

Deepening Intrusion Detection Understanding via a Simple IDS Assignment

Steven McCarthy
Computer Science
sjm4144@uncw.edu

Geoff Stoker
Congdon School
stokerg@uncw.edu

University of North Carolina Wilmington
Wilmington, NC 28403, USA

Abstract

Intrusion detection systems (IDS) have become a common feature of modern computers and networked systems. In this paper, we describe an IDS assignment designed to take the average student studying information systems, computer science, information technology, cybersecurity, or a similar major from a definitional understanding of what an IDS is and does to a deeper understanding of how it does it. This demystification assignment is of medium difficulty and could potentially be conducted as an in-class lab exercise. It is an outgrowth of a directed independent study with one student and will be used in future larger classes.

Keywords: IDS, Intrusion Detection, Cybersecurity, Education

1. INTRODUCTION

Over twenty years ago, and about twenty years after publication of what is generally considered the first work on automated computer intrusion detection (Anderson, 1980), the National Institute of Standards and Technology (NIST) published Special Publication (SP) 800-31, Intrusion Detection Systems (Bace & Mell, 2001). In that SP, intrusion detection systems (IDS) were defined as “software or hardware systems that automate the process of monitoring the events occurring in a computer system or network, analyzing them for signs of security problems” (p. 1). For additional clarity, intrusions were defined as “attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network” (Bace & Mell, 2001, p. 1).

Superseding SP 800-31 a few years later, NIST published SP 800-94 (Scarfone & Mell, 2007) and

though NIST subsequently released a draft SP 800-94 Revision 1 for public comment in July 2012, it was never published and was officially retired July 15, 2022 (NIST, 2022). Thus, we consider the 2007 SP 800-94 to be the current authoritative source, which defines intrusion detection in the following way:

Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible *incidents*, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices. (p. 2-1)

In the same NIST SP, an IDS is defined simply as “software that automates the intrusion detection process” (Scarfone & Mell, 2007, p. 2-1).

Searching Google Scholar for “Intrusion Detection System” returns over 130,000 results (2022). This provides some indication of the extent in depth and breadth to which the IDS topic has

been investigated. And, while the IDS has become an increasingly common feature of modern computers and networked systems, we have perceived a widening gap between an awareness of what an IDS does and an understanding of how it does it among the average student seeking a computer-based major. This perceived gap, given explicit voice by one senior computer science undergraduate student, led to a semester-long directed independent study undertaken to better understand how an IDS works. The discoveries made in that 1-on-1 course led us to the conclusion that a relatively simple exercise would likely go a long way towards helping other students understand how an IDS works. That realization led to the creation of the exercise described herein and the writing of this document.

In this paper, we present our idea for an assignment designed to take students studying a computer-related discipline from definitional knowledge of IDSs to a more detailed, hands-on understanding of how an IDS functions. In section 2 of this paper, we provide an overview and some background related to IDS technology; in section 3 we state the simple IDS assignment parameters; in section 4 we present the details of an example implementation; in section 5 we consider some additional aspects of the assignment; and section 6 concludes.

2. IDS OVERVIEW AND BACKGROUND

Conceptual Overview

At a conceptual level, the function of any IDS is conceptually straightforward (Figure 1). Data is collected, it is analyzed as appropriate depending on the detection method being used, and then a decision is made as to whether a suspected intrusion has taken place. If an intrusion is suspected, an alert is raised.

IDS Background

Anderson (1980) is generally credited with first publishing a paper discussing the IDS concept within a computing environment. He sketched out the elements needed for "the automatic generation of security exception reports" (p. 28) and suggested a metric for identifying unusual sessions that would cause an exception. The idea was to collect parameters for each session, e.g., number of input characters, number of file references, etc., and then create an alert when some measured parameter value was 2.58 standard deviations greater or less than the mean for other similar sessions.

After a few years of IDS research and the development of experimental systems exploring the IDS concept, Denning (1987) published a model intended to serve as "a framework for a general-purpose intrusion-detection expert system" (p. 1). This work recognized and was motivated by four primary factors:

- 1) existing systems have security flaws for which it is not technically/economically feasible to find/fix all of them;
- 2) flawed systems are not easily replaced by more secure systems for feature-related or economic reasons;
- 3) developing secure systems is extremely difficult;
- 4) even secure systems are vulnerable to insider threats.

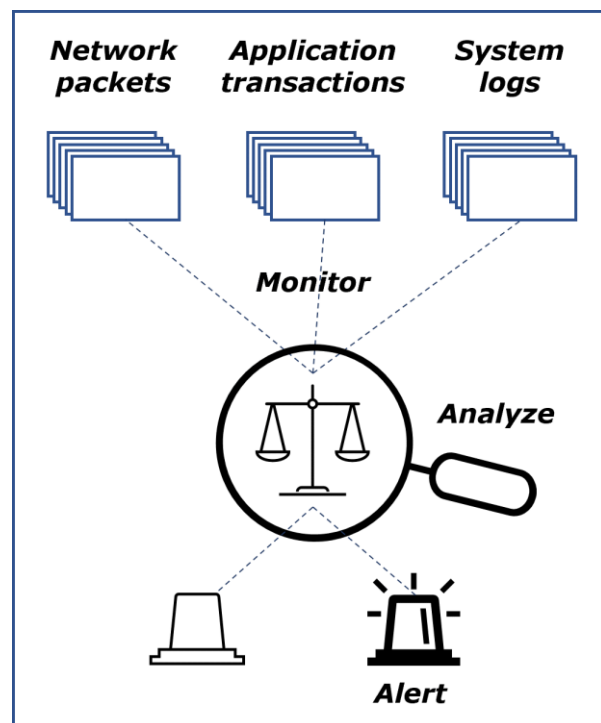


Figure 1 – conceptual generic IDS function

Over the last 40+ years, many researchers have incrementally extended the field while a few have conducted surveys in an attempt to capture snapshots of the IDS landscape at different moments in time (Ahmed et al., 2016; Axelsson, 2000; Deepa & Kavitha, 2012; Khraisat et al., 2019; Liao et al., 2013; McAuliffe et al., 1990; Sulaiman et al., 2021). We elaborate further on the results of some of these surveys in the next subsection.

IDS Classification

Researchers have chosen several different ways to group IDS technologies. Generally, the

groupings are based on either where detection takes place or how it takes place (Sulaiman et al., 2021). Based on where detection takes place, IDS technologies generally fall into one of two basic types: host-based IDS (HIDS) or network-based IDS (NIDS) (Axelsson, 2000; Bace & Mell, 2001; Debar et al., 1999; Sulaiman et al., 2021).

NIST SP 800-94 adds two additional categories based on how the technology is deployed and what it monitors: wireless-based IDS (WIDS), and network behavior analysis (NBA) (Scarfone & Mell, 2007). Other types that have been considered important distinctions and have been enumerated over the years include application-based IDS (Bace & Mell, 2001), mixed IDS (MIDS, Liao et al., 2013), stack-based IDS, protocol-based IDS, and graph-based IDS (Deepa & Kavitha, 2012).

Descriptions of the four categories enumerated by NIST SP 800-94 are as follows.

- A HIDS is designed to run on a single host machine and analyze computer events that occur only on that host.
- A NIDS is deployed to monitor network traffic that is moving between and among host machines. They are often positioned at network boundary locations.
- A WIDS is designed to monitor part of the radio frequency (RF) spectrum in order to identify, capture, and analyze wireless transmissions.
- An NBA IDS is focused on looking for threats that generate unusual network traffic flows, such as distributed denial-of-service (DDoS) attacks or port scanning.

The second primary way to categorize an IDS is based on detection method. The three primary types of detection are: signature-based detection (SD), anomaly-based detection (AD), and stateful protocol analysis detection (SPA) (Ahmed et al., 2016; Bace & Mell, 2001; Khraisat et al., 2019; Liao et al., 2013; McAuliffe et al., 1990; Scarfone & Mell, 2007).

- With SD, an IDS compares patterns of known attacks or threats against monitored network traffic or computer events. This technique is sometimes alternatively called misuse detection (Bace & Mell, 2001) or knowledge-based detection (Liao et al., 2013).
- To use AD, profiles must be created that represent normal activity against which observed network traffic and computer events can be compared. Anything considered abnormal will cause an alert.

Sometimes, this is called behavior-based detection (Liao et al., 2013).

- Somewhat similar to AD, SPA uses existing profiles of benign protocol activity against which to compare observed events. The key differences are that the baseline consists of universal profiles typically provided by vendors, and they involve protocols that have a notion of state that can be tracked. The SPA approach is also known as specification-based detection (Liao et al., 2013).

There are potentially many other IDS features that could be used for IDS taxonomical classification (e.g., see Appendix A), but in this paper, we will only reference the two general features outlined above – where detection takes place and the detection method used.

3. ASSIGNMENT OVERVIEW

The overall objective of the IDS assignment is to have students with a definitional knowledge of IDS concepts move beyond their theoretical understanding and gain some hands-on awareness by constructing a simple IDS (Figure 2). To keep scope manageable, the simple IDS will be host-based and signature-based; it will only look for a single type of network packet – Internet Control Message Protocol (ICMP) echo request packets or, more popularly, ping packets.

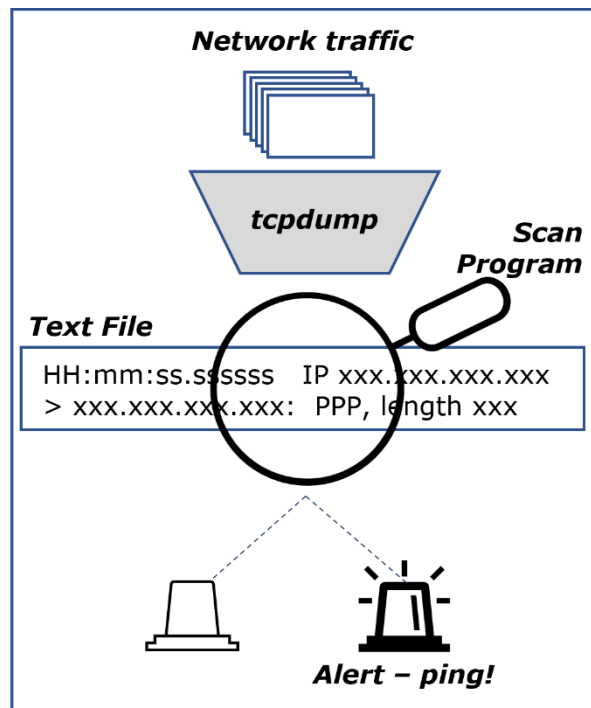


Figure 2 – simple IDS concept diagram

While we provide implementation details in the next section, we describe the assignment in broad strokes here to provide a useful context for later details. This system is installed on a host and makes use of an existing packet sniffer, *tcpdump* (<https://www.tcpdump.org/>), that captures all network traffic and writes it to a text file. A search program examines the text file for "ICMP echo request" and, if found, prints an alert out to the screen.

4. IMPLEMENTATION

Environment

The simple IDS assignment requires the existence of a host machine with a network interface and the presence of some other communicating device. Several options are available for creating the necessary environment. For example, two different physical machines could be connected to the same switch or directly connected to one another, or two virtual machines (VM) could be created on a physical machine and the two VMs could be virtually connected to each other. We used a single physical machine, a desktop, as the communicating device and a virtual machine running on that physical machine as the host on which we created the IDS. The stylized view of the environmental setup is shown in figure 3 along with some details like the operating system (OS), assigned IP address, and an indication of where the different parts of the assignment ran.

The physical machine had Windows 10 installed and ran Oracle VM VirtualBox within which an Ubuntu 20.04.3 LTS virtual machine (VM) was created. The particular VM configuration details are not especially important, but we note that we set it up to have 4GB of RAM, 10GB of hard drive storage space, and configured the network setting so that the network adapter was attached to "bridged adapter" for networking.

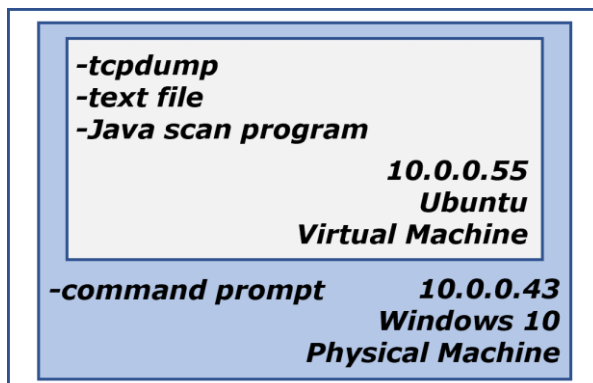


Figure 3 – stylized diagram of assignment environment

Physical machine

The physical machine only needed to ping the VM to ensure the traffic we wanted to find would be present. So, we opened a command prompt window and ran ping with the -t option (figure 4) so that the VM would be pinged until explicitly stopped.

```
Command Prompt - ping 10.0.0.55 -t
C:\Users\Home>ping 10.0.0.55 -t

Pinging 10.0.0.55 with 32 bytes of data:
Reply from 10.0.0.55: bytes=32 time<1ms TTL=64
Reply from 10.0.0.55: bytes=32 time<1ms TTL=64
Reply from 10.0.0.55: bytes=32 time<1ms TTL=64
Reply from 10.0.0.55: bytes=32 time<1ms TTL=64
```

Figure 4 – physical machine pinging the VM

Virtual machine

The actual simple IDS ran entirely on the Ubuntu VM. It consisted of three components: *tcpdump*, an associated *tcpdump* output text file, and a program, written in Java, that searched for the presence of ICMP echo request packets within the *tcpdump* output text file.

Despite its name, *tcpdump* is not restricted to looking at Transmission Control Protocol (TCP) packets. It is equally adept at examining User Datagram Protocol (UDP) and ICMP traffic, the latter of which is actually integrated into the layer 3 Internet Protocol (IP) suite. Often, *tcpdump* is preinstalled on Linux; however, some distributions do not include it, and this was the case with our Ubuntu VM. So, we first had to download *tcpdump* and install it. Once installed, we ran *tcpdump* from the command line with several options to perform as we particularly desired:

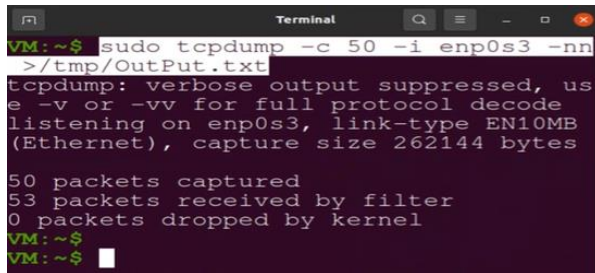
- c allows the specification of how many packets to capture before exiting *tcpdump*
- i specifies the network interface from which to capture packets
- nn disables domain name and port resolution so that only IP addresses are output

For our implementation, we ran *tcpdump* prefaced by the *sudo* command since it requires elevated privileges to run. We specified with -c that it should capture 50 packets and then stop. We redirected the *tcpdump* output into a text file using the '>' symbol (see Figure 5).

The text file that captured the output from *tcpdump* was generally in the format:

```
time IP source > IP destination protocol
```

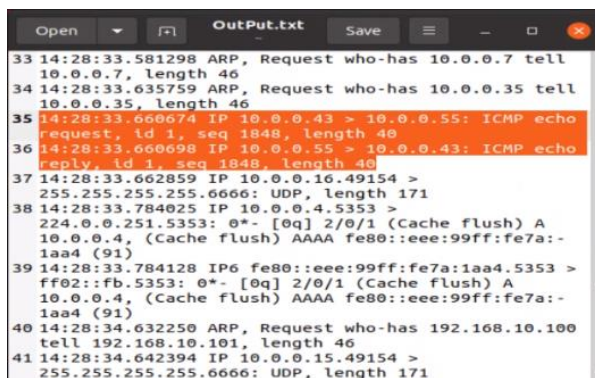
The specific output generated by tcpdump during our implementation of this assignment is shown in figure 6. The ICMP echo request and reply packets are highlighted in orange.



```
VM:~$ sudo tcpdump -c 50 -i enp0s3 -nn  
>/tmp/OutPut.txt  
tcpdump: verbose output suppressed, use  
-v or -vv for full protocol decode  
listening on enp0s3, link-type EN10MB  
(Ethernet), capture size 262144 bytes  
50 packets captured  
53 packets received by filter  
0 packets dropped by kernel  
VM:~$  
VM:~$
```

Figure 5 – running tcpdump with options

Once the text file was created, we ran a program, written in Java, to parse the file, find instances of “ICMP echo request,” and print alerts to the screen. Though the details of this simple program are not necessarily important, we provide in Appendix B the Java code that we used.



```
33 14:28:33.581298 ARP, Request who-has 10.0.0.7 tell  
10.0.0.7, length 46  
34 14:28:33.635759 ARP, Request who-has 10.0.0.35 tell  
10.0.0.35, length 46  
35 14:28:33.660674 IP 10.0.0.43 > 10.0.0.55: ICMP echo  
request, id 1, seq 1848, length 40  
36 14:28:33.660698 IP 10.0.0.55 > 10.0.0.43: ICMP echo  
reply, id 1, seq 1848, length 40  
37 14:28:33.662859 IP 10.0.0.16.49154 >  
255.255.255.255.6666: UDP, length 171  
38 14:28:33.784025 IP 10.0.0.4.5353 >  
224.0.0.251.5353: 0* [0q] 2/0/1 (Cache flush) A  
10.0.0.4, (Cache flush) AAAA fe80::eee:99ff:fe7a:-  
1aa4 (91)  
39 14:28:33.784128 IP6 fe80::eee:99ff:fe7a:1aa4.5353 >  
ff02::fb.5353: 0* [0q] 2/0/1 (Cache flush) A  
10.0.0.4, (Cache flush) AAAA fe80::eee:99ff:fe7a:-  
1aa4 (91)  
40 14:28:34.632250 ARP, Request who-has 192.168.10.100  
tell 192.168.10.101, length 46  
41 14:28:34.642394 IP 10.0.0.15.49154 >  
255.255.255.255.6666: UDP, length 171
```

Figure 6 – text file of tcpdump output

5. DISCUSSION

The idea and implementation details for the simple IDS assignment were an outgrowth of a directed independent study, a 1-on-1 teacher-student course designed to emphasize the value of applied learning (Galizio, n.d.), focusing on IDSs. In the process of working out the assignment details, we encountered some issues and had some ideas for additional aspects that we believe merit mention.

Issues

The parts of the assignment directly related to the simple IDS are relatively straightforward. Pinging the host, installing/running tcpdump on the host, and redirecting tcpdump output into a text file should not be a big challenge for students in a computing-related major. As well, writing a program in a preferred language to search through a text file for a specific string should be

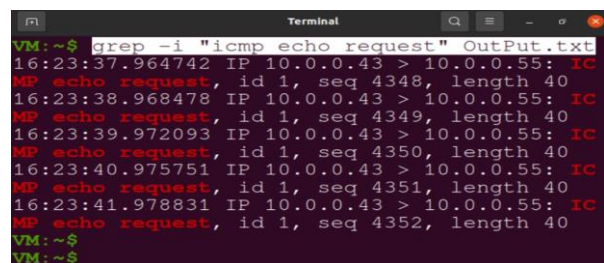
within reach of anyone with introductory programming experience.

Setting up the environment in which to run this assignment could be challenging. The particular VM environment we describe required some troubleshooting and configuration adjustments that many students might find difficult, especially if they have no previous hands-on experience with VMs. For example, the desktop we used has an AMD chipset that presented an error after VirtualBox was installed and the Ubuntu VM invoked. The problem was that AMD virtualization (AMD-V) needed to be enabled in the basic input/output system (BIOS) for the VM to function correctly.

Choosing instead to use an environment with two physical machines would certainly be visually simplifying, but it may also create additional challenges related to networking. For example, the Windows Defender Firewall blocks ping requests by default so, if a Windows machine were used as the host, it would need to have that firewall rule disabled. If the machine used is not specifically a lab machine, the user would want to remember to reenact that rule after the assignment to ensure the intended protection is provided.

Additional Aspects

There are many potential ways to change or extend this simple IDS assignment that can add value in a range of ways from simple to complex. One obvious way is in how the text file is scanned for the presence of the string “ICMP echo request.” While the person coding this assignment was most comfortable and proficient with Java, we also wrote a scanner in Python and used grep from the command line. Using grep (figure 7), a text pattern matching program built into Linux, as the scanner is one way in which this assignment could more easily be done in a lab environment.



```
VM:~$ grep -i "icmp echo request" OutPut.txt  
16:23:37.964742 IP 10.0.0.43 > 10.0.0.55: IC  
MP echo request, id 1, seq 4348, length 40  
16:23:38.968478 IP 10.0.0.43 > 10.0.0.55: IC  
MP echo request, id 1, seq 4349, length 40  
16:23:39.972093 IP 10.0.0.43 > 10.0.0.55: IC  
MP echo request, id 1, seq 4350, length 40  
16:23:40.975751 IP 10.0.0.43 > 10.0.0.55: IC  
MP echo request, id 1, seq 4351, length 40  
16:23:41.978831 IP 10.0.0.43 > 10.0.0.55: IC  
MP echo request, id 1, seq 4352, length 40  
VM:~$  
VM:~$
```

Figure 7 – using grep to search for “icmp echo request”

While tcpdump is a free, highly available tool for packet sniffing/capture, it is not the only one.

This assignment could be altered to include any of a myriad other sniffer/capture tools or explore the differences among them. Since production IDSs work continuously to inspect network traffic, an obvious, though non-trivial, extension of this assignment would be to figure out how to mimic this behavior.

One nice ancillary benefit to working on a hands-on activity like this simple IDS assignment is that it is likely to stimulate curiosity about things not directly related to the specific IDS assignment tasks. As a small example, when getting ready to take the screen capture for figure 5, we decided that the default command prompt was unnecessarily long. Since our distribution of Linux was using the Bourne Again Shell (Bash – <https://www.gnu.org/software/bash/>), figuring out how to change it by editing the .bashrc file was a new experience (Jevtic, 2020). After shortening it and discovering that it defaulted to the same text color as the commands being typed in, a second curiosity was stimulated that led to figuring out how to change the color of the prompt to ensure some contrast was present in the image (Gite, 2020).

6. CONCLUSIONS

In this paper, we set out to provide a relatively simple way to cultivate a deeper understanding of an IDS in students who are following the curriculum of a computer-based major. Providing applied learning assignments that demystify concepts that are only typically understood at a conceptual level can be an effective means to do this. It was an extremely useful and enlightening exercise for the one student engaged in the directed independent study and we have plans to use it in future larger classes. While we cannot know that it will definitely be as useful to all students, we are optimistic that most will find it illuminating.

We provided a brief overview of IDS evolution and described common features used when classifying IDSs. We also described the base features of an assignment that students could use to build their own simple IDS, provided an example implementation of that IDS, and then suggested some extensions that could potentially springboard students into further exploration of this topic.

7. REFERENCES

Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer*

Applications, 60, 19-31. <https://www.sciencedirect.com/science/article/abs/pii/S1084804515002891?via%3Dihub>

Anderson, J. P. (1980, April 15). *Computer Security Threat Monitoring and Surveillance*. Technical Report. <https://csrc.nist.gov/csrc/media/publications/conference-paper/1998/10/08/proceedings-of-the-21st-nissc-1998/documents/early-cs-papers/ande80.pdf>

Axelsson, S. (2000, March 14). *Intrusion Detection Systems: A Survey and Taxonomy*. Chalmers University of Technology, Sweden, Technical Report 99-15. https://www.researchgate.net/publication/2597023_Intrusion_Detection_Systems_A_Survey_and_Taxonomy

Bace, R. & Mell, P. (2001, November). NIST SP 800-31, *Intrusion Detection Systems*. <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-31.pdf>

Debar, H., Dacier, M., & Wespi, A. (1999, April 23). Towards a taxonomy of intrusion-detection systems. *Computer networks*, 31(8), 805-822. <https://www.sciencedirect.com/science/article/pii/S1389128698000176?via%3Dihub>

Denning, D. (1987, February). An intrusion-detection model. *IEEE Transactions on software engineering*, (2), 222-232. <https://apps.dtic.mil/sti/pdfs/ADA484998.pdf>

Deepa, A. J., & Kavitha, V. (2012). A comprehensive survey on approaches to intrusion detection system. *Procedia Engineering*, 38, 2063-2069. <https://www.sciencedirect.com/science/article/pii/S187705812021613>

Galizio, M. (n.d.). Directed independent study. Retrieved July 30, 2022, from <https://uncw.edu/cte/resources/dis.html>

Gite, V. (2020, October 29). BASH shell change the color of shell prompt on linux or unix. <https://www.cyberciti.biz/faq/bash-shell-change-the-color-of-my-shell-prompt-under-linux-or-unix/>

Google Scholar. (2022). Retrieved July 21, 2022, from <https://scholar.google.com/scholar?q=%22intrusion+detection+system%22>

Jevtic, G. (2020, May 12). How to customize bash prompt in linux. <https://phoenixnap.com/kb/change-bash-prompt-linux>

Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and

- challenges. *Cybersecurity*, 2(1), 1-22. <https://link.springer.com/article/10.1186/s42400-019-0038-7>
- Liao, H. J., Lin, C. H. R., Lin, Y. C., & Tung, K. Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1), 16-24. <https://www.sciencedirect.com/science/article/pii/S1084804512001944>
- McAuliffe, N., Wolcott, D., Schaefer, L., Kelem, N., Hubbard, B., & Haley, T. (1990, December). Is your computer being misused? A survey of current intrusion detection system technology. In *IEEE Proceedings of the Sixth Annual Computer Security Applications Conference* (pp. 260-272).
- National Institute of Standards and Technology. (2022, July 15). SP 800-94, Guide to Intrusion Detection and Prevention Systems (IDPS). <https://csrc.nist.gov/publications/detail/sp/800-94/final>
- Scarfone, K. & Mell, P. (2007, February). NIST SP 800-94. Guide to Intrusion Detection and Prevention Systems (IDPS). <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-94.pdf>
- Sulaiman, N. S., Nasir, A., Othman, W. R. W., Wahab, S. F. A., Aziz, N. S., Yacob, A., & Samsudin, N. (2021, May). Intrusion Detection System Techniques: A Review. In *Journal of Physics: Conference Series* (Vol. 1874, No. 1, p. 012042). IOP Publishing. <https://iopscience.iop.org/article/10.1088/1742-6596/1874/1/012042/pdf>

APPENDIX A

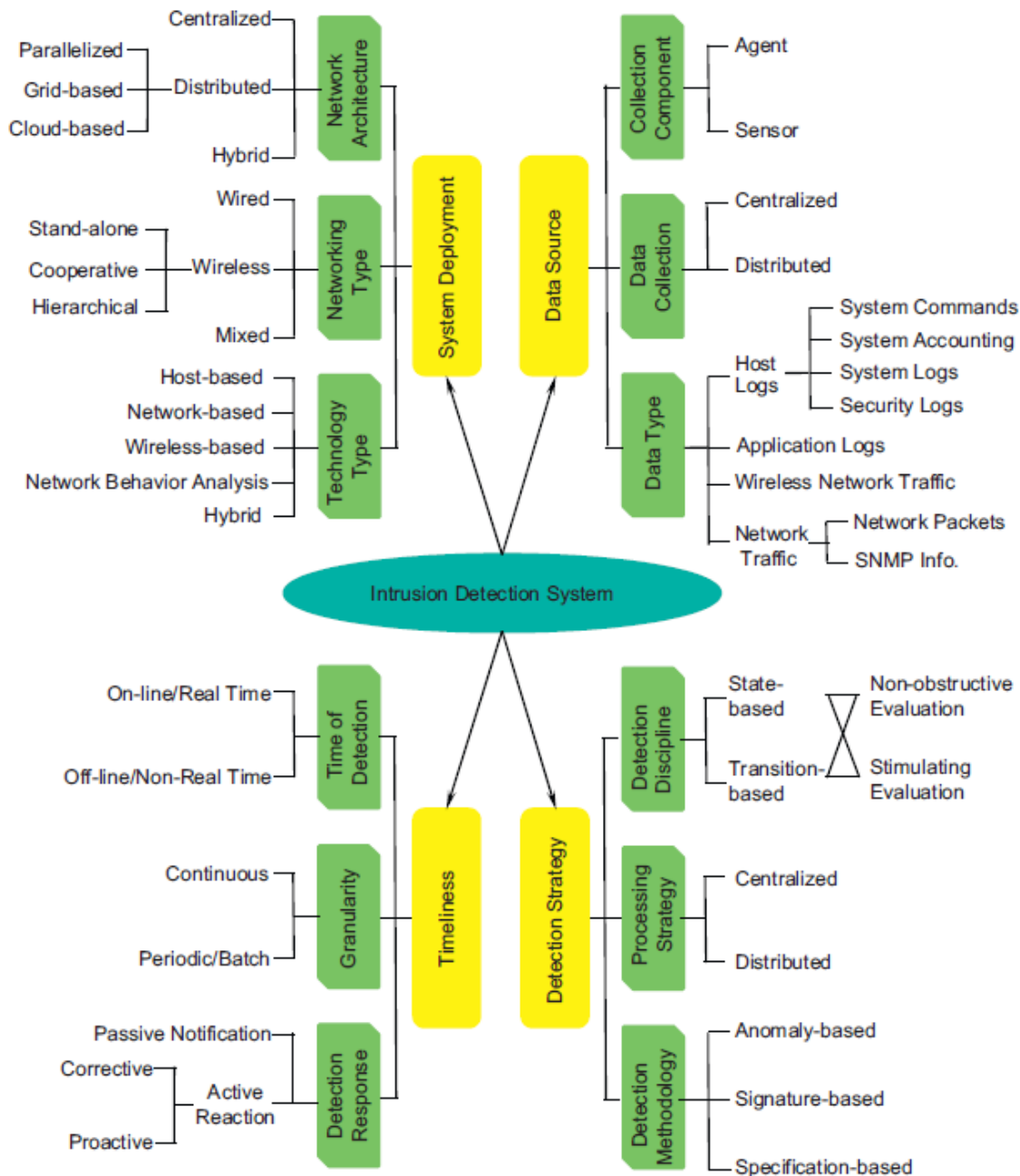


Figure A-1 – example taxonomy for IDS (Liao et al., 2013, Figure 1)

APPENDIX B

Java code for searching output file for "ICMP echo request"

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class SimpleIDS
{
    public static void main(String[] args) throws FileNotFoundException
    {
        File file = new File("/tmp/OutPut.txt");
        Scanner scan = new Scanner(file);
        String findings = new String("ICMP echo request");
        String line = scan.nextLine();
        while(scan.hasNextLine()) { //loop through the file lines
            //does the word match the findings?
            if(line.contains(findings)) {
                line = scan.nextLine();
                System.out.println("Alert Triggered");
                return; } //end if
            else {
                //get the next word
                line = scan.nextLine(); } //end else
        }
        System.out.println("No Alerts Found");
    }
}
```