the Petri nets by inserting a token in each of the "Start" states, and running each Petri net separately. At each time-step, for each of the missions, one of the eligible transitions (a transition where all inputs had tokens) was randomly chosen to fire. Then, the token was consumed on all the inputs to the chosen transitions, and a token was placed on all the outputs. The four chosen transitions (one from each mission) were leaked to the teams after each time-step. Then, after each mission was executed, the service suspended for a random amount of time between one and two minutes, and the process repeated until the end of the competition. If a token was in an "End" state, or the network became stuck (no eligible transitions) then the network was reset.

For example, running the CARGODSTR mission, Figure 1 and Figure 2a, would involve first placing a token in its "Start" position. As `T1` is the only eligible transition to fire, it is chosen, and this information is leaked to the teams. The token moves from "Start" to "Ship." Because services `S8` and `S3` are associated with the "Ship" state, and it has a token, they are active. After a pause of one to two minutes, the next time-step occurs. Once again, there is only one eligible transition, `T2`. It is chosen, and the token on "Ship" moves to "Validate Cargo." Now, services `S8` and `S3` are no longer active, but service `S1` becomes active. After another pause, the process repeats, but with two eligible transitions, `T4` and `T5`. One of these is randomly chosen, say `T5`, and the token moves. This process repeats until the end of the competition. A visualization of the execution of all four missions throughout the iCTF competition is available[1].

From this example of the execution of the CARGODSTR mission, the teams received the sequence: `T1 T2 T5`. With only this information, they had to reverse engineer the state of the mission to find out which services were active. The teams would then attack only the active services. In the CARGODSTR mission, this is simple because the transitions are unique, however this is not the case for all the missions, as shown in the SEDAFER mission (Figure 2d).

3.3.7 Flags

A flag was a sequence of hexadecimal values prefixed with `FLG` and was specific to a service. Flags were not directly accessible: a service must be compromised to access the associated flag. Therefore, flags are used by the participants as proof that, at a certain time, they were able to compromise a specific service. On each step of the service that executed the Petri nets, a new flag specific to each service was distributed to the corresponding service. Each flag contained (cryptographically) the service that it belonged to, the state of the service (active or not), and a timestamp signifying when the flag was created. Thus, when a flag was submitted by a team, the flag submission service had all the necessary information to determine the flag's validity (flags were valid for 5 minutes).

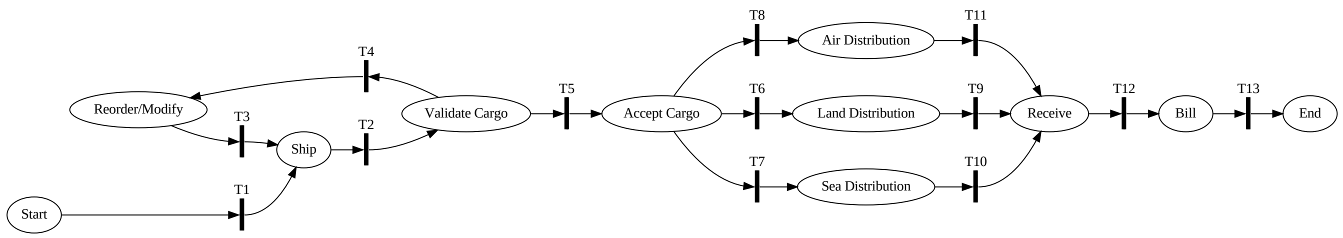
3.3.8 Vulnerable Services

There were 10 services in the iCTF, each service could be exploited only once per Petri net execution round; exploiting a service when it was not active resulted in an equal amount of negative points. Thus, to win the competition it was essential to understand and follow the missions. Table 1

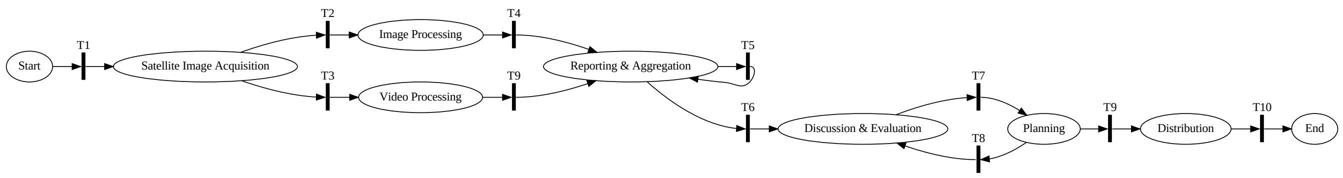
Service	Vulnerability
LityaBook	Cross-Site Scripting
LityaHot	Session Fixation
icbmd	Foam Rocket Firing
StormLog	Off-By-One Overflow
StolenCC	Perl's `open` abuse
SecureJava	Broken Crypto
IdreamOfJeannie	Java JNI Off-By-One Error
WeirdTCP	TCP IP Spoofing
MostWanted	SQL-Injection
OvertCovert	Format String

Table 1: Brief description of vulnerable services.

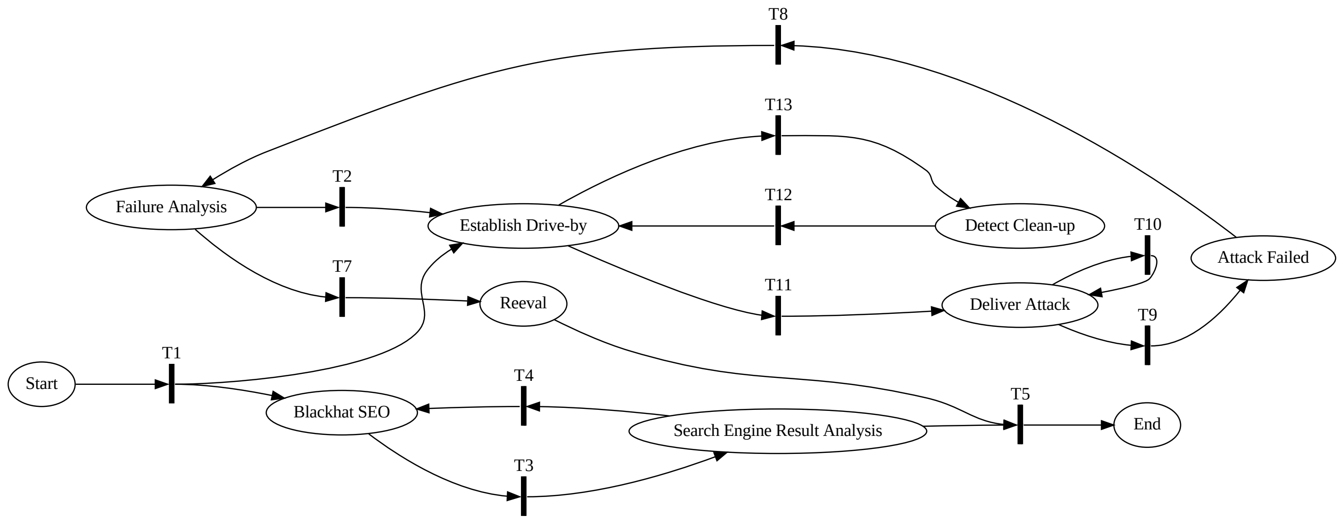
[1] `http://ictf.cs.ucsb.edu/data/ictf2010/final.gif`



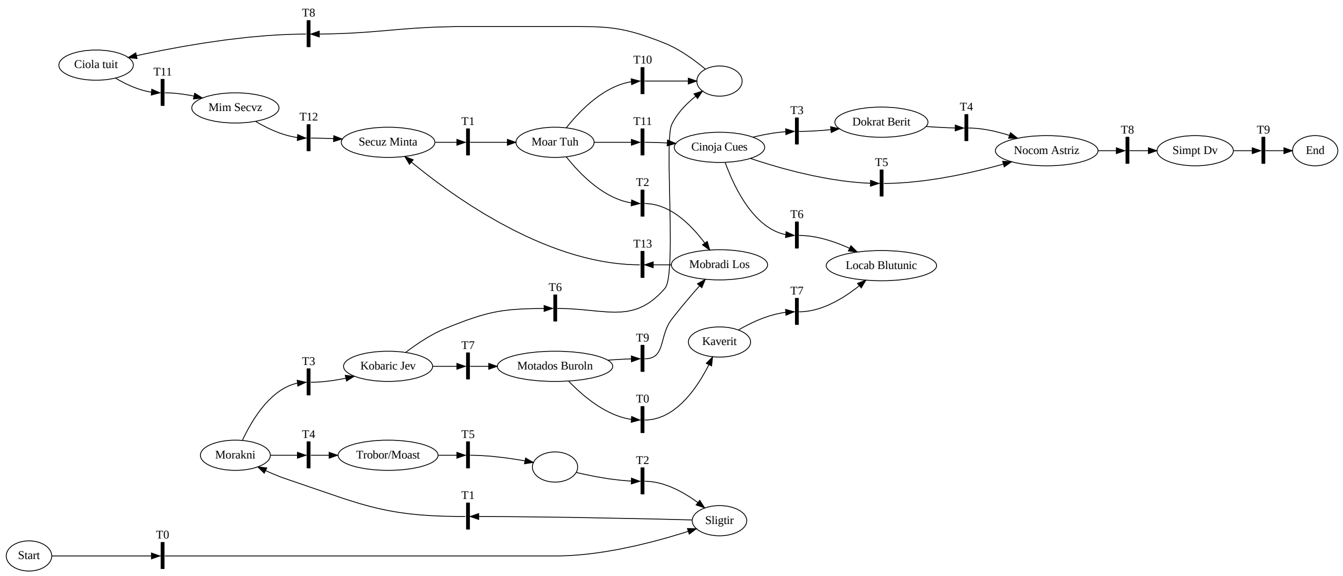
(a) CARGODSTR



(b) COMSAT



(c) DRIVEBY



(d) SEDAFER

Figure 2: Graphical representation of the missions given to the teams. The teams were actually given formats similar to Figure 1. Not shown here are the associations of the services to states in the Perti nets.

briefly summarizes the services. We direct the interested reader to Appendix A for an extended description of the services.

4. DATA ANALYSIS

In addition to being an excellent learning exercise for the teams involved, a security competition, if properly designed, can be a great source of data that is difficult to obtain in other contexts. In the iCTF competition, we created a game scenario to generate a Cyber Situational Awareness dataset.

Traffic collected during a security competition can be easier to analyze than real-world traffic, because there is more information about the network and participants in the competition. For example, all teams are identified, the vulnerable services are known, and there is no "noise traffic." Of course, a dataset collected in such controlled conditions also suffers from a lack of realism and is limited in scope. Nonetheless, the data collected during this competition is the first publicly available dataset that allows researchers to correlate attacks with the missions being carried out.

The iCTF competition generated 37 gigabytes of network traffic and complete information about services broken, challenges solved, flags submitted, bribes paid, IDS alerts, and bot connections. This data is made freely available[2].

As this is the first Cyber Situational Awareness dataset, many possibilities exist for its use in Situational Awareness research. One example would be using the dataset to train a host-based CSA intrusion detection system that could use more restrictive rules for a rule-based system (or tighter thresholds in an anomaly-based system) when a service is critical to a mission. One can also think of extending a host-based IDS to a network CSA intrusion detection system that understands not only the criticality of the services, but also their dependencies and relationships. Another example is the visualization of a network's activity with CSA in mind that helps a system administrator know which services are currently critical and which will become critical soon, helping them defend their network.

The firewall, bribing, bot, and money/points system can be viewed in a game theory light. The teams had to decide on the best way to allocate a scarce resource (money) to access the network and potentially win the game. The teams could perform any combination of bot connection and/or bribing to access the network. Further research could investigate how the choice of resource allocation affected each team's final result.

4.1 Description of Results

One problem with designing and implementing a novel competition is that teams may not understand the rules. This was a concern during the design of the iCTF competition. We worried that the novel aspects of the competition, especially the Petri net mission model, would be too complex for the teams to understand. However, when the first flags were submitted at 13:29, and subsequently when teams started submitting flags only for active services, it became apparent that many teams understood the competition.

Of the 72 teams, 39 submitted a flag, with 872 flags submitted in total. 48% may seem like a low number, however this means that almost half the teams broke at least one service. Many of the 39 teams submitted multiple flags, indicating that they understood the Petri net mission model.

[2]http://ictf.cs.ucsb.edu/data/ictf2010/

Service	Total	Active	Inact.	% Inact.	Teams	Flags/Team
MostWanted	680	562	118	17	38	17.895
OvertCovert	97	82	15	15	6	16.167
IdreamOf.	49	37	12	24	6	8.167
WeirdTCP	24	23	1	4	2	12
LityaBook	16	12	4	25	3	5.333
icbmd	5	3	2	40	1	5
StolenCC	1	0	1	100	1	1

Table 2: Flags submitted per service.

At 17:00, "Plaid Parliament of Pwning" (PPP) of Carnegie Mellon University, took first place with 24,000 points. PPP submitted a total of 93 flags, with only 3 inactive flags (thus generating negative points), by compromising IdreamOfJeannie, MostWanted, and OvertCovert. Because PPP was able to compromise three services as well as understand the Petri net model (as evidenced by the submission of only three negative flags), they won first place.

Overall the teams exploited 7 of the 10 services: icbmd, IdreamOfJeannie, LityaBook, MostWanted, OvertCovert, StolenCC, and WeirdTCP. We believe this is because we underestimated the difficulty of the other 3 services. SecureJava and StormLog required a complex, multi-step process that proved too difficult for the teams to exploit. The teams also had trouble understanding the steps involved to exploit the session fixation vulnerability in LityaHot.

Table 2 describes the number of flags submitted for each service. The "Total" column is the total number of flags submitted for the service, "Active" and "Inact." are the number of flags that were submitted when a service was active or inactive. "% Inact." is the percent of flag submissions when the service was inactive. "Teams" shows the number of teams that submitted flags for the service and "Flags/Team" shows the average number of flags submitted per team.

MostWanted was the most exploited service, with 680 total flags submitted, followed by OvertCovert, with 97 flags submitted. It is clear that we did not estimate the difficulty of the services correctly, and, as evidenced by the number of teams that broke it, MostWanted was the easiest. Because the teams did not know the difficulty of the services, some luck is involved when teams decide which service to analyze first.

When we decided to create a complex competition, we knew that not every team would have the skills, experience, and luck to exploit a service and understand the Petri net mission model. However, we included 33 challenges in the competition of varying levels of difficulty and needing various skills to solve. We knew from past experience that even if a team couldn't exploit a service or understand the Petri net model of the missions, they would at least learn from (and enjoy) solving challenges. In fact, 69 out of 72 teams solved at least one challenge. Thus, even if a team was unable to exploit a service, they solved a challenge and hopefully had fun or learned something while competing in the iCTF.

4.2 Network Analysis

A benefit of designing a security competition is the ability to create an environment that allows for the testing of models and theories. By focusing the iCTF on Cyber Situational Awareness, we were able to create and evaluate Situational Awareness metrics. These metrics are applicable to many aspects of CSA. We introduce *toxicity* and *effectiveness*, which are explained in the rest of this section.

First, we define three functions: $C(s,t)$, $A(a,s,t)$, and $D(s,t)$, each with a range of [0, 1]. Every function is specific to a service, s, and $A(a,s,t)$ represents an attacker, a.

$C(s,t)$ represents how critical a service, s, is with respect to time for a specific mission or set of missions. A value of 1 means that the service is very critical, while 0 means that the service is not critical.

$A(a,s,t)$ represents an attacker's, a, activity with respect to a service, s, throughout time. The value of the function is the perceived risk to the mission associated with the service. In most cases, the function has a value of 1 when an attack occurs and a value of 0 when there is no malicious activity. However, other, more complex models could be used (e.g., the type of attack could be taken into account).

$D(s,t)$ represents the damage to any *attacker* for attempting an attack on a service, s, at a given time, t. This function models the fact that every time an attack is carried out, there is a risk to the attacker, e.g., an intrusion detection system might discover the attack, the person using the targeted machine/service might notice unusual activity, etc.

We wish to define a metric, called *toxicity*, that captures how much damage an attacker has caused to a service over a time frame. Intuitively, it is the total amount of havoc the attacker has caused to the mission (or missions) associated with a service. *Toxicity* is calculated by first subtracting the damage to an attacker, $D(s,t)$, from the criticality of the service, $C(s,t)$. The resulting function, with a range of [-1, 1], describes at each point in time how much *any* attacker can profit by attacking at that moment. A negative value indicates that the attacker should not attack at that time.

The previously calculated function is general and has no bearing on a particular attacker. To calculate the damage caused by a specific attacker over time, we take the previously calculated function, $C(s,t) - D(s,t)$, and multiply it by $A(a,s,t)$. The resulting function, with a range of [-1, 1], shows how much damage a specific attacker caused to a given service. To calculate toxicity from this function, for a given time interval, t_1 to t_2, we take the integral of $A(a,s,t) * (C(s,t) - D(s,t))$ with respect to time. Equation (1) shows the calculation of the toxicity metric.

Toxicity is a measure for how much damage an attacker has caused to a given service, and can compare two attackers against the same service to see who did the most damage, however, it is specific to one service, and thus is useless as a comparison between a single attacker attacking multiple services or two attackers attacking different services. We propose *effectiveness* as a measure of how close an attacker is to causing the maximum toxicity possible. Intuitively, it is the ratio of the toxicity caused by an attacker to the toxicity an optimal attacker would cause. We define an optimal attacker as an attacker who attacks whenever $C(s,t)$ - $D(s,t)$ is positive, and this is shown in Equation (2). By substituting the optimal attacker in Equation (1) for $A(a,s,t)$, we obtain the formula for maximum toxicity, given in Equation (3). Taking the ratio of toxicity to maximum toxicity gives effectiveness, shown in Equation (4).

Toxicity, effectiveness, and $C(s,t)$, $A(a,s,t)$, and $D(s,t)$ can be used in future Cyber Situational Awareness research. By using the ideas presented here, an IDS could predict the behavior of an optimal attacker. Other tools could enable a network defender to perform "what-if" scenarios, seeing what would happen by increasing the damage to an attacker (e.g., by getting a new IDS), versus decreasing the criticality of the service (e.g., by getting a new server to perform the same function).

$$\mathit{Toxicity}(a,s,t_1,t_2) = \int_{t_1}^{t_2} A(a,s,t) * (C(s,t) - D(s,t))\,\mathrm{d}t \tag{1}$$

$$\mathit{OptimalAttacker}(s,t) = \begin{cases} 1 & \text{if } C(s,t) - D(s,t) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$\mathit{MaxToxicity}(s,t_1,t_2) = \int_{t_1}^{t_2} \mathit{OptimalAttacker}(s,t) * (C(s,t) - D(s,t))\,\mathrm{d}t \tag{3}$$

$$\mathit{Effectiveness}(a,s,t_1,t_2) = \frac{\mathit{Toxicity}(a,s,t_1,t_2)}{\mathit{MaxToxicity}(s,t_1,t_2)} \tag{4}$$

The definitions of *toxicity* and *effectiveness* are general and apply to any arbitrary functions $C(s,t)$, $A(a,s,t)$, and $D(s,t)$. However, we constructed the iCTF competition so that we could measure and observe these functions and ensure they are valid metrics. We expected the higher ranked teams to show high toxicity and effectiveness for the services they broke.

The criticality, $C(s,t)$, of each service was defined in the following way: the function takes the value 1 when the service is active, and 0 when the service is inactive. Figure 3 shows the criticality graph for the most exploited service: MostWanted. When the function has a value of 1, one of the missions is in a state associated with the MostWanted service, otherwise the function has a value of 0. Note that for these and all the rest of the graphs of the competition, the X-axis is time, and starts at 13:30 PST, when the first flag was submitted, and ends at 17:00 PST, which was the end of the competition.

In our analysis, we define the damage to the attacker, $D(s,t)$, as the complement of the criticality graph, because if an attacker attacked a service when it was not active, they would get an equal amount of negative points. The damage graph alternates between 0 and 1, becoming 1 when the criticality is 0 and 0 when the criticality is 1. In our analysis, the criticality and damage functions are related as a byproduct of our design; however our definitions of toxicity and effectiveness do not depend on this; criticality and damage can be arbitrary and independent functions.

In order to calculate the toxicity of Plaid Parliament of Pwning against the various services, we must first calculate $A(a,s,t) * (C(s,t) - D(s,t))$ (note that this function has a range of [-1, 1]. Negative values in this context denote flags submitted when a service was inactive). This is shown in Figure 4 for the service MostWanted, and Figure 5 for the service OvertCovert. As can be seen in Figure 4, PPP did not attack at the incorrect time for the MostWanted service, but submitted several incorrect flags for OvertCovert, as evidenced by the negative values in Figure 5.

Toxicity is calculated by taking the integral of this function between 13:30 and 17:00 PST. However, since the time in-between each flag change is a random value between 60 and 120 seconds, and a team is able to exploit the service only once per flag change, we simplified the time between

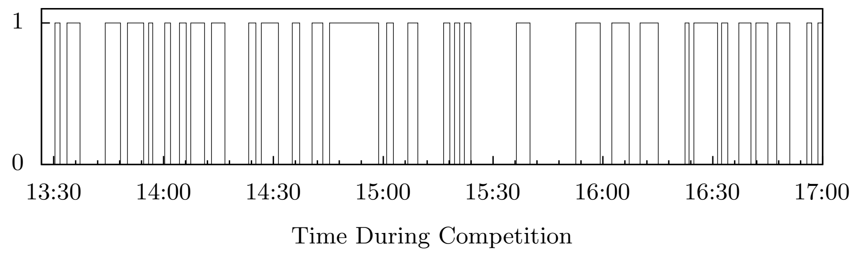


Figure 3: $C(s,t)$ of the service MostWanted.

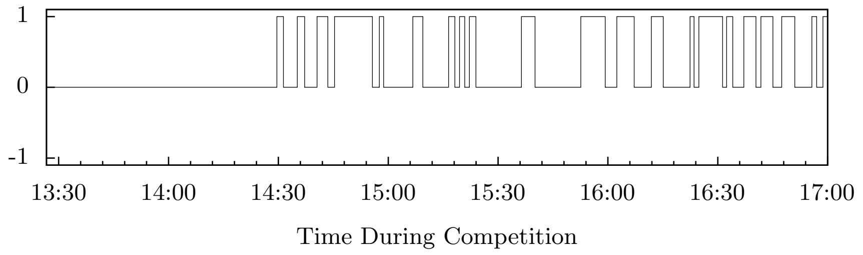


Figure 4: $A(a,s,t) * (C(s,t) - D(s,t))$ of team PPP against the service MostWanted.

flags as 1, which returned a round number for the toxicity metric. In the general case, however, the amount of time a service is critical is very important for calculating toxicity and should not be oversimplified. Because the criticality of our services changed at discrete intervals, we are able to make this simplification without adversely affecting our results.

Table 3 shows the toxicity and effectiveness of the top 5 teams for each of the services that were successfully exploited. The results are as we expected; many of the most effective teams placed high in the final rankings. The first place team, PPP, team #113, was not only the most effective for three different services: IdreamOfJeannie, MostWanted, and OvertCovert, but, also, with 65% effectiveness on MostWanted, had the highest effectiveness of any team. PPP's dominance is apparent because they did not just break three services, but they were also highly effective. The second place team, 0ld Eur0pe (team #129), was the second most effective at IdreamOfJeannie and third most effective at MostWanted.

5. LESSONS LEARNED

For this edition of the iCTF competition, we tried to capitalize on our previous experience by learning from mistakes of years past. However, we may hope to the contrary, we are still human: we made some mistakes and learned new lessons. We present them here so that future similar competitions can take advantage of what worked and avoid repeating the same mistakes.

5.1 What Worked

The pre-competition setup worked extremely well. Having the teams connect to the VPN and host their own VMware bot image was helpful in reducing the support burden on the day of the competition, where the time is extremely limited.

In the past, having a complex competition frustrated many teams and caused them to spend a substantial amount of time trying to figure out the competition instead of actually competing. To combat this, we released details about the structure of the game, the Petri net models of the missions, and the Snort configuration in advance. We hoped that this would give teams the opportunity to come to the competition well-prepared. Another advantage in giving advance notice is that it rewards teams who put in extra time outside of the eight hours of the competition. This is important, as the larger part of the education process is actually associated with the preparation phase, when students need to become familiar with different technologies and brainstorm possible attack/defense scenarios.

Another positive feedback we received through informal communication was that the theme of the competition was clear and consistent. The iCTF competition has always had a well-defined background story, which supports understanding and provides hints on how to solve specific challenges. People explicitly appreciated the effort put into creating a consistent competition environment and complained about competitions that are simply a bundle of vulnerable services to exploit.

From the comments of the players, it was clear that a substantial amount of effort was put into preparing and developing the right tools for the competition. This is one of the most positive side-effects of the participation in this kind of live exercises. Having to deal with unknown, unforeseen threats forces the teams to come up with general, configurable security tools that can be easily repurposed once the focus of the competition is disclosed. The continuous change in the iCTF design prevents the "overfitting" of such tools to specific competition schemes.

In general, through the past three years we found that radical changes in the competition's design helped leveling the playing field. Although the winning teams in the 2008,

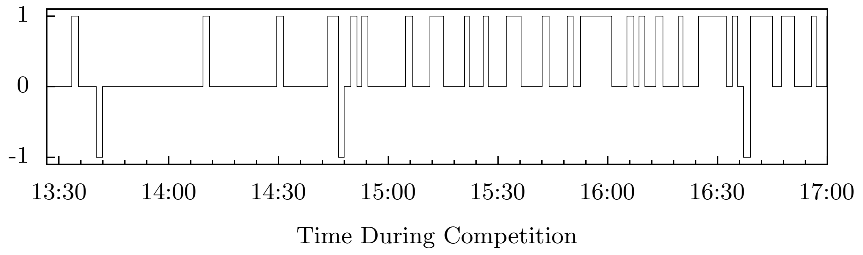


Figure 5: $A(a, s, t) * (C(s, t) - D(s, t))$ of team PPP against the service OvertCovert.

Service	Team	Toxicity	Effectiveness
icbmd	126	3	0.03896
icbmd	124	1	0.01298
IdreamOfJeannie	113	14	0.23728
IdreamOfJeannie	129	12	0.20338
IdreamOfJeannie	123	10	0.16949
IdreamOfJeannie	111	2	0.03389
IdreamOfJeannie	128	-6	-0.10169
LityaBook	149	8	0.11428
LityaBook	166	5	0.07142
LityaBook	150	-1	-0.01428
LityaBook	137	-2	-0.02857
StolenCC	123	1	0.02040
StolenCC	105	1	0.02040
StolenCC	152	0	0.0
MostWanted	113	42	0.65625
MostWanted	114	40	0.625
MostWanted	129	36	0.5625
MostWanted	105	34	0.53125
MostWanted	152	30	0.46875
OvertCovert	113	36	0.48648
OvertCovert	131	16	0.21621
OvertCovert	123	10	0.13513
OvertCovert	117	9	0.12162
OvertCovert	127	2	0.02702
WeirdTCP	156	13	0.23214
WeirdTCP	105	6	0.10714

Table 3: Top 5 most effective teams per service.

2009, and 2010 editions were still experienced groups, teams of first-time competitors placed quite high in the ranking. This was possible because we intentionally did not disclose in advance to the teams the nature of these new competitions. Many "veteran" teams expected a standard CTF and were surprised to learn that this was not the case. Of course, it is hard to keep surprising teams, as designing new competitions requires a substantial amount of work. However, it is arguable that this type of competition is inherently easier for novice teams to participate in.

Finally, the competition generated a unique, useful dataset that can be used to evaluate cyber situation awareness approaches. This aspect of security competitions cannot be overemphasized, as a well-designed data-capturing framework can provide a wealth of useful data to security researchers.

5.2 What Did Not Work

LityaLeaks, the part of the infrastructure used to distribute the fired transitions of the Petri nets, as well as various hints and clues about services and challenges, was an integral part of our design (and the name fit in nicely with the theme). However, using a base MediaWiki [9] installation on a virtual machine with 256 MB of RAM was a mistake. As soon as the competition started, LityaLeaks was brought to a crawl due to the amount of traffic created by the teams.

Having LityaLeaks down was very problematic, because if teams couldn't see which transitions were firing then they couldn't submit flags. Eventually, a static mirror of LityaLeaks was brought up. Because of this, we had to change the Petri net software on the fly to update a publicly accessible file with the transition firings instead of using LityaLeaks. Once the change was made, at 13:30 PST, teams started submitting flags, and the rest of the competition went fairly smoothly.

As the scoreboard is the only way for teams to understand the current state of the game, making the scoreboard accurately reflect the status of the competition was essential. However, each piece of the competition's infrastructure was developed and tested independently. Knowing that getting the firewall, mothership, and Snort systems working properly was very important, those parts of the functionality were heavily tested in isolation. However, the interaction of these systems with the scoreboard was not tested before the competition. Thus, during the competition we discovered that the reasons given to teams for being blocked on the scoreboard were not correct, and in some instances the connection status of some teams were incorrect. Due to one of the developers being ill, it took us most of the competition to completely resolve this issue. While we were fixing the issue, we communicated to teams that to test their network connectivity, they could simply try connecting to one of the services. In the future, we will be testing our infrastructure as a whole, including important pieces like the scoreboard.

One issue with creating a complex and novel competition is that some teams might not "get" the competition. This can be on a number of levels, perhaps the team has never heard of Petri nets or could not exploit any of the services. This puts them at an extreme disadvantage in the rankings, as they cannot score any points. This was the case for 33 teams. However, for the 39 teams that submitted flags, a novel competition challenged them to create new solutions and tools, learning in the process. Ultimately, it is up to the competition administrators to balance novelty, complexity, and fairness.

5.3 What Worked?

Putting a backdoor into the bot VM that we distributed to the teams was something that we implemented five hours before the distribution of the VM. Something that we saw as funny turned out to have serious implications. One team came to us and said that they had an exploit to reduce every team's money to zero, effectively removing everyone else from the competition. Using the backdoor, they could bribe the Litya administrators as the team's bot, thus draining all of the team's money. We asked them not to do this, as it was unsporting to completely shut off most team's access to the services, and fixed this avenue of attack. We also alerted the teams to the existence of a backdoor on their VMs. Later in the competition, a team came to us complaining that their points kept decreasing. Looking into it, a team was exploiting a service, and submitting all the inactive flags (worth negative points) through another team's compromised bot. The team that this happened to came in last place (with -3300 points).

The backdoor provided some interesting (and funny) situations, however it came at a price. The last place team felt that this was an unsporting thing to do and were rightly upset over their last-place standing. We ruled that, since we had given notice about the backdoor, and given the extremely easy fix (filter the traffic from other teams), the outcome was acceptable. However, this situation did highlight an issue that these kind of "easter eggs" can produce: while it may be exciting and interesting for the teams who discover them, the more inexperienced teams who are not looking for them and/or can't find them are put at a disadvantage. This just increases the gap between the experienced and inexperienced.

6. CONCLUSIONS

Live cyber-security exercises are a powerful educational tool. The main drawback of these exercises is that they require substantial resources to be designed, implemented, and executed. It is therefore desirable that these exercises provide long-lasting byproducts for others to use for security research. In this paper, we presented a unique, novel design for a live educational cyber-security exercise. This design was implemented and a competition involving almost a thousand world-wide students was carried out in December 2010. We discussed the lessons learned, and we presented the dataset we collected, which we believe is the first public dataset focused on Cyber Situational Awareness. We hope that this dataset will be useful to other researchers in this increasingly popular field and that future security exercises will yield interesting datasets.

7. REFERENCES

[1] T. Augustine and R. Dodge. Cyber Defense Exercise: Meeting Learning Objectives thru Competition. In *Proceedings of the Colloquium for Information Systems Security Education (CISSE)*, 2006.

[2] N. Childers, B. Boe, L. Cavallaro, L. Cavedon, M. Cova, M. Egele, and G. Vigna. Organizing Large Scale Hacking Competitions. In *Proceedings of the Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA)*, Bonn, Germany, July 2010.

[3] W. Clark. *The Gantt chart: A working tool of management.* New York: Ronald Press, 1922.

[4] C. Cowan, S. Arnold, S. Beattie, C. Wright, and J. Viega. Defcon Capture the Flag: defending vulnerable code from intense attack. In *Proceedings of the DARPA Information Survivability Conference and Exposition*, April 2003.

[5] A. D'Amico, L. Buchanan, J. Goodall, and P. Walczak. Mission Impact of Cyber Events: Scenarios and Ontology to Express the Relationships between Cyber Assets, Missions and Users. In *Proceedings of the International Conference on Information Warfare and Security*, Dayton, Ohio, April 2010.

[6] D. R. Hipp. Sqlite. `http://www.sqlite.org/`, 2010.

[7] Justin.tv. `http://justin.tv/`.

[8] S. Liang. *Java Native Interface: Programmer's Guide and Reference.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1999.

[9] Mediawiki. `http://www.mediawiki.org/`.

[10] B. Mullins, T. Lacey, R.Mills, J. Trechter, and S. Bass. How the Cyber Defense Exercise Shaped an Information-Assurance Curriculum. *IEEE Security & Privacy*, 5(5), 2007.

[11] J. Peterson. Petri Nets. *ACM Computing Surveys*, 9(3), September 1977.

[12] L. Pimenidis. Cipher: capture the flag. `http://www.cipher-ctf.org/`, 2008.

[13] Pwn2own 2009 at cansecwest. `http://dvlabs.tippingpoint.com/blog/2009/02/25/pwn2own-2009`, March 2009.

[14] W. Schepens, D. Ragsdale, and J. Surdu. The Cyber Defense Exercise: An Evaluation of the Effectiveness of Information Assurance Education. Black Hat Federal, 2003.

[15] Simpp. 3vilsh3ll.c. `http://packetstormsecurity.org/files/view/64687/3vilSh3ll.c`.

[16] Snort. `http://www.snort.org/`.

[17] B. Stone-Gross, R. Abman, R. Kemmerer, C. Kruegel, D. Steigerwald, and G. Vigna. The Underground Economy of Fake Antivirus Software.

[18] The HackerDom Group. The ructf challenge. `http://www.ructf.org`, 2009.

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

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

[21] VMware. `http://www.vmware.com/`.

APPENDIX

A. VULNERABLE SERVICES

A brief description of the 10 services in the iCTF and the vulnerabilities associated with it follows.

LityaBook was a social networking website, similar to Facebook. By creating an underage girl profile, the attacker would cause President Bironulesk to visit their profile. They could then use a Cross-Site Scripting attack to steal President Bironulesk's browser's cookie, which contained the flag.

LityaBook also had a session fixation vulnerability. The authentication cookie contained the MD5 of the session ID. Therefore, an attacker could lure a victim to log in with a specific session ID, allowing an attacker to impersonate the victim. This vulnerability could have been exploited by using another website, LityaHot.

LityaHot was a website where young models posted links to their pictures, waiting for casting agents to contact them. Periodically, a member of President Bironulesk's staff, Femily Edeo, visited this site, clicking on links people had posted. If the link was a LityaBook page, he logged in to check the pictures. Thus an attacker could post a link on LityaHot, leveraging the session fixation vulnerability to log into LityaBook as Edeo and obtain the flag.

icbmd was the first iCTF service with perceptible effects on the real world. A USB foam rocket launcher was connected to a control program, pointing in the direction of a physical target. A time-sharing mechanism was used to share the missile launcher amongst the teams. Each team had a visual clue of where the launcher was aiming, via a webcam mounted on the missile launcher with a live streamed video to the Justin.tv on-line video streaming service [7]. The team currently controlling the missile launcher could exclusively connect to the control and move the launcher's turret. An encoded version of the launch code was leaked to the teams. After deciphering the code, the teams were able to launch a missile. Once a team successfully hit the target, the flag was sent to them.

StormLog was a web application that displayed log files generated by a fake botnet called "Storm." This service had a directory traversal vulnerability which allowed an attacker to download a copy of the cgi-bin program. An attacker had to exploit an off-by-one overflow in the cgi-bin program to execute arbitrary code and obtain the flag.

StolenCC was a web service that displayed text files containing credit card numbers. The cgi-bin program was written in Perl and contained a directory traversal vulnerability. By inserting a null character into the `filename` parameter, an attacker could bypass the program's sanity checking and open any file. Then, an attacker could use additional functionality of Perl's `open` to execute any command, finding and displaying the flag.

SecureJava was a web service that used a Java applet to perform authentication. An attacker needed to get past the authentication to find the flag. This involved reverse engineering the encryption algorithm. Once understood, the attacker leveraged a flaw in the encryption algorithm to steal the flag.

IdreamOfJeannie was a Java service that collected credit card information. Even though the bulk of the service was written in Java, JNI [8] was used to include a function written in C, which contained an off-by-one error. The attacker could utilize the off-by-one error to obtain the flag.

WeirdTCP was a C service that acted as a file server with a trust relationship with a specific IP address. A blind TCP spoofing attack against the service pretending to be the trusted IP address was required to find the key. However, due to the VPN technology we were using, packets could not be spoofed. A custom IP protocol RFC was given to the teams, which introduced an IP option that could be used to overwrite the source address of an IP packet. Thus an attacker had to use the IP option to spoof the trusted IP address, and, in addition, perform a sequence number guessing attack, in order to provide the correct acknowledgment number during the TCP handshake. Once the TCP connection was established, the attacker received the flag.

MostWanted was a Python service with a SQLite [6] backend. The service hosted mugshots of various wanted "criminals," and allowed a user to create or view mugshots. MostWanted had a stored SQL-injection vulnerability, which an attacker had to exploit to access the flag.

OvertCovert was a C-based service that allowed a user to store and access encrypted data. An attacker had to first exploit a printf vulnerability (which disallowed `%n`) to extract the encryption key. Then, an off-by-one error was used to access the encrypted flag. Using the key previously obtained, the attacker could decrypt the flag and exploit the service.