Network Traffic Proxy System

Version 1.5

10/14/18

Document Control

Approval

The Guidance Team and the Customer will approve this document.

Document Change Control

|  |  |
| --- | --- |
| Initial Release: | Version 1.0 |
| Current Release: | Version 1.6 |
| Indicator of Last Page in Document: | 13 |
| Date of Last Review: | 10/9/2018 |
| Date of Next Review: | 10/14/2018 |
| Target Date for Next Update: | 10/14/2018 |

Distribution List

This following list of people will receive a copy of this document every time a new version of this document becomes available:

Guidance Team Members:

Elsa Tai Ramirez

Jose Cabrera Maynez

Customer: Dr. Jaime Acosta

Mr. Christian Murga

Mr. Caesar Zapata

Software Team Members: Alan Caldelas

Oscar Galindo

Kevin Gonzalez

Isai Gonzalez

Julio De La Cruz

Change Summary

The following table details changes made between versions of this document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modifier | Description |
| 1.1 | 10/02/2018 | Alan Caldelas  Isai Gonzalez | Added the Purpose, Justification, and Use case sections. |
| 1.2 | 10/06/2018 | Oscar Galindo | Added the description of the considerations made and added the proposed solutions. |
| 1.3 | 10/8/2018 | Isai Gonzalez  Julio De La Cruz  Kevin Gonzalez  Alan Caldelas  Oscar Galindo | Added sections 2, 3, 4, 5, and 7. |
| 1.4 | 10/11/2018 | Isai Gonzalez | Updated section 1 with corrections and new use case diagram. Fixed reference list format. |
| 1.5 | 10/13/2018 | Julio De La Cruz | Updated section 2 and section 3.1 |
| 1.6 | 10/13/2018 | Oscar Galindo | Updated section 1.3 and 1.4, organized section 3, and made description modifications to section 4. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Document Control ii](#_Toc527276101)

[Approval ii](#_Toc527276102)

[Document Change Control ii](#_Toc527276103)

[Distribution List ii](#_Toc527276104)

[Change Summary ii](#_Toc527276105)

[1. Introduction 5](#_Toc527276106)

[1.1. Purpose of the Feasibility Report 5](#_Toc527276107)

[1.2. Justification for the Proposed System 5](#_Toc527276108)

[1.3. Requirements Definition 5](#_Toc527276109)

[1.4. Use Cases 6](#_Toc527276110)

[2. Existing Solutions 7](#_Toc527276111)

[2.1. Complete System Solutions 8](#_Toc527276112)

[3. Considerations 8](#_Toc527276113)

[3.1. Consideration 1 - Operating System 9](#_Toc527276114)

[3.2. Consideration 2 - Programming Languages 9](#_Toc527276115)

[3.3. Consideration 3 - Network Sniffer 10](#_Toc527276116)

[3.4. Consideration 4 - Packet Replayer 10](#_Toc527276117)

[3.5. Consideration 5 - Enqueueing Tool 11](#_Toc527276118)

[3.6. Consideration 6 - Packet Replayer 11](#_Toc527276119)

[3.7. Consideration 7 - Fuzzer 11](#_Toc527276120)

[4. Solutions 11](#_Toc527276121)

[4.1. Solution 1 11](#_Toc527276122)

[4.2. Solution 2 12](#_Toc527276123)

[5. Comparison of Solutions 13](#_Toc527276124)

[6. Conclusions 14](#_Toc527276125)

[7. References 15](#_Toc527276126)

# 1. Introduction

## 1.1. Purpose of the Feasibility Report

The purpose of the feasibility report is to document the outcomes of the feasibility study. The feasibility study will evaluate the overall feasibility of the project. The feasibility report is made up of different topics of feasibility. These topics include the potential cost of the system, the hardware requirements for the system, the amount of time that building the system will take, how much training our team will require, and what outside software will be integrated into our system. Using these topics, Team 5 will consider different possible solutions to build the system. By reporting on our findings, we will give a better idea to the client, Dr. Acosta, Mr. Murga, and Mr. Zapata, as to what approximation of the system they require is possible to build and with what limitations. The client will then be able to choose which feasible system will work better for them.

## 1.2. Justification for the Proposed System

Security in the world of communications is a crucial aspect that is always needed to be tested, especially in network systems. With a security perspective in mind, network analysts need to be able to test the way systems communicate, which vary from protocols and backend software. Analysts need software that will aid them to facilitate the testing of security in any particular network. The Network Traffic Proxy System will do just that. It will help “test and evaluate network systems [1]” and simplify this task by having plenty of features under one interface.

## 1.3. Requirements Definition

As requested by Dr. Acosta, Mr. Murga, and Mr. Zapata, the Network Traffic Proxy System (NTPS) will allow an analyst to perform various actions. First, the NTPS should allow an analyst to intercept packets through the use of a packet filtering software that can set specific rules of packet capture at the kernel level. Interception should also be able to be activated or de-activated. Packets captured should be able to be modified as these are registered into the system. And these modifications should happen in different display modes including raw hexadecimal, binary, and decoded modes. In addition, the “packet data should be separated by layer [1].”

Additionally, the analyst should have access to a randomizer tool, or Fuzzer, that allows an analyst to make a selective modification to the layers or the sections of the intercepted packet. As specified by the requirements given by the client, our system should support the American Fuzzy Lop (AFL), which is an already built fuzzing software. [1] Also, the NTPS should support the use of PCAP files. This means that intercepted traffic will be saved into a PCAP file. The analyst will be able to load PCAP file(s) and display their packets. As stated before, these packets should be modifiable by the user. And interception does not have to be activated for the user to view and modify packets. [1]

Graphically, the NTPS should support a display of all the system’s functionalities, including the packets being “sniffed” or being seen by the network sniffer, as graphic elements or options to execute in a graphical user interface (GUI). These options should include the ability to introduce hooks (scripts) into the system. Analysts should be able to load hooks, activate and stop the execution of hooks and track the status of every hook, as well as being able to produce combinations of loaded hooks or a collection altogether. Within every collection, analysts should be able to turn on and off any hook, as well view their status. [1]

Additional features of the system include an enqueueing tool that can hold at least 100 packets, that the NTPS, according to Acosta, Murga, and Zapata, “should be usable by an intermediate to expert level analyst and that [1].”

As requested by Dr. Acosta, Mr. Murga, and Mr. Zapata, the Network Traffic Proxy System will allow the user to set packet filtering rules and save them to filter out what packets will be intercepted by the system. The interception system could be turned off if the user requests it. The user will be able to modify any of the fields of a packet, as well as forward and/or drop the packets. There will be different display modes for each packet including raw hexadecimal, binary, and decoded modes. In addition, the “packet data should be separated by layer [1].”

The system should also support PCAP files. This means that intercepted traffic will be saved into a PCAP file. The user will be able to load PCAP file(s) and display their packets. As stated before, these packets should be modifiable by the user. Live interception does not have to be on for the user to view and modify packets. [1]

Analysts will also be able to load hooks into our system. They will be able to turn them off/on and view their status. They will also have the option to combine loaded hooks into a collection or load a collection altogether. From the collection, analysts will still be able to turn them on and off as well view their status. [1]

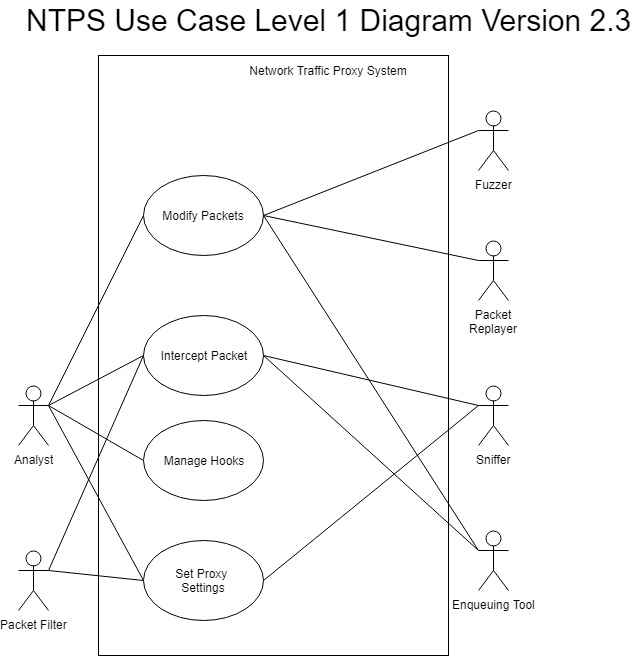
Our system should be able to use a fuzzer to fuzz packets. The analyst will be able to choose what to fuzz, generate fuzzed packets, and be able to send or not send packets. As specified by the requirements given by the client, our system should support the American Fuzzy Lop (AFL), which is an already built fuzzing software. [1]

Additional features of the system include a queue that can hold at least 100 packets, a graphical interface that according to Acosta, Murga, and Zapata, “should be usable by an intermediate to expert level analyst [1].”

## 1.4. Use Cases

A use case diagram is very commonly used to show the interactions that people or things will have with a system. This can be used to explain the overall purpose of the system and what it can be used for. A use case diagram includes actors, use cases, and the relationships between them. An “actor” is an entity that is outside of the system that will have some sort of interaction with the system. A “use case” is a function of a system that interacts with at least one actor. A use case should not be a step to accomplish a functionality. A line that connects an actor to a use case is used to show that there will be some sort of interaction between the two. An interaction does not necessarily mean that the actor will directly use that functionality, but the actor must be involved in some way with the linked use case. [15]

There are two levels of a use case diagram. A level 1 diagram will only show the main use cases of the system, while a level 2 will be a more specific version of the use case diagram. A level 2 diagram can show smaller tasks that an actor can do, as well as steps that are required by more than one use case. The level 2 use case diagram for the Network Traffic Proxy System will not be included, but the following is the level 1 diagram that Team 5 came up with, along with an explanation:



Description of use cases:

* Modify Packets: The system will allow for the modification of packets. This includes being able to change specific fields of a packet and generate completely new packets.
* Intercept Packets: The system will be able to intercept packets off of a network and save them into a queue.
* Manage Hooks: Hooks will be able to be loaded into the system, be turned on/off, and be grouped together into a collection and the status of every hook can be tracked in the system.
* Set Proxy Settings: The system will allow for the proxy settings to be adjusted in terms of what packets to forward or dropped when acting as a proxy.

Description of actors:

* Analyst: The main user of the system
* Fuzzer: Software used to create random data for a packet, it possible in some cases to generate its own packets.
* Packet Replayer: Software that can send out reconstructed packets into a network.
* Enqueuing Tool: Software allows to perform modifications to the packet queue built at the kernel level of the operating system.
* Sniffer: Software that can see the packets passing through a network and intercept them.
* Packet Filter: Software that allows us to set network parameters of capture, either by user, port, protocol, and some other.

# 2. Existing Solutions

This section provides the information on other systems that that satisfies some or most of the solution to our problem and why they are important to learn from. Also, this section will aim to compare the existing solutions to the customer’s need mentioned in the RDD. Each existing solution will be evaluated based on what need they satisfy, and the cost required to use their solution.

## 2.1. Complete System Solutions

* **BurpSuite:** A graphical tool for testing Web application security, operating on HTTP Proxy and is used for man-in-the middle between browser and web servers. Allows for interception of traffic in both directions. This application accomplishes a variety of task that the customer wants, it’s able to intercept packets, modify packets, decode packets that is capable of taking raw data and display it into hex or decimal. The draw backs that this product has is it only supports HTTP packets and our customer wants to be able to intercept IP, Ether, TCP, and ICMP. The cost must also be considered and BurpSuite has three levels of cost. There is a free version of BurpSuite but this doesn’t come with all of the tools available and is very limited because it only provides their essential manual tools. Professional version that costs $399 per year and only includes their web scanner and advanced manual tools. The final option is the Enterprise which costs $3,999 per year and includes all of their functionalities they have to offer.

* **WireShark:** Is a tool that is used to analyze network protocols, this tool allows you to see what is going on in your network. This tool allows for live capture that stores packets in a PCAP file, but only allows offline analysis. It can be integrated with other tools such as TCPDump, and already includes TShark, TShark is mainly used for it command line interface instead of WireSharks GUI options. This also satisfies the customer’s need of being able to intercept Ether, IP, TCP and ICMP. This system doesn’t allow for queueing or live modification of packets. This would be a good software to integrate into our system as it is open source, which means it is a free software. It is capable of completing a lot of the customers’ needs for intercepting packets and storing them, but it doesn’t satisfy all needs.

* **NFQUEUE with Scapy:** Scapy is an open source packet manipulation program that can read and decode packets, modify values, and create packets. This software also supports all the protocols IP, TCP, TCMP, Ether and other not listed by the customer such as UDP and DNS. Scapy also allow for “on the fly” packet manipulation and creation.[5] The user is able to create new packets exactly how they want them to instead of modifying an existing packet. NFQUEUE is a Linux tool that can be used with Scapy to intercept packets and store them into a queue. Since NFQUEUE is a queue and can hold up to 100 packets in the queue it satisfies one of the needs of the customer. However, the downside to all of this is that Scapy is known to be a very buggy system and is constantly being updated because it is open source.[5]

# 3. Considerations

In this section we introduce the conglomerate of elements that are considered as fundamental for the well development of this project. We list elements in form of initial consideration and explain why the elements are needed.

## 3.1. Consideration 1 - Operating System

In the RDD and interview the client specifically asked for the Kali Linux Operating System however we will introduce a secondary Operating System that the client may want to consider alongside Kali Linux.

* **Kali Linux Version 2018.2:** Kali Linux is specifically geared to meet the requirements of professional penetration testing and security auditing. This Operating System also contains systemic hooks that disable network services by default and also uses an upstream kernel, patched for wireless injection. Kali also comes pre-installed with Wireshark, BurpSuite. However, Kali Linux may be the most widely known Operating System for penetration testing it does have a few drawbacks. Though the system is meant to be highly customizable we can’t expect to be able to add random unrelated packages and repositories and have all implementations work coherently and without any issues. [20] This means it might take some extra effort and tinkering to allow for other application to be used in this Operating System. [20]

* **BackBox Ubuntu:** BackBox is a version of the Ubuntu flavor of Linux and is geared for penetration testing and security assessment that provides network and system analysis toolkits. This system includes a wide variety of tools that range from web application analysis, network analysis, stress tests, sniffing, and vulnerability assessments. [18] This system also includes Scapy, WireShark, and Ettercap, which are all tools that our client wants. BackBox is highly customizable and is constantly updated to the latest stable version of the most known and used ethical hacking tools. Some disadvantages to BackBox is the learning curve for this system is relatively high and it doesn’t have a bug bounty program like Kail Linux does. Also, this operating system is open source, but it does lack some community support in that aspect.

## 3.2. Consideration 2 - Programming Languages

* C
* C++
* Python

## 3.3. Consideration 3 - Network Sniffer

* **Scapy** – A packet manipulation program that can read and decode packets, modify values, and create packets. The system analyst will be able to use Scapy in an extensive number of protocols in order to send protocols through, save them, compare them and much more other functionalities.
  + *Advantages of Scapy:* Supports multiple network protocols, it covers the range of tools needed from capturing to executing functions on a packet,
  + *Disadvantages of Scapy:* Cannot handle a large number of packets simultaneously and offers only partial support for certain complex protocols.
* **T-Shark** – A network protocol analyzer. Let's the system analyst capture packet data from a live network, or read packets from a previously saved capture file, either printing a decoded form of those packets to the standard output or writing the packets to a file. This is a key option for our system.
  + *Advantages of T-Shark:* Can be used inside a script, or in the case for the project, could be used within hooks.
  + *Disadvantages of T-Shark:* Does not display a graphical user interface for easy access and control, especially when searching for a great amount of network data.
* **TCPDump** – Is a packet analyzer that runs on the command line, it is used for displaying TCP/IP and other packets that are transmitted or received over a network.
  + *Advantages of TCPDump:* Flexible interfacing with other sniffers through the creation of PCAP files, readily available in most Linux environments and installable in some others.
  + *Disadvantages of TCPDump:* The lack of analysis, less filtering options available than other sniffers, and the limitations on the type of network traffic that can be analyzed. No easy implementation, it manual requiring the user to know most of the options for the packets. Packets blocked by a gateway firewall may not be seen.

* **Wireshark** – A network packet analyzer, this will attempt to capture network packets and displays the packet data in a detailed fashion.
  + *Advantages of Wireshark:* is the ability to open packet data with several other programs. Wireshark can import from many different capture files and export files for other programs. Another added benefit is that it is Open Source Software and it contains.
  + *Disadvantage of Wireshark:* is notifications will not make it evident if there is an intrusion in the network and Wireshark can only gather information form the network.

## 3.4. Consideration 4 - Packet Replayer

The Network Traffic Proxy System not only needs to be able to intercept packets, but also be able to resend those packets if the user wants to. This could also be called “replaying” packets. Depending on what sniffer the client wants, this might or might not be supported. Because of this, here are some software that can work as packet replayers.

* **Scapy** –
* **Tcpreplay** –
  + *Advantages of TCPreplay:*
  + *Disadvantages of TCPreplay:*
* **PyPacker** – Used for packet creation and parsing for Python. Allows the creation of packets manually by defining all aspects of all header data. Pypacker also allows for the dissection of packets.
  + *Advantages of PyPacker:* Creating packet on different layers, and the concatenation of layers. Live packet reading and writing using a capsulated socket API
  + *Disadvantages of PyPacker:* It is not as developed as Scapy, so no automatic port-scanner or using as a fuzzer. It requires implementation with tools like gnuplot.

## 3.5. Consideration 5 - Enqueueing Tool

* **NFQUEUE** –
  + *Advantages of NFQUEUE:*  Allows to make verdict in a more flexible way, instead of ridged set of rules you can use software that allows more flexibility
  + *Disadvantage of NFQUEUE:* The drawback is that if the queue is full the Kernel will drop incoming packets.

## 3.6. Consideration 6 - Packet Filter

* **ip-tables** –
  + *Advantages for ip-tables:*
  + *Disadvantages of ip-tables:* Any change made to the current table of rules is slow, since every operation of adding and removing has to process the entire table of rules. Adding rules is not an atomic operation, which means that if two rules are added at the same time through ip-tables system only one will be in effect after execution.
* **nftables** –
  + *Advantages for nftables:* The functionalities the client desires to cover with iptables are covered by nftables.
  + *Disadvantages of nfttables:*

## 3.7. Consideration 7 - Fuzzer

* **American Fuzzy Lop (AFL)** – Generates arbitrary fuzzed packets that will allow the system analyst to send randomized packets in order to test and trigger hooks and collect data.
  + *Advantages for AFL:* Allows the analyst to generate as many packets as desired and are all randomized in which trims the work load of creating new and different packets in order to test and trigger hooks.
  + *Disadvantages of AFL:* Relies on feedback from the system and the user. Built-in feedback that AFL provides does not produce solutions for errors in code or in testing cases and does not guide the user for a basic framework on how to use AFL.

# 4. Solutions

This section lists our solutions that will be chosen from the considerations section. In our solutions we will include the requirements met, not met and the resources needed. There will be a minimum of two solutions for this section.

## 4.1. Solution 1

In this solution, we would construct the system by focusing around the programming language, python, this language was selected because the client's needs did not specify which language to use. We would use Scapy as our packet interceptor; this also works with our system since we chose a tool written in python. Another need from the user is that we would like to use Kali as stated in the presentation, Scapy runs natively on Linux. Scapy is also a well-known packet manipulation tool and a major need for the client is to be able to modify packets.

(I know we have more reasons to use Scapy over other options, I am googling rn) and sniffer and AFL as the fuzzer for our system. Our system would handle IP-Tables and hooks by itself. Once the system has a graphical user interface, it will meet all the requirements.

## 4.2. Solution 2

In this solution, instead of focusing on python, we would focuz on using C/C++. This means that we would use T-shark as our interceptor/sniffer. We would still use AFL as our fuzzer, and the IPTables and hooks would be handled internally by our system. A graphical user interface will also be made for this system. This solution will also meet the requirements.

# 5. Comparison of Solutions

Both of our solutions are expected to run with Kali Linux, which should not be a problem. The computer that is running the hardware is also expected to be able to handle both systems specified in each requirement. The computer is expected to have at least a 7th generation i5 core and at least 8gb of RAM as said by the client.

The time to build both systems should be about the same and should be doable in one semester. This is only the case assuming that we can use outside software as our interceptors/sniffers and fuzzers. Otherwise, the time needed could increase by a wide margin.

Both systems should provide about the same ease of use. This is because our system will implement a graphical user interface to make it easy for the user to do what they want in a simple manner. The interface might be aesthetically different because two different languages will be used, but its core concept will remain the same.

The amount of training required will be different for each solution because the programmers would have to learn python for one solution and C/C++ for the other. This will depend on the programmers and what language they are more familiar with; however, python is usually known to be a simpler language to learn than C/C++. In other words, the first solution might require less time in learning the language that will be used. For both systems, the team will have to learn as much as possible about network programming and building graphical user interfaces. The amount of training will be the same for both solutions in that regard. The only concern will be learning the programming language.

As both solutions include having a graphical user interface, they should both allow the same amount of user preference. The user will be able to do what they want and how they want as long as they are inside the limitations of the user interface.

In terms of security, python is usually considered more secure than C/C++. More specifically, C/C++ is prone to memory leakage.

The following table summarizes the comparison for both solutions.

|  |  |  |
| --- | --- | --- |
|  | Solution 1 | Solution 2 |
| Hardware/Software req. | Kali Linux  7th gen i5, 8gb RAM | Kali Linux  7th gen i5, 8gb RAM |
| Time constraints | Fall semester of 2018. | Fall semester of 2018. |
| Ease of use | Graphical user interface to make it easy for user. | Graphical user interface to make it easy for user. |
| Staffing levels/Training | Team must know/learn python (3).  Must understand networking systems. | Team must know/learn C/C++.  Must understand networking systems. |
| User preference | Allowing the user to modify or view anything they wish with no limitation. | Allowing the user to modify or view anything they wish with no limitation |
| Security issues | Python is considered to be more secured. | Since this is being programmed in C/C++ this may cause issues with memory. |

# 6. Conclusions

The production of the NTPS seems achievable in the scope of a semester. The modules assessed for the design of this project were fundamentally related in that they achieved the same result when combined properly. We categorically emphasize that our two proposed solutions can be produced and implemented with the necessary resources.

We recommend the second of the possible solutions presented in this document as it reflects a natural, cohesive nature of the modules we must integrate to achieve the product. Option 2 also represents a solution that is more prone to succeed to sudden changes in implementation expectations, if needed. Nevertheless, we emphasize that the first implementation/solution option represents fundamentally the same approach but with the advantage of the flexibility offered by Phyton. In terms of the overarching element that the operating system is we offer the following perspective. Since Kali Linux is designed to be a secured system, our concern deals with the adaptability and cohesiveness of ‘foreign’ languages like python to those domestic like C and C++.

Whether is finally of interest to the client to have an open possibilities software is part of this report. But we believe immediate and progressive adaptions of other functionalities are more easily achieved by using implementation option 2. We resolved that tools not mentioned in this set of solution options can be left as plan “B”s. We finally, remark that the components of the system seem to have non-obvious deficiencies that even as non-obvious circumstances do not affect the achievement of the use-cases/tasks for which the system is designed but do pose a possibility of increasing the necessity of training before a user could sufficiently understand all the functionalities of the system regardless of previous experience.

# 7. References

[1] J. Acosta, C. Murga, C. Zapata, “Network Traffic Proxy System,” 2018.

[2] L.M. Garcia, “TCPDump & LibPCAP,” *tcpdump.org*, 2010. [Online]. Available: <http://www.tcpdump.org/>. [Accessed Oct. 8, 2018].

[3] T. Fortunato, “Wireshark Packet Capture: Tshark Vs. Dumpcap,” *NetworkComputing.com*, June 5, 2018. [Online]. Available: <https://www.networkcomputing.com/networking/wireshark-packet-capture-tshark-vs-dumpcap/1149900924>. [Accessed Oct. 8, 2018].

[4] K. Grigorenko, “Wireshark Automation with Lua Scripting,” *Kevin Grigorenko's IBM WebSphere SWAT Blog*, Dec. 9, 2018, IBM. [Online]. Available: <https://www.ibm.com/developerworks/community/blogs/kevgrig/entry/wireshark_automation_with_lua_scripting?lang=en>. [Accessed Oct. 8, 2018].

[5] P. Biondi and the Scapy community, “About Scapy,” *Scapy*. [Online]. Available: <https://scapy.net/>. [Accessed Oct. 8, 2018].

[6] C. A. Lucas, “Python network packet dissection frameworks shootout: Scapy vs Construct vs Hachoir vs Kaitai Struct,” *Adventures in Python*, Mar. 9, 2017. [Online]. Available: <https://pythonistac.wordpress.com/2017/03/09/python-network-packet-dissection-frameworks-shootout-scapy-vs-construct-vs-hachoir-vs-kaitai-struct/>. [Accessed Oct. 8, 2018].

[7] M. Casado, “The Sniffer’s Guide to Raw Traffic,” [Online]. Available: <http://yuba.stanford.edu/~casado/pcap/section1.html>. [Accessed Oct. 7, 2018].

[8] D. Miessler, “Tcpdump Primer and Tutorial With Examples,” *DanielMiessler*, Jan. 4, 2004. [Online]. Available: <https://danielmiessler.com/study/tcpdump/#protocol>. [Accessed Oct. 7, 2018].

[9] M.Zalewski, “american fuzzy lop,” 2016. [Online]. Available: <http://lcamtuf.coredump.cx/afl/README.txt> [Accessed Oct. 9, 2018]

[10] G, Yang, C. Feng, C. Tang, “Static analysis assisted vulnerability-oriented evolutionary fuzzing,” *AIP Conference Proceedings*, Mar. 13, 2017. [Online]. Available: <https://aip.scitation.org/doi/pdf/10.1063/1.4977410> [Accessed Oct. 9, 2018]

[11] J. Regehr, “What afl-fuzz Is Bad At,” *Embedded In Academia*, May 3, 2015. [Online]. Available: <https://blog.regehr.org/archives/1238> [Accessed Oct. 9, 2018].

[12] O. Awodele, O. Oluwabukola, A.C Ogbonna, A.Adebowale, “Packet Sniffer – A Comparative Characteristic Evaluation Study,” *Proceedings of Informing Science & IT Education Conference (inSITE)*, p. 91-100, 2015. [Online]. Available: <http://proceedings.informingscience.org/InSITE2015/InSITE15p091-100Awodele1537.pdf> [Accessed Oct. 9, 2018]

[13] M. Tsoukalos, “Using tshark to Watch and Inspect Network Traffic,” *Linux Journal,* Aug. 31, 2015*.* [Online]. Available at: <https://www.linuxjournal.com/content/using-tshark-watch-and-inspect-network-traffic> [Accessed Oct. 9, 2018].

[14] P. Kruchten, *The Rational Unified Process: An Introduction, Second Edition.* Boston, MA: Addison-Wesley Longman Publishing Co., 2000.

[15] PortSwigger Ltd, “Burp Suite Editions,” *PortSwigger Web Security.* [Online]. Available: <https://portswigger.net/burp>. [Accessed Oct. 13, 2018]

[16] Wireshark Foundation, “About Wireshark,” *Wireshark*. [Online]. Available: <https://www.wireshark.org/>. [Accessed Oct. 13, 2018].

[17] P. Biondi and the Scapy community, *tshark – Dump and analyze network traffic,* Wireshark Foundation. [Online]. Available: <https://www.wireshark.org/docs/man-pages/tshark.html> [Accessed Oct. 13, 2018].

[18] BackBox Team, “Backbox.org,” *Backbox*. [Online]. Available: <https://www.backbox.org>. [Accessed Oct. 13, 2018].

[19] Fedora, “Security Lab,” *fedoraproject*, 2017. [Online]. Available: <https://labs.fedoraproject.org/en/security>. [Accessed Oct. 13, 2018].

[20] Offensive Security, “Our Most Advanced Penetration Testing Distribution, Ever,” *Kali.* [Online]. Available: <https://www.kali.org>. [Accessed Oct. 13, 2018].

[21] <https://linux-audit.com/differences-between-iptables-and-nftables-explained/>