A Real-Time Information Warfare Exercise On A Virtual Network

James Walden
University of Toledo
1005 Abbe Rd N
Elyria, OH 44035
jwalden@eecs.utoledo.edu

ABSTRACT

Information warfare exercises, such as "Capture the Flag," serve as a capstone experience for a computer security class, giving students the opportunity to apply and integrate the security skills they learned during the class. However, many information security classes don't offer such exercises, because they can be difficult, expensive, time-consuming, and risky to organize and implement. This paper describes a real-time "Capture the Flag" exercise, implemented using a virtual network with free, open-source software to reduce the risk and effort of conducting such an exercise.

Categories and Subject Descriptors

K.3 [**Computers & Education**]: Computer & Information Science Education—*Computer Science Education*

General Terms

Experimentation, Security, Virtual Machine

Keywords

Information Warfare, Capture the Flag, Computer Security, Network Security, Laboratory, Exercise, Virtual Machine, User-mode Linux

1. INTRODUCTION

An overview course in computer security covers a wide range of theoretical and practical topics. While exercises on individual topics aid student understanding of particular aspects of security, effectively securing information systems requires consideration of multiple areas of security theory and technology simultaneously. An exercise where students secure and defend systems against live attack serves as a capstone experience, integrating student understanding of theoretical and practical aspects of security. To provide such a capstone experience for our first offering of a computer security class at the University of Toledo in the Spring 2004 semester, we developed a live information warfare exercise.

Information warfare is the defensive and offensive use of information and information systems to exploit an opponent's information resources. "Capture the Flag" is a symmetric information warfare exercise, where students work in teams to defend their own computer system while also attempting to exploit systems controlled by other teams by planting their team's flag on the other systems. Due to the unusual nature of this exercise, where students design and execute attacks on live systems, it is essential to educate students beforehand about the ethical and legal implications of attacking computer systems outside of the exercise.

We implemented our exercise using a network of virtual machines. A virtual machine (VM) is software that emulates the hardware of a computer. The virtual machine software is started on the host computer, then users run an operating system and their applications on the virtual machine as they would on a physical computer. A virtual network is constructed using network emulation software to connect the virtual machines' emulated network interfaces to each other. Virtual machines reduce the hardware and maintenance requirements of an information warfare exercise.

"Capture the Flag" exercises have been run since 1996 at the Defcon conference[1, 4, 7] and recent papers describe their use as teaching tools in computer security courses[15, 16, 18]. The exercise described in this paper differs from those in prior educational papers in its real-time nature, scoring system, and in its use of a virtual network.

2. INFORMATION WARFARE EXERCISE

2.1 Goals

The purpose of the exercise was to integrate theoretical and practical student understanding of security concepts and technologies presented in the course. The exercise was designed to achieve this purpose through three specific goals:

1. Students would gain experience securing a host as a team, preparing it for live attacks during the exercise.
2. Students would gain experience reacting to an attack in real time, learning to adapt their defenses in a crisis situation.
3. Students would gain experience with attacker strategies and tactics, as they researched, designed, and constructed attacks against other teams.

SIGCSE '05 February 23-27, St. Louis, Missouri, USA

We emphasized to students that attack strategies and tactics were taught because of their importance in understanding how to effectively construct and test defenses against such attacks. Students were warned that use of attack tools or tactics outside of an authorized context such as the "Capture the Flag" exercise or a pre-approved penetration test was unethical and likely to result in legal consequences.

2.2 Rules

Students chose teams of 3-4 people to participate in the exercise. They had a one week preparation period before the exercise to research the operating system and software that they would use in the exercise, ask questions of the instructor, download defensive and offensive tools, prepare configuration files, and write their own tools.

Teams met in the computer lab on a Saturday for the actual 8 hour exercise, which was divided into two 4 hour segments with a short break between them. Having the teams sitting adjacent to each other in the lab facilitated interaction between teams during the exercise, including offering the opportunity for social engineering attacks.

When the exercise began, each team was given the root password and IP address of their assigned virtual machine. They had access to their machines via ssh. During the first half of the exercise, student teams installed and configured software to secure their respective virtual machines. Teams also installed offensive software for use during the second half of the exercise. All virtual machines had limited access to the campus network and Internet during this period to facilitate downloading software and applying security patches. A firewall on the machine hosting the virtual machines blocked outgoing packets destined for ports other than ssh, ftp, http, and https. The firewall also prevented the virtual machines from contacting each other during the defensive phase. All packets on the network were captured and examined to ensure that no team attempted to begin the attack phase early.

After the initial defensive segment ended, students were given a break to eat while the virtual network was reconfigured to isolate it from the campus network for the offensive phase of the exercise. It is important to isolate the exercise network in order to avoid stray packets from causing damage to production networks. The virtual network was also reconfigured to allow the participating teams' virtual machines full access to each other. Teams were permitted to react to attacks and enhance their defenses during the offensive period, as well as to research and launch attacks. The attack segment of the exercise was conducted in real time in order to simulate the crisis situation of an actual attack.

Teams were scored according to the number of times their team's flag was read on the scored network services by the scoring system. Teams received points for maintaining their flag on their own machines, as well as for placing their flag on other team's machines. Flags were simple text strings created by appending the team's numerical identifier to the constant string "team."

One point was awarded to a team for each of the three scanned network services that displayed that team's flag during a scan by the scoring system, for a maximum of three points per machine on the virtual network. The scoring system scanned the network once every 10 minutes during the offensive segment of the exercise. The three network services chosen for scoring the exercise were telnet, SMTP, and HTTP.

Teams were permitted to patch or even remove the scored services, but no points were assigned unless the service was available and displayed the team's flag to the scoring server. Blocking IP addresses on those ports so that only the scoring system could access a team's services was forbidden, as was sending spoofed data to the scoring server. The network was scanned from different IP addresses to check for any illegal firewalling of the flagged services. Scores were announced on a regular basis, but several teams independently checked their host's integrity by writing their own software to scan its flags.

Several types of attacks were forbidden, including attacks on the host machine or outside network, denial-of-service attacks except as a short term part of another attack such as IP spoofing, and destructive attacks designed to disable a hacked host. Network traffic during the exercise was recorded and examined to check for forbidden attacks, including attempts to spoof to the scoring system.

Afterward, teams were required to write a report on the exercise, detailing defensive preparations made before and during the exercise, successful and unsuccessful attacks that they attempted, and defensive and offensive reactions to attacks against their own machine.

3. IMPLEMENTATION

3.1 Virtual Networks

An information warfare exercise requires an isolated network, that is, a network which is separated by a physical or software barrier from other networks in order to protect their integrity. A "Capture the Flag" exercise generates a wide array of dangerous network traffic, some of which may damage networks that are not participating in the exercise unless the exercise network is isolated. Students also need to have administrative access to their machines, so that they can harden systems and use attacker tools that require root access. Machines are extensively reconfigured during the exercise by defenders and possibly also by attackers who penetrate the machine's defenses.

The common solution to these issues is to construct a lab with an isolated network of physical machines[9, 12, 13, 14, 19]. However, such laboratories require time, money, and space to set up. Computers need to have operating systems and software installed and reconfigured before, after, and possibly during the exercise, and networks need to be specially configured, requiring extensive maintenance skill and effort.

Virtual networks offer instructors the ability to set up an information warfare exercise at a lower cost in resources. A network of virtual machines doesn't require a dedicated space or dedicated machines. The virtual network can be instantiated when needed and shut down when not used, allowing ordinary lab machines to be used temporarily to create an isolated network for computer security exercises without a great deal of effort. The virtual network provides the necessary isolation for a security exercise by default, but can be connected to the Internet if so desired. Restoring a virtual machine to its initially configured state simply requires deleting a single file and restarting the virtual machine software.

A variety of free, open source virtual network and virtual machine software exists today. We chose User Mode Linux (UML)[6] for our virtual machine software because it is free, open source, and has good performance. UML is limited to emulating a computer running the Linux operating system, but that was not a problem for the exercise as each team was assigned only one virtual machine and we wanted all teams to be using the same operating system.

If we had wanted to use an operating system other than Linux or to assign multiple machines, each running a different OS, to each team, we could have used a program like the commercial product VMware[17], which is a virtual machine that can run a variety of operating systems, including FreeBSD, Microsoft Windows, and Solaris x86. Open source virtual machine programs that support multiple operating systems exist, such as Bochs[3] and Xen[2], but are either slower or not as stable as VMware.

We used the open source TUN/TAP virtual Ethernet interface[11] and bridge-utils[8] for Linux to construct our virtual network[5]. To instantiate the virtual network on the host computer, a script ran bridge-utils to create a virtual network switch with one network port for each virtual machine. The script also created one virtual Ethernet interface per virtual machine and connected it to a virtual port on the switch, so that traffic sent through that interface would be forwarded by the switch to its destination. When each UML virtual machine was started, it was given one of the virtual Ethernet interfaces so that it could communicate on the network.

At this point in the process, the virtual network switch only connects the virtual machines to each other. The network is completely isolated. In order to connect the virtual machines to the campus network, bridge-utils is used to connect the Linux host machine's physical network interface to the virtual switch. Network packets from the virtual machines destined for the campus network are then forwarded by the virtual switch to the host machine's physical network interface which is connected to the campus network.

Maintenance is simple, as User Mode Linux virtual machines store changes to the filesystem in a copy-on-write (COW) file containing the user's modifications to the VM filesystem. Restoring the virtual machine to its original state is simply a matter of deleting the COW file and restarting the UML software. Multiple virtual machines can share a filesystem image, using their own COW files to store their different modifications. Virtual networks are also portable, easily moved to another host machine by copying the UML software and filesystem image. After the exercise, the virtual machines and network are shut down, making the host machine immediately available for normal use.

There are some disadvantages to virtual machines. They cannot be used for exercises involving physical security. Virtualization also imposes some overhead costs, so that a virtual machine is not quite as fast as the physical machine on which it is running. However, the virtual network can be faster than the host machine's physical network since it is not limited to Ethernet speeds.

3.2 Specifics

We created a Redhat Linux 9.0 image for each virtual machine to run. No security patches were installed on the image, and, in fact, the system's security was reduced by introducing user and guest accounts with weak passwords and by turning on unnecessary network services. The filesystem image was stored as a sparse file on the host machine.

Each virtual machine used the same filesystem image file but had its own COW file. The only differences between machines at the start of the exercise were their hostnames, IP addresses, and their root passwords. All files, including both startup scripts and the filesystem image are available from the exercise web site[10].

Three network services were set up and modified for the scoring system. The services chosen were telnet, SMTP, and HTTP, in order to require teams to maintain and defend a commonly used set of network services. Since these services were necessary to gain points, there was no need to impose additional rules requiring that students run specific services on their machines. The modifications were small changes to the service's configuration or source code so that the service would produce the team's flag on request. Teams were permitted to modify the configuration of these services, or even run different programs to provide the same services, but would not receive any points if they broke the flag functionality.

The telnet service displayed the team's flag on its login banner; the flag was simply included in the `/etc/issue` file. The `sendmail` program provided SMTP service and was patched to add a new SMTP protocol command `FLAG`, which would output the team's flag when issued. The Apache web server provided HTTP service, and the flag for this service was retrieved by sending a request with the appropriate form parameters to an intentionally insecure CGI script.

The Linux `iptables` firewall on the host machine was set up to allow Internet access for the initial defensive half of the exercise, and was reconfigured after the break to prevent Internet access during the offensive portion of the exercise. The host machine monitored all packets on the virtual network.

3.3 Issues

We originally planned to give teams access to their virtual machines using ssh to contact the host machine, where they could access the virtual serial consoles of their machine. This method would have permitted the virtual network to be completely isolated from the campus network during the offensive phase of the exercise. However, due to problems we experienced with virtual serial consoles, we gave teams direct ssh access to their machines and provided isolation by blocking all outbound connections initiated from the virtual machines during the attack phase using the firewall on the host machine.

While our host machine, a 2.8GHz Pentium 4 temporarily upgraded with 1.5GB of RAM borrowed from other lab machines, was able to run over a dozen UML network servers without any performance issues, we discovered that 5 virtual machines on which 3-4 interactive users each were compiling tools and cracking passwords required much more processing power. The virtual machines slowed to a crawl, with keystrokes taking minutes to be echoed back to students' terminals until we paused the exercise to reconfigure the virtual network. We moved two virtual machines from the original host machine to another host machine and reconfigured the firewall and virtual switch to forward packets between the two hosts.

The reconfigured network solved our performance problems for the remainder of the exercise. We plan to bench-

mark an identical system using Linux 2.6 kernel-based UML virtual machines to determine if the interactive scheduler enhancements in the 2.6 kernel will provide a sufficient performance improvement to run the exercise on a single host machine next year.

4. LEARNING EXPERIENCES

Students discovered that designing and carrying out an attack on a live system was much more difficult than they expected. While most teams were able to gain user level access to at least one other team's machine, they were unprepared to elevate their privileges to gain root access. One team gained root access by obtaining another team's root password without their knowledge through a simple social engineering attack. No other team was able to gain root access. We used the success of the social engineering attack in the post-mortem analysis of the exercise as an example to illustrate that security is a process involving people, not purely a technical exercise.

We suspect that part of the difficulty was the students' focus on downloaded remote root exploits, especially as no such tool worked successfully for any team. One such tool crashed the attacking team's machine, losing them points for not having their flag displayed on their own machine during one scoring scan. After the exercise, several teams wrote in their reports that they should have focused on multiple part exploits, where initial user-level access was gained then later elevated to root access. Students developed only one unique offensive a tool, a program to set all flags on the current machine to their own, but that team did not gain the root access required to successfully run their program.

Most teams focused more of their efforts on defense than attack. Network security was generally good, but host security was weaker. Besides one team that accidentally left an ftp server running that they thought they had disabled, all unnecessary network services were shut down as part of the initial defensive stage. Teams also deployed a variety of defensive network software, including firewalls and intrusion detection systems.

However, most teams missed some elementary host security problems, such as the existence of insecure user and guest accounts. Only one team disabled those accounts before experiencing an attack using them. Some teams enhanced host security by backing up system binaries in case of their replacement with trojan horse versions through a rootkit, while others deployed tripwire.

Several teams wrote and deployed their own defensive software. Multiple teams deployed software to scan their flags at regular intervals, and one deployed a program that restored the flags to their initial state every minute, requiring a successful attacker to either synchronize their attack carefully with the scoring scans or to find and disable this unique defensive software as part of their attack. Another team wrote a shell wrapper, which required a password to be entered before a shell would be launched. This wrapper saved their insecure user and guest accounts from being exploited, but the wrapper also caused a delay when their machine crashed and the wrapper had to be disabled before it would restart in multiple user mode.

Student response to the exercise was enthusiastic, despite having to come into the lab on a Saturday. Students comments rated the exercise highly, both in terms of fun and in terms of its educational value, with one student writing "Capture the Flag was fun, in a mentally exhausting way that I really wasn't expecting. While our defense made progress the entire time, offense was frustrating for the first 5 or 6 hours; but when things started to make sense and we began to get into other machines, it was a rush. It takes weeks of lectures and applies them all at once, which is a great experience." Another student suggested running the exercise again as part of the final exam.

5. LESSONS LEARNED

5.1 Issues

While one team requested more time for defense in their report, the extent of the defensive measures deployed and the difficulty all teams had in carrying out successful attacks suggests that the exercise needs to be biased more toward offense, not defense. We plan to emphasize the importance of multiple part exploits involving privilege escalation during lecture and laboratory sessions next year. We also plan to modify the flags used for scoring and to insert additional security flaws into the system image in order to increase the difficult of securing systems.

Several teams stated in their reports that they felt underprepared for the UNIX system administration aspects of the exercise. While most students could use common UNIX utilities and programming tools, they had little experience building and installing software packages, administering user accounts, or configuring servers. We intend to address this lack next year by creating additional exercises using UML virtual machines to give students hands-on experience with these tasks before the "Capture the Flag" exercise.

5.2 Future Directions

In order to improve the chances of attackers next year, we intend to add new flags to the scoring system, including ones that will require logging into student machines to check them and others that will require the ability to transfer files to and from the machine to be checked. We also intend to create flags that do not require root access to modify. These requirements should increase the effort require to harden and defend a team's host machine, while decreasing the effort required for a successful attack, which should result in more teams gaining experience with intrusion detection and incident response.

If we can obtain sufficient resources, we would like to provide an entire virtual network for each team to defend, including multiple virtual machines running different operating systems. Such an environment would offer a better simulation of a network, allowing us to create additional flagged services for students to defend, such as authentication and database servers.

We would also like to improve the simulation by adding virtual hosts to the environment that are not administered by student teams. Such hosts would have trust relationships with machines administered by student teams, offering more complex configurations to defend and opportunities for students to attempt more complex attacks. In addition, we plan to insert decoy traffic into the virtual network to better simulate a production network environment. Students will have to sift through network traffic to discover which packets were sent by attackers and which were normal parts of the environment, instead of being able to assume that any

packets destined for their machine from IP addresses other than that of the scoring system were hostile.

6. CONCLUSIONS

We found that a "Capture the Flag" information warfare exercise provided an excellent capstone experience for an overview of computer security class. Students were highly enthusiastic about the exercise and expressed their appreciation of how it helped them integrate and apply much of what they learned in the course.

The use of open source User-Mode Linux virtual machines and Linux virtual networking software allowed us to construct the isolated network we needed for the exercise without the requirement of a dedicated space or machines. The two machines used in the exercise were removed from normal use for only the 8 hours of the actual exercise plus an hour of setup and tear down time. The virtual machines were run one at a time after the exercise for grading purposes, while leaving the machine in normal use.

Tying the scoring system's flags to the network services we wanted students to defend eliminated the need to create a separate scanning system for those services. The scoring system also offered minimal opportunities for attackers to gain points by requiring that the servers be present and be configured to provide the flag when requested. We plan to continue using a "Capture the Flag" exercise as a capstone experience for our security class and to improve the rules, scoring system, and virtual network in future iterations.

7. REFERENCES

[1] Defcon IV announcement. http://www.defcon.org/html/defcon-4/, 1996.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177. ACM Press, 2003.

[3] Bochs. http://bochs.sourceforge.net/.

[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 DISCEX III Conference*, pages 120–129. IEEE CS Press, 2003.

[5] D.Cannings. Networking UML using bridging. http://edeca.net/articles/bridging/index.html, 2004.

[6] J. Dike. A user-mode port of the Linux kernel. In *Proceedings of the 4th Annual Linux Showcase and Conference (Usenix 2000)*, 2000.

[7] Ghetto-Hackers. Root-fu. http://www.ghettohackers.net/rootfu/, 2004.

[8] S. Hemminger. Bridge-utils. http://bridge.sourceforge.net/.

[9] J. M. D. Hill, C. A. Carver, Jr., J. W. Humphries, and U. W. Pooch. Using an isolated network laboratory to teach advanced networks and security. In *Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education*, pages 36–40. ACM Press, 2001.

[10] J.Walden. Capture the flag. http://www.eecs.utoledo.edu/~jwalden/ctf.html, 2004.

[11] M. Krasnyansky and M. Yevmenkin. Universal TUN/TAP driver. http://vtun.sourceforge.net/tun/, 2001.

[12] P. Mateti. A laboratory-based course on internet security. In *Proceedings of the 34th SIGCSE technical symposium on Computer science education*, pages 252–256. ACM Press, 2003.

[13] M. Micco and H. Rossman. Building a cyberwar lab: lessons learned: teaching cybersecurity principles to undergraduates. In *Proceedings of the 33rd SIGCSE technical symposium on Computer science education*, pages 23–27. ACM Press, 2002.

[14] J. Schafer, D. J. Ragsdale, J. R. Surdu, and C. A. Carver. The iwar range: a laboratory for undergraduate information assurance education. In *Proceedings of the sixth annual CCSC northeastern conference on The journal of computing in small colleges*, pages 223–232. The Consortium for Computing in Small Colleges, 2001.

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

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

[17] VMware. http://www.vmware.com/.

[18] P. J. Wagner and J. M. Wudi. Designing and implementing a cyberwar laboratory exercise for a computer security course. In *Proceedings of the 35th SIGCSE technical symposium on Computer science education*, pages 402–406. ACM Press, 2004.

[19] T. Wulf. Implementing a minimal lab for an undergraduate network security course. *J. Comput. Small Coll.*, 19(1):94–98, 2003.