EDURange Instructor's Manual

January 21, 2016

A Introduction

EDURange is working. This document will be updated as changes are made. EDURange is both a collection of interactive, collaborative cybersecurity exercises and a framework for creating these exercises. Currently, we have several exercises that are ready to use: ssh Inception, Total Recon, ELF Infection, strace, and scapy hunt. There are designs for a DNS exercise, DDoS, fuzzing, ROP, and firewall. They can be done in any order, although scapy hunt and ELF infection are more advanced. Most of the exercises require a minimal level of understanding of some standard Linux tools.

Recon was the first exercise created and was based on a scenario from PacketWars. It focuses on reconnaissance to determine hosts in an unknown network. The standard tool for this is `nmap`, and while the student will need to learn how to use that tool in order to do this exercise, that is not the most important learning goal. The most important learning goal is developing the analytical skills of understanding complex systems and complex data. Similarly, the Elf Infection exercise can use standard tools such as `netstat` and `objdump` but requires that students reason about the behavior of a complex system to discover which binary is infected and what it is doing, e.g. it opens a port and listens for connections, which it should not be doing. Scapy hunt is about listening passively to discover hosts on the local network and which other hosts they are talking to. There are several more exercises planned, and they will be described as they become available. An important part of these exercises is discussing them in class after the students have completed them. More information about the exercises can be found in the Student Manual.

A.1 Using EDURange: AWS

As an instructor, when you login to `https://cloud.edurange.org`, you will be able to run scenarios, view student data, and monitor scoring events. Once you have a registration code, you can create your account and the EDURange team can designate it as an instructor account.

Once you login, you will see several tabs at the top. From the `Home` page, you can create student groups. These groups can be thought of as teams that will work together. The `Tutorials` tab has the essentials you need to get started. The `Scenarios` tab will let you run scenarios. Booting a scenario creates new VM instances and configures them. This interface is a front end for interacting with Amazon's AWS console in a constrained environment. For each scenario, there a YAML file that specifies the exercise. The YAML interpreter will run scripts that will configure the VMs. As part of the configuration, student accounts and credentials are created on the appropriate VM instances. These temporary credentials are for the VMs in the exercise and are distinct from those on cloud.edurange.org. You can see the instructor login credentials and student login credentials for the scenario near the bottom of the page. *This allows you to try the scenario as a student.*

Currently, you can modify a scenario, for example changing the host IP addresses by using `Load New Scenario` tab and then the `Clone` tab from there. A new scenario will be created and you can edit it subject to the resource limitations of the account. The underlying mechanism is that the YAML file is modified for that scenario. Before booting the scenario, you should add an students whom you want to have access. In general, students will each have their own AWS instances to login to and work on the exercises. There should be at least one team VM created even if you have not assigned any students to the scenario. You will be able to find your own password as instructor, and once everything is booted, you can login to the NAT host, which has a public IP address, using an ssh client.

Using EDURange as an instructor

You should be able to perform all instructor functions from the EDURange console on `https://cloud.edurange.org` There is a form on this website for you to request access, but you can also send e-mail to kaheah [at] gmail.com or weissr [at] evergreen.edu . Once you have an account, here are the basic steps to run a scenario:

- Login to https://cloud.edurange.org.
- Give your registration code to students – this allows them to register for their student account. Students then use the registration code to sign up.
- When students register and you refresh your home page, you will see that they are automatically added to the group All. You may create new groups and add students to those groups to simplify administration.
- Now, you are ready to start a new scenario. Click on the `Scenarios` button at the top. Then, you will see a page with buttons to create the different scenarios. Create the scenario you want, and you can add students from one or more of your groups. You can also remove groups through this interface. When you have done that, you should click on `Boot` to start the VMs. This will take about 5 min.
- To try the exercise as a student, you can use a separate terminal to ssh as instructor to the NAT that was created by the script. The NAT will have an external IP address.

B Exercises

- **Strace** (dynamic analysis of binaries) poses the challenge of understanding what a process is doing based on its system calls. Students learn to filter large amounts of data to distinguish between normal and anomalous behavior.
- **Elf infection** assesses the student's understanding of the structure of an executable file. The goal is to teach the student, having identified that a program is doing something malicious, where that code has been injected and how it works. This is a reverse engineering problem and can use a range of tools, including readelf, objdump, gdb, strace and netstat.
- **Recon** is an exercise to examine how network protocols such as TCP, UDP, and ICMP can be used to reveal information about a network. Recon focuses on reconnaissance to determine hosts in an unknown network. The student can explore tradeoffs between speed and stealth when using tools such as nmap. More advanced levels will address intrusion detection and network monitoring.
- **Scapy Hunt** poses the challenge of analyzing network traffic to understand who is communicating with whom and how. The player is trying to get data from an ftp server which is not on the same subnet, but one of the hosts on its network is communicating with it. By default the player can only see packets sent to the server and must craft packets to get them routed to the target and get a response back.
- **Fuzzing** is an attack-defense exercise where the defender must correctly implement a user interface that filters malformed expressions. In the simplest version, the interface is for a calculator, and the defender must implement both the interface and the application.

- **Firewall** is still being implemented. It requires students to understand the interactions between multiple firewalls which have different rule sets.

B.1 strace

This exercise is based on ones in Hacking the Abacus. Our goals in writing this exercise are to teach students how to perform dynamic analysis on software. This includes being able to distinguish between anomalous behavior and normal behavior. This can be very useful for identifying malware. This means they should learn the important lesson that all programs need to make certain system calls just to run, and therefore one would ignore these to look for more interesting calls. In order to see this, one can run strace on an empty program on local Linux computer or VM. `strace` is a useful tool for understanding how programs interact with the operating system through system calls.

Prerequisite knowledge:

- using ssh
- Linux command line: navigate using cd, ls, a text editor, such as nano, vi, or emacs, gcc
- Linux PATH conventions, echo $PATH, the order of search for executables
- What a system call is
- A basic understanding of processes and subprocesses
- A basic knowledge of the main options for strace and how to run it.

Preparation for doing the exercise:

- run strace on some common utilites, find some common syscalls and look up what they do.
- look up the basic categories of syscalls
- read about syscalls to:
  - create a new file
  - make a new name for a file
  - execute a process
  - terminate the calling process
  - create message buffer and read from message queue
  - assign the local IP address and port for a socket
  - modify file protections

Additional assessment exercises:

- make a system call in C, for example call fexecve,
- make a system call in x86 assembly

Learning Goals for strace exercise

Understanding Level:

- How to recognize when a process unexpectedly forks another process
- How to recognize when a process unexpectedly opens a file or socket
- How to recognize when a process unexpectedly deletes a file
- How the kernel handles system calls
- Which system calls introduce threats and how that happens
- How errors are handled
- How to fuzz the input to a system call

Synthesis Level:

- How to analyze the sequence of sys calls and recognize patterns

Instructions:

From the Instructor Machine, select $Scenarios \rightarrow NewScenario \rightarrow CreateStrace \rightarrow Boot$ This will take 5-10 min to finish. The browser will display your temporary username and password for the NAT instance. From there, you can ssh to the Player Instance and do the exercise as a student would.

B.2 Elf Infection

The Elf infection exercise only uses a single VM for each team.

Prerequisite knowledge:

- Linux command line: navigate using cd, ls, a text editor, such as nano, vi, or emacs, gcc
- How to run gdb, objdump
- strace is helpful
- How x86 assembly works, including branches.
- A basic understanding of the segments and sections of an ELF file.

Here are the associated learning goals:

- know the capabilites of `readelf` and how to use the basic options.
- know the format for the header of an ELF file.
- know which system calls do the following:
  - make a new name for a file,
  - execute a process,

  - terminate the calling process
  - create message buffer and read from message queue
  - assign the local IP address and port for a socket

- know the general classes of system calls.
- Be able to read a system trace and know what is normal vs abnormal.
- be able to make a system call in C
- be able to make a system call in x86 assembly
- understand how the kernel handles system calls
- understand how some system calls introduce threats
- understand how errors are handled

B.3 Recon

Learning Outcomes for Recon

Students will answer the following questions:

1. what is the 3-way handshake?
2. what does the SYN flag do?
3. What does /17 mean? how many IP addresses does that include?
4. What are the options for nmap and what are their differences in terms of time, stealth and protocols?
5. which ports does nmap -sT scan?
6. which ports does nmap -sU scan?

- understanding networking protocols (TCP, UDP, ICMP) and how they can be exploited for recon.
- developing the security mindset
- understanding CIDR network configuration and how subdivide a network IP range.
- use nmap to find hosts and open ports on a network.

Diagram of Recon network

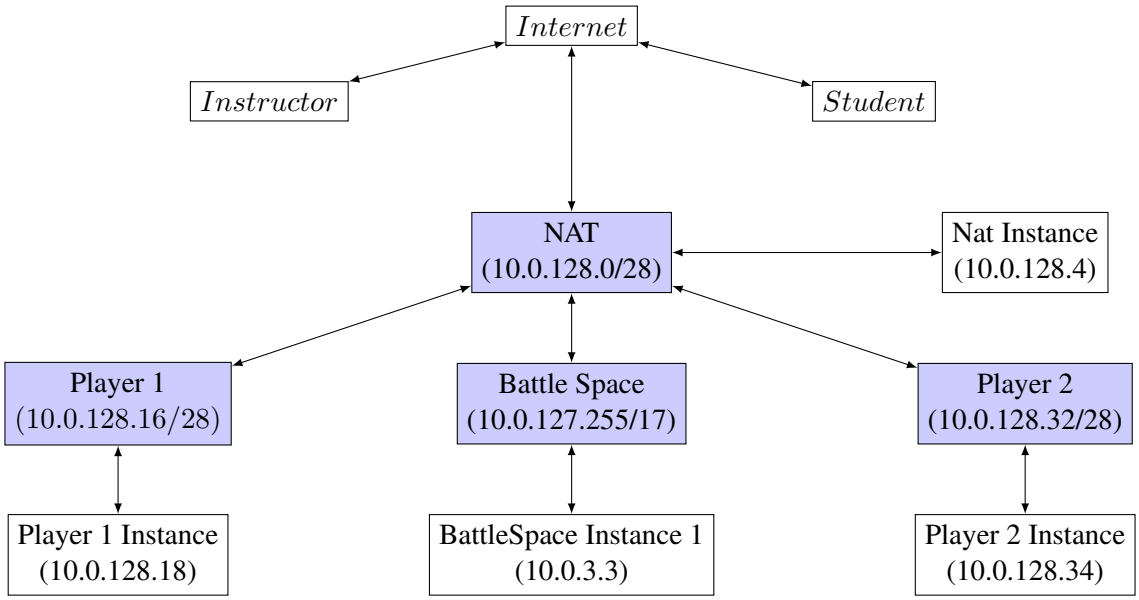

Figure 1: Conceptual diagram of the Recon I game. Note subnets are blue, and IP addresses are just as an example. The Battlespace is currently 10.0.0.0/17, but you can make your own copy of the scenario file and change this. You can also change the IP addresses of the targets.

B.4 Fuzzing

The Fuzzing scenario runs with 3 primary instances: scoring, defending and attacking, plus a NAT.

To test this scenario - first log into the defending instance by first logging into the NAT and then logging into the defending team's machine (defending_team@10.0.26.130), and make a calculator, the author of this scenario has provided a naïve calculator downloadable here (`http://ada.evergreen.edu/~weidav02/crashing_calc.c`). After compiling your calculator, you may submit it by running the following command (submit_calc /path/to/calc).

After submitting your calculator, you are ready to fuzz. Connect to the attacking team's machine (attacking_team@10.0.26.129), You may run my naive fuzzer example downloadable here (`http://ada.evergreen.edu/~weidav02/example_fuzz.py`) or you may try testing the calculator by hand with examples such as these following commands:
(`send_fuzz_data"22"`), or
(`send_fuzz_data"$(perl-e'print"\x03"')"`). Sending a hexidecimal character such as 0x03 is a good test to run as it is outside the valid character set for this calculator and should cause errors.

To view the scoreboard, you must log into the scoring machine (`scorer@10.0.26.128`) and run this command (cat /tmp/scoring/answers) - This will be updated in the next few days so that players may view the scoreboard from their machines.

C Future Work

- ** in progress ** Make it possible for a student to take a snapshot and stop and restart the exercise.

- We plan to create an API to choose random IPs for the Battlespace and other randomizations (passwords).
- Scorebot. A simple version is running.
- Instructor versions: each instructor can have his/her own YAML scripts that will be used to generate the scenarios page.

D References

E Student Tutorials

We have made an attempt here to summarize some of the prerequisite knowledge.

E.1 What are TCP and UDP?

In order to understand this exercise, you should be familiar with the 3-way handshake for TCP. You should also know something about ICMP and UDP. You will learn in this exercise, how these and other protocols can be used to discover hosts on a network, which ports on those hosts are open, and what applications are running on them. In practice, each message that is sent over the Internet uses multiple protocols, which are divided into five layers: physical layer, link layer, network layer, transport layer and application layer. For example, the physical layer handles what is a 0 or 1. The link layer handles communication on local area networks (LANs). The network layer handles rounting on wide area networks (WANs), e.g. IP. The transport layer handles ports and processes, e.g. TCP, UDP, ICMP. The application layer handles applications communicating with each other, e.g. http. by nesting packets inside of packets. In general, these packets correspond to layers of functionality: TCP is connection-oriented and is responsible for a number of things including reliably conveying messages between the application layers on two hosts. The three-way handshake establishes this pairing with the following sequence: SYN, SYN-ACK, and ACK You can get a summary of the important protocols and their layers in: Chapter 4 of ¡em¿Hacking¡/em¿ (Erickson)[1] or Chapter 2 of Counter Hack Reloaded [2]. ¡em¿Network Security¡/em¿ by Kaufman, Perlman, Speciner

subnets: how to split an IP range, IP CIDR notation

tcpdump (man tcpdump) on VM Wireshark will sniff network traffic. Although it may not capture all traffic on your network, it will at least show you what you network interface can see, including all of the messages that your computer is sending. This can be very useful when you are figuring out what nmap is doing. Since we are not currently using Xwindow forwarding,

Linux command line: man man, cd, ls , pwd, mkdir, teach the basics of Linux, find the file called password.txt

E.2 nmap

How does nmap work? nmap generates packets, sends them, and examines the responses (if there are any). For example, nmap -sT 192.168.12.1 will send TCP packets to ports 1-1024 at IP address 192.168.12.1. In order to understand what nmap is doing, we will use tshark, which is a CLI for Wireshark. read [3] on Wireshark and t-shark. The parameters to check on nmap include -sS, -sP, -sT, -sU, what about the -T option and -n?

This is important so students can see what is happening. Can you see the 3-way handshake? Is the IP address of the attack host visible? What happens if you spoof the return IP address? Can nmap do that? What does nmap -sS do? Suppose you want to map the network faster, without being stealthy? What are the -T options? How would you detect that a computer is scanning your network, running t-shark?

Tips: use -n for no DNS lookup. What is DNS? read [4] nmap has lots of options, so focus on the following first: -T, -sS, -sP, -sT, -n, -o

Using EDURange (students): In order to get started, you need to understand to connect to Amazon's AWS, the gateway and team hosts. You need some kind of credentials to login. You need to login to the a gateway on Amazon in order to use EDURange, but first you will need a password. You will be given a URL at the time of the exercise that will tell you your password and that will be the gateway computer. Note that for now, pw will be generated randomly by the yaml parser. navigate to gateway url in browser and get a browser-based ssh client, which will ask for uname/password

ideas for level-0 edurange games: debug challenges, clone from github. `netcat` on port 8000, why doesn't work? ARP tabl, delete gateway,

1 Hacking: The Art of Exploitation(Chapter..)

2 Counterhack Reloaded (Chapter 2)

3 man pages for nmap

contributors

Stefan Boesen, Erin Davis, Michael Locasto, Jens Mache, Lyn Turbak, Richard Weiss,