ASTES

Advances in Science, Technology and Engineering Systems Journal Vol. 6, No. 4, 355-367 (2021)

www.astesj.com

Special Issue on Multidisciplinary Sciences and Engineering

ASTESJ

ISSN: 2415-6698

Enhance Student Learning Experience in Cybersecurity Education by Designing Hands-on Labs on Stepping-stone Intrusion Detection

Jianhua Yang[1], Lixin Wang[*,1], Yien Wang[2]

[1]*TSYS School of Computer Science, Columbus State University, Columbus, GA 31907, USA*

[2]*College of Engineering, Auburn University, Auburn, AL 30060, USA*

ARTICLE INFO

Article history:
Received: 07 June, 2021
Accepted: 09 August, 2021
Online: 26 August, 2021

Keywords:
Stepping-stone
Intrusion Detection
Cybersecurity Curriculum
Ethical Hacking
Hands-on Experience

ABSTRACT

Stepping-stone intrusion has been widely used by professional hackers to launch their attacks. Unfortunately, this important and typical offensive skill has not been taught in most colleges and universities. In this paper, after surveying the most popular detection techniques in stepping-stone intrusion, we develop 10 hands-on labs to enhance student-learning experience in cybersecurity education. The goal is not only to teach students offensive skills and the techniques to detect and prevent stepping-stone intrusion, but also to train them to be successfully adaptive to the fast-changing dynamic cybersecurity world.

1. Introduction

1.1. Cybersecurity Significance

We live in a world where digital technologies are needed for various daily activities. The Internet has revolutionized data communications and significantly changed our daily lives. However, hackers can now easily launch cyberattacks using the Internet. As cyberattacks continue to grow, it is important to secure our critical infrastructures, organizations, business and networks.

1.2. The Importance of Stepping-stone Intrusion Detection

Intrusion techniques are widely used by intruders to invade a computing system. Intrusion detection systems (IDS) are installed on a lot of computer and network systems. Intruders tend to use several compromised hosts, called stepping-stones, to send attacking commands to a remote target host, in order to avoid being detected. Attacks that are launched through a chain of stepping-stone host are called stepping-stone intrusion. With a stepping-stone attack, intruders remotely login to such stepping-stones using tools such as SSH, rlogin, or telnet, and then send the attacking packets to the remote target host.

In this paper, after the survey of many known detection techniques for the stepping-stone intrusion, we propose ten hands-on labs which are developed based on the cutting-edge techniques in stepping-stone intrusion detection. The goal is to help students to learn the techniques of stepping-stone intrusion detection. We aim at educating learners to be qualified professionals in cybersecurity in order to defend various digital data and resources. It is also expected to enhance students' learning in cybersecurity education by conducting the hands-on labs designed.

2. Key Challenges

Before designing the hands-on labs on stepping-stone intrusion and its detection, we discuss how challenge the known detection approaches for stepping-stone intrusion are integrated into cybersecurity curricula. In order to educate learners to be qualified professionals in cybersecurity, it is necessary to teach offensive skills in college cybersecurity major curriculum.

Integrating stepping-stone intrusion and its detection techniques into cybersecurity curriculum can make us move forward a big step to achieve this goal. Although a great number of detection approaches for stepping-stone intrusion have been proposed since the emerging of the Internet, there are still a lot of challenges to integrate these detection approaches into cybersecurity curricula at the college level. The first challenge is why we need to teach college students ethical hacking skills. Would it be possible educate our students to become a hacker against us, not for us? The second challenge is that, since there are

[*]Corresponding Author: Lixin Wang, 4225 University Ave., Columbus, GA 31907, USA. Contact No: 001-706-507-8190. Wang_Lixin@ColumbusState.edu

too many algorithms for stepping-stone intrusion detection proposed in the literature, which approaches among them are suitable to our college students as learning materials? The third challenge is what hands-on labs can be developed and integrated into cybersecurity curriculum. We all know that the difficulty in teaching cybersecurity is not at the delivery of the theory and techniques; it is at the development of hands-on labs for students to practice hacking and defensive skills. Considering the limited budget in each four-year college, the cost is an important factor when designing these hands-on labs. However, we still want to motivate our students to learn cybersecurity skills via hands-on learning experience.

2.1. The Rationale to Teach Ethical Hacking Skills

Should we teach ethical hacking skills to cybersecurity major students? To the best of our knowledge, even though some four-year institutions have included ethical hacking skills as part of their cybersecurity curriculum, there are still some concerns and doubts from students' parents and local communities about the possibility that teaching ethical hacking skills would make their kids to conduct some malicious activities, and commit crimes. We must convince students' parents as well as the local communities with the following advice: 1) the word 'hacker' has long been understood negatively. Hacking actually involves computing skills to find vulnerabilities of a system, penetrate a system, and be able to remove evidence of accessing to a system [1]. Similar to the case that doctors who might criminally abuse their medical skills to hurt humans, a hacker who knows some special offensive hacking skills might also misuse their techniques. However, we should not define the term hacking by its misuse; 2) cybersecurity is a two-edged sword: offensive and defensive. To be effective at defence, students must fully understand the capabilities of hackers and the way how hackers perform cyberattacks; 3) it is widely believed that including both perspectives of "defender" and "attacker" and the related skills could make the cybersecurity curriculum more meaningful and practical [2]. On the other hand, teaching hacking skills can make cybersecurity professionals be equipped with offensive techniques, and well prepared to defend their computing and network system; 4) regardless of teaching hacking skills or not, hackers were out there, and will still be out there. Should hacking skills be integrated into cybersecurity curricula, it would be possible to promote conscious ethical practices and minimize the likelihood that students would misuse the skills.

2.2. Challenging to Integrate the Techniques to a 3-Credit Hours Course

What techniques should be selected to train our students with cybersecurity skills, as there are tons of approaches that have been proposed to detect stepping-stone intrusion since 1995? In a regular course with 48 academic credit hours, it is infeasible to cover all the techniques developed so far, but we do want to train our students not only to have an overall picture of the techniques on stepping-stone intrusion detection, but also to deeply understand some specific and typical intrusion detection approaches. The challenge is to develop contents modules and design hands-on lab exercises. In this paper, we only focus on the designing the hands-on labs on stepping-stone intrusion and its detection. Refer to our prior work [3] for the course modules we developed for integration of detection techniques for stepping-stone intrusion into cybersecurity curricula.

2.3. Challenge on Developing Hands-on Labs of Stepping-stone Intrusion and its Detection

The most difficult part of teaching cybersecurity courses is to design appropriate hands-on labs. We all know the importance of hands-on labs in cybersecurity education. Without the practicing of the techniques covered in cybersecurity class, it is hard to make our students to digest the cybersecurity skills. Conducting cybersecurity hands-on labs needs hardware and software that are more likely not free. Most colleges are equipped with good hardware, such as computers, routers, switches, and different type of servers, but lack of appropriate software. One reason is that some software helping students to practice cybersecurity skills are usually not free, and may be extremely expensive, such as Cyber-range, its price can be as high as more than one million dollars. Therefore, the challenge is how to design appropriate hands-on labs not only can help students to practice stepping-stone intrusion and its detection techniques, but also can reduce the cost to make labs affordable to most colleges.

3. Survey of the Techniques on Stepping-stone Intrusion and its Detection

Many methods have been proposed to detect stepping-stone intrusion. In [4], the authors proposed a thumbprint method to detect stepping-stone intrusion in 1995. This method was developed to compare the contents of TCP/IP packets from the incoming and outgoing sessions of a computer that is chosen to be the sensor for detection. In [5], the authors proposed a detection approach for stepping-stone intrusion by considering the time gaps between the packets captured from the outgoing connection and the incoming connection from a host. In [6], the authors proposed another method for stepping-stone intrusion detection. Their method did not follow the idea of using time-based thumbprints. Instead, the authors in [6] used the deviation between the incoming and outgoing sessions of a computer.

After 2000, a lot more methods were proposed for stepping-stone intrusion detection. One popular approach is to compare the number of packets from the incoming connection with that from the outgoing connection. For the details of this type of approach, please refer to the references [7-9]. A watermark correlation technique was proposed for stepping-stone intrusion detection [10-12]. The idea of using a watermark in stepping-stone intrusion detection is to insert a watermark in the incoming connection of a detection sensor, and then pay attention to the outgoing connections to see if the same watermark can be found in any of these outgoing connections. The rationale used in the papers [10-12] is to analyse and compare the incoming and outgoing connections of a sensor to see if there is any relayed pair. A sensor is defined as a computer host in which all the packets are captured and a detection program runs. If an incoming connection of a sensor is relayed with an outgoing connection, the sensor is considered as a stepping-stone host. However, a user might sometimes use a host as a stepping-stone legitimately due to some special applications. If so, the watermark approach discussed in [10-12] for stepping-stone intrusion detection may produce false positive errors, since this method simply compares an incoming connection with an outgoing one.

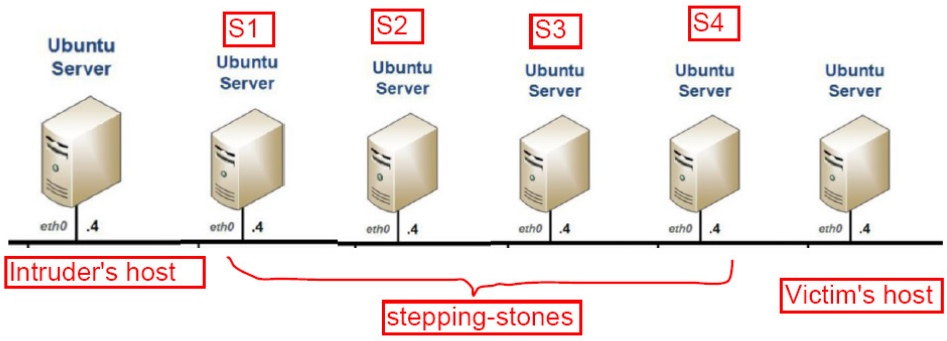


Figure 1: Four Stepping-stone Network Topology

A significant research conducted in [5] has shown that very few professional software employs three or more stepping-stones to access a remote server, although certain legal applications may utilize one or two stepping-stones to access a remote server. Therefore, in order to produce smaller false-positive errors to detect stepping-stone intrusion, an effective method is to estimate the length of a connection chain of stepping-stones. It is extremely challenging to estimate the length of an upper stream connection chain (from the attacker's host to the sensor in the connection chain). Thus, it is impossible to estimate the length of a whole connection chain. By far, most proposed approaches in the literature could only calculate the length of the downstream connection chain (from the sensor to the victim host). This approach to estimate the length of a downstream detection chain was investigated first in [13].

In [13], the authors studied the ratio between the Ack-RTT value and the Echo-RTT. Ack-RTT is defined as the gap between the time to send a packet out and the time to receive its corresponding acknowledgement packet. Echo-RTT is defined as the gap between the time to send a packet out and the time to receive its echo packet. In this way, the length of a downstream connection chain can be approximately estimated. However, this approach could incur false-negative errors.

In [14], the authors proposed a step-function approach motivated by the work that was done in [13] with the purpose of more accurately calculate the length of a downstream connection chain. In [15], the authors proposed another approach by mining network traffic to estimate the number of stepping-stones of a downstream connection chain in 2007. A couple of other methods were also developed in recent years for stepping-stone intrusion detection, including the method using the RTT-based random walk [16], and the method using the idea of RTT Cross-Matching [17].

The stepping-stone intrusion detection approaches have been investigated for about twenty-five years since 1995, unfortunately by far, these important methods have not yet been integrated into cybersecurity curricula at the college level in the U.S. It is vital to educate learners about the known detection approaches for stepping-stone intrusion as more and more professional attackers tend to launch their cyberattacks by using a chain of stepping-stones. Most universities/colleges' professors support to teach the skills and topics of ethical hacking and integrate them into the cybersecurity curricula due to two reasons. First, as far as we know, very few well-educated college students became malicious intruders; second, teaching offensive skills of ethical hacking for college students may produce more and more well-qualified professionals of cybersecurity workforce [18]. We propose ten hands-on labs that allow students to practice in various stepping-stone intrusion detection topics and help them better understand the topics included in the well-designed cybersecurity modules. These hands-on labs will also help enhance students' learning engagement significantly and greatly improve their hands-on experience in cybersecurity.

4. Hands-on Lab Development

Five modules for students to study stepping-stone intrusion and its detection techniques have been proposed and integrated into cybersecurity curriculum [3]. In these five modules, the most popular and the most recently developed techniques have been included. In order to help students to digest the detection and prevention techniques included in the five modules quickly and thoroughly, we design ten hands-on labs as the following,

1) setting up a stepping-stone intrusion connection chain;

2) capturing network traffic;

3) make C# code to capture network traffic;

4) content-based thumbprint detection;

5) time-based thumbprint detection;

6) step-function detection;

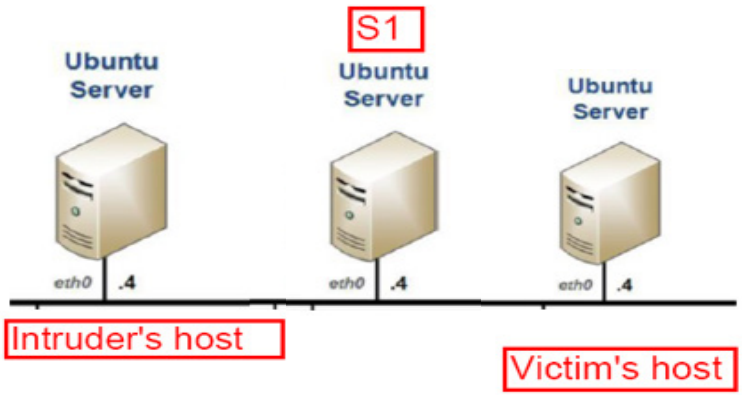
7) packet matching;

8) RTT-based random-walk detection;

9) estimating the length of a long connection chain;

10) intrusion detection using crossover packets.

We apply two rules including relevance and affordability to examine each hands-on lab developed. Relevance means if the lab is closely tied to the modules developed. Affordability means all the labs designed do not use expensive hardware and software. An ideal scenario is that students only need to use the Internet, and free download software to conduct the labs designed.



This designing rule can make it possible for most teaching-focus colleges/universities to offer the labs to cybersecurity majors. Depending on the curriculum design in different institutions, it is not necessary to adopt all the ten labs. However, Lab 1 and Lab 2 are not optional. All the computer hosts used in each lab must be connected in a local area network (LAN). Student must have login credential for each host. All the following labs share the same lab setup as below,

Hardware:

- Each computer must have minimally 4G memory and 500G hard drive capacity.
- Wired or Wireless computer network connection.

Software:

- Ubuntu server or any other type of Linux/Unix installed in each host.
- SSH/OpenSSH client side tool must be installed.
- Each host must have SSH server installed.
- Wireshark, or TcpDump

Login Credentials:

- User Name: Student (Assumed)
- Password: cpsc4166 (Assumed)

All the labs proposed in this paper need students to make a connection chain and to capture TCP/IP packets. A connection chain can be established using OpenSSH under Linux OS which can be a physically installed, or virtual one, such as an OS from VirtualBox, or VMware. It does not need too much memory and second storage. We tried computers with different memory sizes and storage capacity, and found that 4G memory and 500G storage are the minimized requirements. As for the software, TcpDump/Wireshark, SSH client and SSH server package are required minimally.

4.1. Setting up a Stepping-stone Intrusion Connection Chain

4.1.1 Lab objectives

1. Understand TCP/IP protocol; 2. Know how to establish a long interactive connection chain spanning multiple hosts; 3. Understand the concept of Stepping-stones; 4. Obtain the knowledge how an intruder lunches attacks over stepping-stones.

4.1.2 Network topology

It is the same topology as shown in Figure 1.

4.1.3 Lab instructions

1) Start up from any computer in the LAN, and login to a computer that is assumed the Intruder's host with the above credentials.
2) Please open a terminal at the Intruder's host.
3) Browse the current folder, and take a screenshot for the files in the folder.
4) Run SSH to connect to a local host S1: ssh Student@S1 (this can also be the IP address of S1 if host name S1 is not known) in the LAN.
5) As long as connecting to S1, you are prompted to input the password for the user.
6) If connected to S1 successfully, please browse the current folder, and take a screenshot including the folder's name, and all the files in the current folder. Run "ifconfig" to show the IP address and other network related information of S1. Take a screenshot of "ifconfig" results.
7) Compare the screenshot taken at the Intruder's host with the one taken at S1 to see if they are the same.
8) Repeat steps 4), 5), 6) 7) to connect to the computer hosts S2, S3, S4, and the last one respectively. The last host connected is called Victim's host.
9) So far you have locally connected to Victim's host via the hosts S1, S2, S3, and S4. Hosts S1, S2, S3, and S4 are used as stepping-stones.
10) If sniffing the packets at Victim's host, we can see all of the packets are from host S4 other than Intruder's host even though we know all the packets come from the Intruder's host originally. So in this way, intruders can protect themselves via the compromised hosts, such as the hosts S1, S2, S3 and S4.
11) Logout from Victim's host to S4 by typing "Exit" at Victim's

host.

12) Browse the current folder and compare with the screenshot taken at host S4 to see if it is disconnected from Victim's host.

13) Repeat steps 11) and 12) until come back to Intruder's host.

4.1.4 Critical Thinking Practice

1) Ethical Issue Discussion:
Please discuss if there are any ethical issues by making a connection chain across the Internet using legal credentials. How about is it by using illegal credentials?
2) Why do intruders make use of stepping-stones?
3) An interactive session can be encrypted by using SSH. Is it possible to get source IP and destination IP if a TCP/IP packet is captured from such a session? If yes, please tell how? If No, please tell why?
4) Compare to directly access a victim host, is it efficient to access the victim host via some compromised hosts?
5) In the lab, it has five connections in the long interactive session from Intruder's Host to Victim's Host. Each connection is encrypted and set up by using SSH/OpenSSH. Is the encryption key used for the connection from Intruder's Host to S1 the same as the encryption key used for the connection from S1 to S2? Why?

4.2. Capturing Network Traffic

4.2.1 Lab objectives

1. Understand the meaning of each field of a TCP/IP packet header; 2. Know how to store captured packets into different files; 3. Understand the features of TCP, UDP, IP, and ICMP packets; 4. Learn how to use Wireshark to capture network traffic.

4.2.2 Network topology

Refer to Figure 2.

4.2.3 Lab instructions

1) Select any three computer hosts in your local area network, and login to each host with the credentials given.

2) Run "ifconfig" to get the IP address at the three computers respectively and take a screenshot at each host.

3) Follow the instructions in Lab 1 to set up a connection chain as shown in Figure 2. This connection chain spans three computer hosts including Intruder's host, S1, and Victim's host.

4) Type some Linux/Unix commands at Intruder's host to make network traffic from Intruder's host to Victim's host via S1.

5) At S1, run Wireshark to capture TCP packets coming from Intruder's host and leaving to Victim's host only.

6) Store all the packet in Step 5) to a readable file (text file) including timestamp, source IP, destination IP, source Port number, destination Port number, Sequence number, Acknowledgement number, Flag, and Length.

7) At S1, run Wireshark to capture TCP packets coming from Victim's host and going to Intruder's host only. Repeat Step 6).

8) Repeat Steps 5), 6) and 7), but capture UDP packets.

9) Repeat Steps 5), 6) and 7), but capture ICMP packets.

4.2.4 Critical Thinking Practice

1) Ethical Issue Discussion:
2) Would it trigger any ethical issue to capture other users' network traffic under a host with legal login?
3) What is the difference between Display filter and Capture filter in Wireshark?
4) Give the display filter to find the packets of three-way handshake for a connection from host 192. 168.0.1.
5) What is a TCP Send packet?

4.3. Making a Code to Capture Network Traffic

4.3.1 Lab objectives

1. Understand LibPacp package for Linux server; 2. Learn the algorithms to capture computer network traffic; 3. Be able to make C code to capture TCP/IP Packets; 4. Obtain the knowledge to detect network adapters, and open an adapter; 5. Understand the techniques to set up and compile a packet-capturing filter.

4.3.2 Network topology

It has the same network topology as Figure 2 in Lab 4.2.

4.3.3 Mechanism on making the code to sniff network traffic

In order to make a code to capture network packets like what Wireshark does, Libpcap package must be installed in the Ubuntu server. If Windows server is used, please install WinPcap. The way to make a code to sniff computer network traffic is to call the functions built in Libpcap (packet capture) package. Libpcap provides an application-programming interface (API) for capturing network traffic.

We take an example, capturing raw IP packets, to examine the steps to sniff packets by making a program under Linx/Unix system. For the details of the code, please refer to the reference [19]. It has four steps to sniff computer network packets: 1) open a packet capture socket; 2) start packet capture loop; 3) parse and display packets; 4) Terminate capture program.

Open a packet capture socket: A socket is an endpoint for network communication that is identified in a program with a socket descriptor. Opening a packet capture socket involves a series of Libpcap calls that are encapsulated in open_pcap_socket() function. There are a couple of steps needed to open a packet capture socket. The first step is to select a network device using function pcap_lookupdev(). The second step is to open the network device selected for live capture using function pcap_open_live(). The third step is to call function pcap_lookupnet() to get the network address and subnet mask. The fourth step is to compile a packet capture filter by calling function pcap_compile(). The last step is to install the compiled packet filter program into the packet capture device. This causes

```c
pcap_t* open_pcap_socket(char* device, const char* bpfstr)
{
  char errbuf[PCAP_ERRBUF_SIZE];
  pcap_t* pd;
  uint32_t srcip, netmask;
  struct bpf_program bpf;

  // If no network interface (device) is specified, get the first one.
  if (!*device && !(device = pcap_lookupdev(errbuf)))
  {
    printf("pcap_lookupdev(): %s\n", errbuf);
    return NULL;
  }
  // Open the device for live capture, as opposed to reading a packet
  // capture file.
  if ((pd = pcap_open_live(device, BUFSIZ, 1, 0, errbuf)) == NULL)
  {
    printf("pcap_open_live(): %s\n", errbuf);
    return NULL;
  }
  // Get network device source IP address and netmask.
  if (pcap_lookupnet(device, &srcip, &amp;netmask, errbuf) < 0)
  {
    printf("pcap_lookupnet: %s\n", errbuf);
    return NULL;
  }
  // Convert the packet filter expression into a packet filter binary.
  if (pcap_compile(pd, &bpf, (char*)bpfstr, 0, netmask))
  {
    printf("pcap_compile(): %s\n", pcap_geterr(pd));
    return NULL;
  }
  // Assign the packet filter to the given libpcap socket.
  if (pcap_setfilter(pd, &bpf) < 0)
  {
    printf("pcap_setfilter(): %s\n", pcap_geterr(pd));
    return NULL;
  }
  return pd;
}
```

(a)

```c
void capture_loop(pcap_t* pd, int packets, pcap_handler func)
{
  int linktype;
  // Determine the datalink layer type.
  if ((linktype = pcap_datalink(pd)) < 0)
  {
    printf("pcap_datalink(): %s\n", pcap_geterr(pd));
    return;
  }
  // Set the datalink layer header size.
  switch (linktype)
  {
  case DLT_NULL:
    linkhdrlen = 4;
    break;
  case DLT_EN10MB:
    linkhdrlen = 14;
    break;
  case DLT_SLIP:
  case DLT_PPP:
    linkhdrlen = 24;
    break;
  default:
    printf("Unsupported datalink (%d)\n", linktype);
    return;
  }
  // Start capturing packets.
  if (pcap_loop(pd, packets, func, 0) < 0)
    printf("pcap_loop failed: %s\n", pcap_geterr(pd));
}
```

(b)

```c
void parse_packet(u_char *user, struct pcap_pkthdr *packethdr,
          u_char *packetptr)
{
  struct ip* iphdr;
  struct icmphdr* icmphdr;
  struct tcphdr* tcphdr;
  struct udphdr* udphdr;
  char iphdrInfo[256], srcip[256], dstip[256];
  unsigned short id, seq;
  // Skip the datalink layer header and get the IP header fields.
  packetptr += linkhdrlen;
  iphdr = (struct ip*)packetptr;
  strcpy(srcip, inet_ntoa(iphdr->ip_src));
  strcpy(dstip, inet_ntoa(iphdr->ip_dst));
  sprintf(iphdrInfo, "ID:%d TOS:0x%x, TTL:%d IpLen:%d DgLen:%d",
      ntohs(iphdr->ip_id), iphdr->ip_tos, iphdr->ip_ttl,
      4*iphdr->ip_hl, ntohs(iphdr->ip_len));
  packetptr += 4*iphdr->ip_hl;
  switch (iphdr->ip_p)
  {
  case IPPROTO_TCP:
    tcphdr = (struct tcphdr*)packetptr;
    printf("TCP  %s:%d -> %s:%d\n", srcip, ntohs(tcphdr->source),
        dstip, ntohs(tcphdr->dest));
    printf("%s\n", iphdrInfo);
    printf("%c%c%c%c%c%c Seq: 0x%x Ack: 0x%x Win: 0x%x TcpLen: %d\n",
        (tcphdr->urg ? 'U' : '*'),
        (tcphdr->ack ? 'A' : '*'),
        (tcphdr->psh ? 'P' : '*'),
        (tcphdr->rst ? 'R' : '*'),
        (tcphdr->syn ? 'S' : '*'),
        (tcphdr->fin ? 'F' : '*'),
        ntohl(tcphdr->seq), ntohl(tcphdr->ack_seq),
        ntohs(tcphdr->window), 4*tcphdr->doff);
    break;
```

(c)

Figure 3: Packet Capture Sample Code

Libpcap to start collecting the packets with selected filter. The sample code in Figure 3-(a) shows the four steps in opening a packet capture socket.

Start packet capture loop: Libpcap provides three functions to capture packets: pcap_next(), pcap_dispatch(), and pcap_loop(). Since function pcap_next() can only grab one packet at the time to be called. So the program must call this function in a loop to receive multiple packets. The other two functions pcap_loop and pcap_dispatch() can loop automatically to receive multiple packets. Datalink type can be determined by calling pcap_datalink(), and then start packet capture. The sample program shown in Figure 3-(b) uses pcap_loop() to sniff multiple packets. In this code, first to determine the datalink type by calling pcap_datalink(), and then start packet capture loop.

Parse and display packets: The general technique for parsing packets is to set a character pointer to the beginning of the packet buffer then advance this pointer to a particular protocol header by the size in bytes of the header that precede it in the packet. The header can then be mapped to an IP, TCP, UDP, and ICMP header structure by casting the character pointer to a protocol specific structure pointer. A parse_packet() function starts off by defining pointers to IP, TCP, UDP and ICMP header structures. The packet pointer is advanced past the datalink header by the number of bytes corresponding to the datalink type determined in capture_loop(). Casting the packet pointer to struct tcphdr and struct udphdr pointers gives us access to TCP and UDP header fields respectively. The struct icmphdr pointer enables us to display ICMP packet type and code along with the source and destination IP addresses. The sample code in Figure 3-(c) shows the steps to parse and display packets, such as TCP packets that are used to detect stepping-stone intrusion.

Terminate Capturing: The last step is to terminate the packet capture by interrupt signals SIGNIT, SIGTERM, and SIGQUIT through calling function bailout() which displays the packet count, closes the packet capture socket then exits the program.

4.3.5 Lab instructions

1) Start up running your code, and select the interface to sniff
2) Click “Start” button to start packet sniffing
3) Display the following information for each packet captured: source/destination IP address, source/destination port number, packet type, sequence number, acknowledge number, TCP flags, fragmentation information, checksum, receive window, TTL, upper layer protocol, timestamps in format of mm/dd/yy.
4) Click one TCP/IP packet captured to show the details in each of its header field. Take a screenshot for the header details.
5) Store captured packet in a .txt file that can be opened by WordPad, or any other text editor tool.

4.3.4 Critical Thinking Practice

1) Ethical Issue Discussion:
   Would it trigger any ethical issue to capture other users’ network traffic using self-made code under a host with legal login?
2) What is the difference between Winpcap and Libpcap?

3) What functions are called in order to open a packet capture socket?

4) What is the purpose to call pcap_compile()?
5) What is the function of pact_next()?
6) Which function is called to determine the datalink type of a packet?

4.4. Content-based Thumbprint Detection

4.4.1 Lab objectives

1. Understand TCP/IP protocols and network traffic behaviour; 2. Know how to establish an interactive TCP session; 3. Understand using Thumbprint to detect Stepping-stone intrusion; 4. To be familiar with TcpDump and Wireshark.

4.4.2 Network topology

The network topology used in this lab is the same as Figure 2 in Lab 4.2.

4.4.3 Lab instructions

1) Select any three computers in your local area network and name them to be Intruder's host, S1, and Victim's host.

2) Start up the computers in Linux and login to each host with given credentials. Open a terminal in each host.

3) Run "ifconfig" to get the IP address for each host, and take a screenshot from each host.

4) Run SSH from Intruder's host to connect to S1, then to Victim's host just as shown in Figure 2. An interactive session is set up spanning three hosts with S1 working as a Stepping-stone.

5) Students will monitor the traffic of the incoming connection from Intruder's host, and the traffic of the outgoing connection to Victim's host from S1. Here we use the number of TCP packets to represent the corresponding network traffic.

6) Run TcpDump at host S1 to monitor the TCP packets coming to/from Intruder's host but to S1 with destination/source port 22 and store all the packets in IncomingTCP.txt, and also monitor the TCP packets going to Victim's host or come back to S1 with destination/source port 22, and store all the collected packets to OutgoingTCP.txt.

7) In either IncomingTCP.txt or OutgoingTCP.txt, each packet is stored in one row including the following fields separated by ";": Packet Order number; Timestamp; Source IP; Destination IP; Source Port; Destination Port; Flag; Sequence Number; Acknowledge Number; Packet Length

8) Keep operating at Intruder's host for about 15 minutes to make network traffic to Victim's host via S1.

9) Count the number of packets in the two files respectively by counting the number of rows, or just simply check the last row "Packet Order number" field.

10) Compare the two number to see if they are close enough.

11) Identify the Send and Echo packets in the two files. Count the number of Send and Echo packets from IncomingTCP.txt, and denote them as In-S and In-E respectively. Similarly count the number of Send and Echo packets from OutgoingTCP.txt, and denote them as Out-S and Out-E respectively.

12) The rules to determine Send or Echo packet at S1 are as the following,

    a. Send packet is a packet in the incoming link that comes to S1 with Flag.P set up, but in the outgoing link that leaves S1 to Victim's host with Flag.P set up;

    b. Echo packet is a packet in the incoming link that leaves S1 to Intruder's host with Flag.P set up, but in the outgoing link that comes to S1 with Flag.P set up.

13) Compare if the following relation maintains,

    a. In-S is close to Out-S, and

    b. In-E is close to Out-E, and

    c. The sum of In-S and In-E is close to the sum of Out-S and Out-E

14) Please draw your conclusion based on the results from Steps 10) and 13).

4.4.4 Critical Thinking Practice

1) Ethical Issue Discussion:

   If a user has a legal login to a host, captures network packets, and obtains the contents of each packet, would the user's action result in an ethical issue?

2) What is the TcpDump command to sniff the packets in the incoming link?

3) What is the TcpDump command to sniff the packets in the outgoing link?

4) What conclusion you can make based on the information you have in step 10) of the Lab Instructions above? Why?

5) What conclusion you can make based on the information you have in step 13) of the Lab Instructions above? Why?

6) Write a TcpDump command to sniff the packets only acknowledge the requests from Intruders' Host at S1.

4.5. Time-based Thumbprint Detection

4.5.1 Lab objectives

1. Understand using time-based thumbprint to detect stepping-stone intrusion; 2. Learn how to generate time-based thumbprint; 3. Know how to compare time-based thumbprint; 4. Understand the efficiency of thumbprint comparison algorithm.

4.5.2 Network topology

The network topology used in this lab is the same as Figure 2 in Lab 4.2.

4.5.3 Lab instructions

1) Refer to Lab 1 to make an interactive TCP session with at least one host in between attacker and victim machines.
2) On either of the machine of your choice except the target, filter the network capture & save the incoming and outgoing packets including timestamp information for each packet through TcpDump.
3) Examine the packets for the incoming connection and look for the timestamp there and list those timestamps in a sequence.
4) Repeat Step 3 but for the outgoing connection
5) For the incoming connection sequence (list) of timestamps, find the difference in neighboring timestamps and list them in a sequence. This can give a sequence of time gaps for this connection. Find difference using the equation: $|p_i - p_{(i+1)}|$, here p_i is the timestamps of i^{th} packet captured.
6) Repeat Step 5 but for the outgoing connection.
7) Compare the two sequences to get a similarity. If the similarity is larger than a predefined threshold, the host is used as a stepping-stone. Otherwise, not.

4.5.4 Critical Thinking Practice

1) Ethical Issue Discussion:
   If a user has a legal login to a host, captures network packets, and but could not obtain the contents of each packet due to encryption, would the user's action result in an ethical issue?
2) Please describe what a time session-based thumbprint is in your own words.
3) Why would an individual want to perform this method to detect a stepping-stone over other methods?
4) Why do we compare the two sequences of time gaps in our own algorithm as oppose to the Longest Common Subsequence algorithm which can also help to measure similarity?
5) Do you have a better method of comparing the sequences' similarity?
6) Would a time session-based thumbprint be effective with an encrypted connection? If yes, explain why.

4.6. Step-function Detection

4.6.1 Lab objectives

1. Understand packet matching algorithm: First-Match; 2. Learn how to use matched Send and Echo packets to determine the number of compromised hosts; 3. Demonstrate Step-Function algorithm; 4. Illustrate the limits of Step-function detection.

4.6.2 Network topology

The network topology used in this lab is the same as the one shown in Figure 1 of Lab 4.1.

4.6.3 Lab instructions

1) Start up with any computers in the LAN, and login to the Intruder's host, Victim's host, S1, S2, S3, and S4 with the appropriate credentials to make a connection chain.
2) Open a terminal on Intruder's host and S1.
3) On desired sensor host (S1 for initial run), start TcpDump to dump captured packets to a file along with any further options
   a) ###.###.###.###.X is Sensor's IP Address and X is a port number
   b) sudo TcpDump 'tcp[tcpflags] & tcp-push != 0 and host ###.###.###.###.X' -n --number > capturedFile
4) On Intruder's host, Run SSH to connect to a remote host S1: ssh Student@S1 (this can also be the IP address of S1 if host name S1 is not known).
5) As long as S1 is reachable, you will be prompted to input the password for the user "Student".
6) On Intruder's host, repeat steps 4 and 5 replacing S1 with S2, S3, S4, and Victim's host, respectively, to login to further hosts as needed.
7) Interact with Victim's host: browse directories, manipulate files, check available interfaces, etc.
8) End current SSH session and stop TcpDump on the sensor host.
9) Repeat steps 3-8 for multiple setups; such as two/three stepping-stones chains with the sensor on different steps each time
10) You may want to use grep to create two files: one for Send packets and one for Echo. Consider that [^\2\]{2,} matches 22 for SSH
    a) (grep -E '>/b###.###.###.###.[^\2\]{2,}'/bcapturedFile) > downEchoFile
    b) (grep -E '###.###.###.###.[^\2\]{2,}/b>'/bcapturedFile) > downSendFile
11) Use First-Match Algorithm to match Send/Echo Packets:

a) Iterate through both lists, starting with the lowest sequence numbered Send Packet

b) If the current packet is a Send, add it to a list of unmatched Send packets

c) If it is an Echo and there is at least one unmatched Send Packet, Search the list of unmatched Send packets from the beginning. Find the first send packet with an appropriate acknowledgement number [Echo.Seq == Send.Ack].

d) Use the absolute difference between the correct Echo's and Send's timestamps to determine the round trip time (RTT) of the request [RTT = |Echo.Timestamp – Send.Timestamp|]

e) Save RTT to a list of RTTs

f) If it is an Echo and all preceding Send packets have been matched, the algorithm fails. Check if a packet was missed, then try to determine what may have occurred.

12) Sketch the graph of RTT vs. Number of matched Packets

a) RTT in whatever unit of time (typically ms or μs);

b) Number of matched packets indexed from 1 to the number of matches.

4.6.4 Critical Thinking Practice

1) Ethical Issue Discussion:

If a user has a legal login to a host, captures network packets, obtains the round-trip time between matched Send and Echo packets, but could not identify the contents of each packet due to encryption, would the user's action result in an ethical issue?

2) What is the purpose of tcp-push != 0 in the above capture?

3) Explain the difference in the grep statements listed above. Why does the first point to Send packets, while the second points to Echo packets?

4) Did you notice any effects to performance (positive/negative) as more links were introduced to the connection chain? Explain.

5) Would there be any difference to this analysis if the data were clear text, sent using Telnet, or encrypted like in SSH? Justify.

6) Can you determine the length of the entire connection chain with this method? If so, explain why. If not, which portion can you determine the length?

4.7. Packet Matching

4.7.1 Lab objectives

1. Understand the significance of packet matching; 2. Determine the differences in the different packet matching algorithms; 3. Learn how to apply packet matching to detect stepping-stone intrusion; 4. Distinguish the limits of different packet matching algorithms.

4.7.2 Network topology

The network topology used in this lab is the same as the one shown in Figure 2 of Lab 4.2.

4.7.3 Lab instructions

1) Start up any computers in the LAN, and login to the computer, which assumes to be called Intruder's host with the above credentials.

2) On desired sensor host (S1 for initial run), start TcpDump to dump captured packets to a file along with any further options

a) ###.###.###.###.X is Sensor IP Address and X is port number

b) sudo TcpDump 'tcp[tcpflags] & tcp-push != 0 and host ###.###.###.###.X' -n --number > capturedFile

3) Make an SSH connection chain from Intruder's host through any stepping-stone saying host S1 (sensor) to Victim's host.

4) Interact with Victim's host from Intruder's host via the connection chain: browse directories, manipulate files, check available interfaces, etc.

5) Terminate the SSH chain by using the 'exit' command on each of the stepping-stones and Victim's host from the shell of Intruder's host

6) You may want to use grep to create two files: one for Send packets and one for Echo

a) Upstream
   i. (grep '###.###.###.###.X/b>'/b</bcapturedFile) > upEchoFile
   ii. (grep '>/b###.###.###.###.X'/b</bcapturedFile) > upSendFile

b) Downstream – consider that [^\2\]{2,} matches 22 for SSH
   i. (grep -E '>/b###.###.###.###.[^\2\]{2,}'/bcapturedFile) > downEchoFile
   ii. (grep -E '###.###.###.###.[^\2\]{2,}/b>'/bcapturedFile) > downSendFile

7) Use First-Match Algorithm to match Send/Echo Packets:

a) Iterate through both lists, starting with the lowest sequence numbered Send Packet

b) If the current packet is a Send, add it to a list of unmatched Send packets

c) If it is an Echo and there is at least one unmatched Send Packet, Search the list of unmatched Send packets from the beginning. Find the first send packet with an appropriate acknowledgement number [Echo.Seq == Send.Ack].

d) Use the absolute difference between the correct Echo's and Send's timestamps to determine the round trip time (RTT) of the request [RTT = $|Echo.Timestamp - Send.TimeStamp|$]

i. Save RTT to a list of RTTs

e) If it is an Echo and all preceding Send packets have been matched, the algorithm fails. Check if a packet was missed, then try to determine what may have occurred.

8) Use the Conservative Algorithm to match Send/Echo Packets:

a) Iterate through both lists, starting with the lowest sequence numbered Send Packet

b) If the current packet is a Send:

i. If previous packet was Send and time gap was 1 second or more, clear the sendQ and make a note of match-flag = true

ii. Otherwise, add it to a list of unmatched Send packets

c) If it is an Echo:

i. If there is at least one unmatched Send Packet and match-flag = true, search the list of unmatched Send packets from the beginning. Find the first send packet with an appropriate acknowledgement/sequence number [Echo.Seq == Send.Ack && Echo.Ack > Send.Seq].

1. Use the absolute difference between the correct Echo's and Send's timestamps to determine the round trip time (RTT) of the request [RTT = $|Echo.Timestamp - Send.TimeStamp|$]

a. Save RTT to a list of RTTs

ii. Otherwise, set match-flag = false

9) Use the Greedy Heuristic Algorithm to match Send/Echo Packets:

a) Iterate through both lists, starting with the lowest sequence numbered Send Packet

b) If the current packet is a Send:

i. If previous packet was Send and time gap was 1 second or more, clear the sendQ

ii. Otherwise, add it to a list of unmatched Send packets

c) If it is an Echo:

i. If there is at least one unmatched Send Packet, search the list of unmatched Send packets from the beginning. Find the first send packet with an appropriate acknowledgement/sequence number [Echo.Seq == Send.Ack && Echo.Ack > Send.Seq].

1. Use the absolute difference between the correct Echo's and Send's timestamps to determine the round trip time (RTT) of the request [RTT = $|Echo.Timestamp - Send.TimeStamp|$]

a. Save RTT to a list of RTTs

ii. Otherwise, no match detected.

4.7.4 Critical Thinking Practice

1) Ethical Issue Discussion:
If a user has a legal login to a host, captures network packets, matches each Send packet with its corresponding Echo, but could not identify the contents of each packet due to encryption, would the user's action result in an ethical issue?

2) Which two TCP packet types can we exploit to properly match packets within a connection?

3) Explain why Echo.Seq == Send.Ack is used.
4) Explain why Echo.Ack > Send.Seq is used.
5) Why does Conservative Algorithm clear the Send Queue?
6) Looking at the results of running through the algorithms, what differences do you see between them? Explain why that might be.

4.8. RTT-based Random-walk Detection

4.8.1 Lab objectives

1. Understand random-walk model; 2. Learn how to apply random-walk model to detect stepping-stone intrusion; 3. Be familiar with the techniques to evade detection; 4. Demonstrate using RTT to resist intruders' evasion.

4.8.2 Network topology

The network topology used in this lab is the same as the one shown in Figure 2 of Lab 4.2.

4.8.3 Lab instructions

1) Refer to Lab 1 to make an interactive TCP session including at least one stepping –stone host that is used as a sensor.

2) On the sensor, filter the network capture & save the incoming and outgoing packets through TcpDump.

3) Examine the packets for the incoming connection, and match the Send & Echo packets using conservative packet matching algorithm from Lab 4.7, and obtain the number of RTTs from matched packets for this connection, N^{RTT}_{in}.

4) Repeat Step 3) for the packets collected from the outgoing connection, and obtain N^{RTT}_{out}.

5) Take the difference of N^{RTT}_{in} and N^{RTT}_{out}. $N^{RTT}_{in\text{-}out} = |N^{RTT}_{in} - N^{RTT}_{out}|$

6) Compare $N^{RTT}_{in\text{-}out}$ to a predefined upper bound. If it is less than the upper bound, then the incoming & outgoing connections are a relayed pair. The sensor is used as a stepping-stone. If not then, the machine is not used as a stepping-stone.

4.8.4 Critical Thinking Practice

1) Ethical Issue Discussion:
If a user has a legal login to a host, captures network packets, obtains the round-trip time between matched Send and Echo packets, but could not identify the contents of each packet due to encryption, would the user's action result in an ethical issue?
2) Please describe how a RTT-based Random-Walk Detection works in your own words.
3) Why would an individual want to perform this method to detect a stepping-stone over other methods?
4) Could an intruder manipulate this approach to give a false negative?
5) Would this method be effective with an encrypted connection? If yes, explain why.

6) Perform a network capture by following the above instructions with the predefined threshold, Γ, being equal 30. From the results, is the machine a stepping-stone?

4.9. Detection by Estimating the Length of a Long Connection Chain

4.9.1 Lab objectives

1. Understand the RTTs of the packets from the same connection chain can be mined to the same cluster; 2. Learn the number of compromised hosts is equal to the number of outstanding clusters; 3. Demonstrate the approach to estimate the length of a connection chain; 4. Obtain the knowledge on how clustering-partitioning algorithm can resist intruders' evasion.

4.9.2 Network topology

The network topology used in this lab is the same as the one shown in Figure 1 of Lab 4.1.

4.9.3 Lab instructions

1) Start up any computers in the LAN, and login to the computer that assumes to be called Intruder's host with the above credentials.
2) We will use at least 5 hosts in this connection chain. Decide which 5 hosts you want to use, and designate the 2nd host as a sensor host
3) On the sensor host, begin packet capture prior to making any of the connections.
4) Please open a terminal at Intruder's host.
5) Run SSH to connect to a remote host S1 (sensor host): ssh Student@S1 (this can also be the IP address of S1 if host name S1 is not known).
6) As long as connected to S1, you must be prompted to input the password for the user.
7) Repeat steps 4), 5), to connect to computer hosts S2, S3, S4, and the last one respectively. The last host you connect to remotely is called Victim's host.
8) So far you have remotely connected to Victim's host spanning hosts S1, S2, S3, and S4. Hosts S1, S2, S3, and S4 are used as stepping-stones in this lab.
9) Generate traffic to be captured by sensor. (ls, pwd, etc.)
10) After complete the packet capture, analyse the packets captured using clustering-partitioning algorithm. For the algorithm details, please refer to the reference [20].

4.9.4 Critical Thinking Practice

1) Ethical Issue Discussion:
   If a user has a legal login to a host, captures network packets, obtains the round-trip time between matched Send and Echo packets, and can estimate how many connections between the current host and the end of the connection chain. If the user could not identify the contents of each packet due to encryption, would the user's action result in an ethical issue?
2) Why is it important to begin packet capture before you initiate the connection chain? Please explain.
3) What results are we looking for after completing the clustering-partitioning algorithm? Why do these results indicate connections?
4) What is the maximum theoretical complexity of the partitioning clustering algorithm? Why is this algorithm likely never reach this complexity level? Please explain.
5) What percentage of the RTTs should be within a cluster to be considered a valid cluster?
6) If we collected 720 send packets and 810 echo packets, at most, how many comparisons would be necessary for partitioning-clustering algorithm?

4.10. Detection Using Crossover Packets

4.10.1 Lab objectives

1. Understand crossover packets; 2. Know the reason of generating crossover packets; 3. Obtain the relation between the length of a connection chain and the number of crossover packets; 4. Learn how to identify crossover packets.

4.10.2 Network topology

The network topology used in this lab is the same as the one shown in Figure 1 of Lab 4.1.

4.10.3 Lab instructions

We assume Intruder's Host is called iHost, and Victim's host is called vHost. After a connection chain is established, please type the following information at iHost to make some network traffic for each of the following: "This is s test from Hands-on lab 10. Please discard all the wrong messages!"

1) Make a connection chain from iHost to vHost via S1 only. Type the above information at iHost and capture Send and Echo packets at S1 from its outgoing connection. Store the packets to PacketFile1.
2) Make another connection chain from iHost to vHost, but via S1 and S2. Type the above information at iHost and capture Send and Echo packets at S1 from its outgoing connection. Store the packets to PacketFile2.
3) Make the third connection chain from iHost to vHost, but via S1, S2, and S3. Type the above information at iHost and capture Send and Echo packets at S1 from its outgoing connection. Store the packets to PacketFile3.
4) Make the fourth connection chain from iHost to vHost, but via S1, S2, S3, and S4. Type the above information at iHost and capture Send and Echo packets at S1 from its outgoing connection. Store the packets to PacketFile4.
5) Count the number Crossover packets in each file and compare them. Please conclude what you would find from the comparing the results.

4.10.4 Critical Thinking Practice

1) Ethical Issue Discussion:
   If a user has a legal login to a host, captures network packets, obtains the crossover packets, but could not identify the contents of each packet due to encryption, would the user's action result in an ethical issue?
2) Why is it unlikely that you will observe much, if any, Crossover in a LAN environment?
3) Does increasing the connection chain length increase or decrease the likelihood of observing packet Crossover? Why or why not?
4) Does packet Crossover help or hinder packet matching? Why?
5) Why are you more likely to observe packet Crossover in a WAN environment?
6) What information about a connection chain can you gather from detecting many packet Crossovers?

5. Discussion on the Labs Designed

In this session, we will discuss the innovation, contribution, and the effectiveness of the proposed work.

All the hands-on labs were designed based on some research papers. To the best of our knowledge, the is the first time that stepping-stone intrusion detection techniques are integrated into cybersecurity curriculum. The contribution is that college students can learn complex stepping-stone intrusion detection techniques and enhance their experience by conducting the hands-on labs. The labs designed are suitable for teaching-focus colleges who may have limited budget for their cybersecurity curriculum.

Each lab proposed has a critical thinking practice component including discussions about ethical issues, and the questions to train students to be qualified professionals of cybersecurity workforce. Most of the labs proposed were adopted in the course of "Intrusion Detection and Prevention" at Columbus State University, GA from 2018 to 2019. The instructors did class survey to ask the students if they agree with the labs adopted for the class. The survey results are shown in Table 1.

Table 1: Lab Survey Results

Item / Semester	Strongly Agree	Agree	Neutral	Disagree	Agree and Neutral Rate	Attending Rate
Spring 2018	5	4	3	1	92.3%	13 out of 15 = 86.7%
Spring 2019	11	9	6	0	100%	26 out of 28 = 92.9%
Spring 2020	9	5	2	2	88.88%	18 out of 19 = 94.7%
Spring 2021	11	11	4	2	92.9%	28 out of 29 = 96.6%
Average Rate					93.52%	92.73%

From the survey results, we can see that over four years, more than 90 percent of the students like the labs. Their comments and feedback are positive. There are also some negative comments and feedback. The following are some negative feedback extracted from the surveys:1) the time given to finish the labs are not enough; 2) most students prefer to use a physically installed Linux system to conduct the lab, other than a virtual Linux system because it is hard to copy the results out; 3) too many packets are required to capture which costs their too much time; 4) some students expect to have the first lab to refresh the Linux command, other than to make a connection chain.

6. Summary

In order to help college students to learn stepping-stone intrusion detection and prevention techniques and enhance their hands-on learning experience, we developed ten hands-on labs based on the significant results published in the area of stepping-stone intrusion detection since 1995. For making these hands-on labs be easily adopted by university professors in undergraduate cybersecurity courses, we used the following strategies while designing these hands-on labs: 1) save budgets for learners; 2) simplify the requirements for required hardware and software; 3) clear step-by-step instructions; 4) easy assessments by evaluators; 5) easy adoption by instructors.

Most of the hands-on labs we designed in this paper have been adopted in the undergraduate course of Intrusion Detection and Prevention at Columbus State University for four years. The average survey result shows that more than 90% of the students liked the labs and enjoyed the hand-on activities involved in the labs. The rate of disagreement/dislike is less than 10%. All the hands-on labs have been shared within the USA via the Clark system managed by Towson University, MD, USA. Records show that at least six colleges/universities downloaded the hands-on labs. We highly believe that our proposed hands-on labs in stepping-stone intrusion detection will help building the nation's cybersecurity workforce.

Cybersecurity is a rapidly changing and expending field. In order to make our students to be adaptable with fast changing cybersecurity techniques quickly after graduation, in the future, we will improve the proposed hands-on labs following NICE cybersecurity workforce framework initiated by NIST. In this framework, there are seven categories and each category contains one or more specialty areas. Each cybersecurity specialty area is composed of multiple work roles. Each work role includes Knowledge, Skills and Abilities (KSAs) and Tasks. The future hands-on labs will help our students to achieve three targets. First, they will obtain a body of information, which can be directly applied to the performance of a function. Second, they will enhance their skills needed for cybersecurity. Third, they will improve their competence to perform an observable behavior, which can result in an observable product.

Conflict of Interest

The authors declare no conflict of interest.

Acknowledgment

This work of Drs. Lixin Wang and Jianhua Yang is supported by National Security Agency (NSA) NCAE-C research grant H98230-20-1-0293 with Columbus State University, Columbus GA 31907, USA.

References

[1] P. Logan, A. Clarkson, "Teaching students to hack: curriculum issues in information security," Special Interest Group on Computer Science Education Symposium, St. Louis, MO USA, 2005.

[2] S. Bratus, A. Shubina, M.E. Lacasto, "Teaching the principles of the hacker curriculum to undergraduates," SIGCSE' 10, Milwaukee, Wisconsin USA, 2010.

[3] J. Yang, Y. Zhang, G. Zhao, "Integrate stepping-stone intrusion technique into cybersecurity curriculum," the Proceedings of 31st IEEE International Conference on Advanced Information Networking and Applications, Taipei, Taiwan, published in IEEE proceedings and Digital Library, 1-6, 2017, doi: 10.1109/WAINA.2017.29.

[4] S. Staniford-Chen, L.T. Heberlein, "Holding intruders accountable on the Internet," Proceedings of IEEE Symposium on Security and Privacy, Oakland, CA USA, 39-49, 1995, doi: 10.1109/SECPRI.1995.398921.

[5] Y. Zhang, V. Paxson, "Detecting stepping-stones," Proceedings of the 9th USENIX Security Symposium, Denver, CO USA, 67-81, 2000.

[6] K. Yoda, H. Etoh, "Finding connection chain for tracing intruders," Proceedings of 6th European Symposium on Research in Computer Security, Toulouse, France, Lecture Notes in Computer Science, 31-42, 2000.

[7] A. Blum, D. Song, S. Venkataraman, "Detection of interactive stepping-stones: algorithms and confidence bounds," Proceedings of International Symposium on Recent Advance in Intrusion Detection, Sophia Antipolis, France, 20-35, 2004.

[8] D.L. Donoho, "Detecting pairs of jittered interactive streams by exploiting maximum tolerable delay," Proceedings of 5th International Symposium on Recent Advances in Intrusion Detection, Zurich, Switzerland, 45-59, 2002.

[9] T. He, L. Tong, "Detecting encrypted stepping-stone connections," Proceedings of IEEE Transaction on signal processing, **55**(5), 1612-1623, 2007, doi: 10.1109/TSP.2006.890881.

[10] X. Wang, D.S. Reeves, S.F. Wu, J. Yuill, "Sleepy watermark tracing: an active network-based intrusion response framework," Proceedings of 16th International Conference on Information Security (IFIP/Sec'01), 369-384, 2001.

[11] X. Wang, D.S. Reeves, "Robust correlation of encrypted attack traffic through stepping-stones by manipulation of interpacket delays," Proceedings of ACM CCS '03, 2003.

[12] X. Wang, "The loop fallacy and serialization in tracing intrusion connections through stepping-stones," Proceedings of the 2004 ACM Symposium on Applied Computing, ACM Press, 2004.

[13] K.H. Yung, "Detecting long connecting chains of interactive terminal sessions," Proceedings of International Symposium on Recent Advance in Intrusion Detection (RAID), Zurich, Switzerland, 1-16, 2002.

[14] J. Yang, S.-H.S. Huang, "A real-time algorithm to detect long connection chains of interactive terminal sessions," Proceedings of 3rd ACM International Conference on Information Security (Infosecu'04), Shanghai, China, 198-203, 2004.

[15] J. Yang, S.-H.S. Huang, "Mining TCP/IP packets to detect stepping-stone intrusion," Journal of Computers and Security, Elsevier Ltd., **26**, 479-484, 2007, doi: 10.1016/j.cose.2007.07.001.

[16] J. Yang, Y. Zhang, "RTT-based random walk approach to detect stepping-stone intrusion," Proc. of 29th IEEE International Conference on Advanced Information Networking and Applications, Gwangju, South Korea, 558-563, 2015, doi: 10.1109/AINA.2015.236.

[17] J. Yang, "Resistance to chaff attack through TCP/IP packet cross-matching and RTT-based random walk," Proceedings of 30th IEEE International Conference on Advanced Information Networking and Applications, Crans-Montana, Switzerland, IEEE proceedings and Digital Library, 784-789, 2016, doi: 10.1109/AINA.2016.17.

[18] Z. Trabelsi, W. Ibrahim, "A hands-on approach for teaching denial of service attacks: a case study," Journal of information technology education: Innovations in Proactive, **12**, 299-319, 2013.

[19] J. Yang, L. Wang, B. Lockerbie, A. Lesh, "Manipulating network traffic to evade stepping-stone intrusion detection," Internet of Things, Elsevier, **3**(4), 34-45, 2018, doi: 10.1016/j.iot.2018.08.011.

[20] J. Yang, S.-H.S. Huang, M.D. Wan, "A clustering-partitioning algorithm to find TCP packet round-trip time for intrusion detection," Proceedings of 20th IEEE International Conference on Advanced Information Networking and Applications (AINA 2006), Vienna, Austria, **1**, 231-236, 2006, doi: 10.1109/AINA.2006.13.