Network Traffic Proxy System

Version 1.6

10/20/18

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: | 19 |
| 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 | 10/14/2018 | Oscar Galindo | Updated section 1.3 and 1.4, organized section 3. Included the solutions of section 4 and compared solutions in section 5. |
|  |  |  |  |

Table of Contents

[Document Control ii](#_Toc527306029)

[Approval ii](#_Toc527306030)

[Document Change Control ii](#_Toc527306031)

[Distribution List ii](#_Toc527306032)

[Change Summary ii](#_Toc527306033)

[1. Introduction 5](#_Toc527306034)

[1.1. Purpose of the Feasibility Report 5](#_Toc527306035)

[1.2. Justification for the Proposed System 5](#_Toc527306036)

[1.3. Requirements Definition 5](#_Toc527306037)

[1.4. Use Cases 6](#_Toc527306038)

[2. Existing Solutions 7](#_Toc527306039)

[2.1. Complete System Solutions 7](#_Toc527306040)

[3. Considerations 8](#_Toc527306041)

[3.1. Consideration 1 - Operating System 8](#_Toc527306042)

[3.2. Consideration 2 - Programming Languages 8](#_Toc527306043)

[3.3. Consideration 3 - Network Sniffer 9](#_Toc527306044)

[3.4. Consideration 4 - Packet Replayer 9](#_Toc527306045)

[3.5. Consideration 5 - Enqueueing Tool 10](#_Toc527306046)

[3.6. Consideration 6 - Packet Filter 10](#_Toc527306047)

[3.7. Consideration 7 - Fuzzer 11](#_Toc527306048)

[4. Solutions 12](#_Toc527306049)

[4.1. Solution 1 12](#_Toc527306050)

[4.2. Solution 2 12](#_Toc527306051)

[5. Comparison of Solutions 14](#_Toc527306052)

[6. Conclusions 15](#_Toc527306053)

[7. References 16](#_Toc527306054)

# 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]

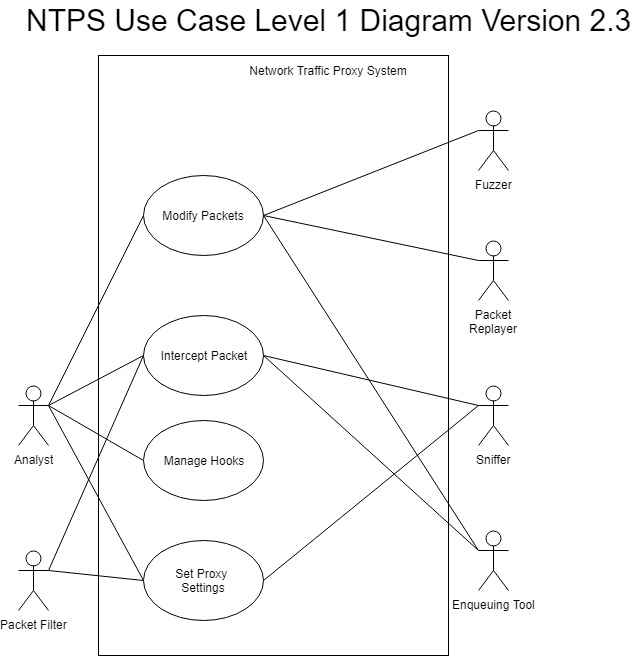
Graphically, the NTPS should support a display of all the system’s functionalities, including the packets being “sniffed” or being seen by the network sniffer, as graphic elements or options to execute in a graphical user interface (GUI). These options should include the ability to introduce hooks (scripts) 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].”

## 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. [15]

* **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. [16]

* **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 [1] 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

As an important consideration for building our system we consider what programming languages might be the best choice to develop our platform. Our analysis is purely based in what languages are the other solutions we have to implement develop on, our logic is that implementing our system on the language where most of our tools are developed we could reduce the compatibility errors at implementation stages.

* **C**
* **C++**
* **Python**

## 3.3. Consideration 3 - Network Sniffer

The Network Traffic Proxy System needs to implement the functionality of intercepting and, if necessary, decoding the packages that are flowing through the network. Network Sniffers, also referred to as packet analyzers allow the system to perform dissection and capturing of packets. For this reason, we list the options we considered to include this functionality in our system.

* **Scapy** – A packet manipulation program that can read and decode packets, modify values, and create packets. Scapy not only has the capability to do packet manipulation, but it can also “sniff” a network with a function called sniff. [26]
  + *Advantages of Scapy:* Since Scapy is a tool for packet manipulation when we like to give the user high level functions which also include the “sniff” function, multi-tasking becomes easier and so does extensibility [24].
  + *Disadvantages of Scapy:* Cannot handle many packets simultaneously and offers only partial support for certain complex protocols and partial support for the more complex protocols. [24]
* **T-Shark/Wireshark** – 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. [17]
  + *Advantages of T-Shark:* Can be used inside a script, or in the case of the project, could be used within hooks [35]. An additional benefit is that our client’s need for PCAP format and T-Shark's native capture file format is PCAP. [17] Another advantage 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.[16, 37]
  + *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. [35]
* **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. [2]
  + *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.[2]
  + *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. [12]

## 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 can 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** – Can receive/capture and send/forward packets. Scapy also allows to specify how many packets a user would like to send, whether it be one packet or 1,000 packets. [26]
  + *Advantages of Scapy:* An advantage of Scapy being used as a replayer is that it can replay as many packets as desired.
  + *Disadvantages of Scapy:* Disadvantages of Scapy being used as a replayer is that when it comes to the traceroute of packets, Scapy sends all packets through and does not know when to stop, thus displaying each packet’s traceroute result. A traceroute shows the complete route to a destination address. It also shows the time taken (or delays) between intermediate routers. Other traceroute programs wait for each node to reply before moving on to the next traceroute of a packet.
* **Tcpreplay** – Offers PCAP editing and replaying utilities that are open sourced and are designed for UNIX systems. It was originally created to replay malicious traffic patterns to Intrusion Detection/Prevention Systems, it has seen many evolutions including capabilities to replay to web servers. [38] The goal of tcpreplay is to provide the means for providing reliable and repeatible means for testing a variety of network devices such as switches, router, firewalls, network intrusion detection and prevention systems (IDS and IPS). [39]
  + *Advantages of TCPreplay:* Tcpreplay provides the ability to classify traffic as client or server, edit packets at layers 2-4 and replay the traffic at arbitrary speeds onto a network for sniffing or through a device.
  + *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 - Interceptor

The Network Traffic Proxy System should be able to create a queue out of the packets intercepted and leave the decision of what to do with them to the userspace or to the application the analyst uses to manage the packets.

* **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

The Network Traffic Proxy System should be able to control the flow of packets throughout the network when acting as a proxy. For this it is necessary to implement software that handles the communication procedures at the kernel level, either by addressing the queue of packets the system should conform or by simply choosing to handle communication of packets in a specific way. In order to satisfy this requirement, we offer the following solutions:

* **ip-tables** – Is a packet filter software that allows an analyst to set rules of transmission at the kernel level pointed to drop or allow the flow of certain type of packets or selected destinations and sources.
  + *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. [22] Performance decays as more packets are scrutinized. [23]
* **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

Fuzzing software is able to generate random data which can then be used to test programs and see how the system would react to seeing different types of data. In the case of the Network Traffic Proxy System, a fuzzer will be used to generate packets with random data, which will then be used to test the network. There are normally two types of fuzzing, mutation fuzzing and generation fuzzing. Mutation based fuzzing, also known as “dumb” fuzzing, is when the fuzzer need to know the format of the data that it needs to produce. These fuzzers are usually easier and faster to use because they can just produce random data without worrying about having to meet any standards. Generation fuzzing, on the other hand, needs to know the format of its data, and it will then produce random data that meets the requirements [28]. It will depend on the software as to what type of fuzzing it supports. The client requested that our system should support AFL, but we will describe a couple of other fuzzers to take into consideration.

* **American Fuzzy Lop (AFL):** AFL is a popular generation type fuzzer that “is designed to be practical [29].” It has been developed with the intention of being optimal and effective.
  + *Advantages for AFL:* It supports a lot of varying types of formatting. It uses “high-gain test case preprocessing and fuzzing strategies [29]” to find important bugs. AFL is fast and optimized, stable, and can be used alongside other tools. It is also known to be user friendly and simple to use.
  + *Disadvantages for AFL:* Because AFL uses file input to decide what and how to fuzz, it can be difficult to use for network fuzzing. There are solutions that have been found to work around this problem including, using other software, using a different version of AFL that is not by the same team, and using the recently added persistent mode. [30, 31]
* **Radamsa:** Radamsa is a mutation based fuzzer that is “intended to be a good general purpose fuzzer for all kinds of data [32].”
  + *Advantages for Radamsa:* Radamsa does not need to worry about the formatting of the data. It can just produce random data to the system. This could make fuzzing specific fields of a packet easier and simpler. [32]
  + *Disadvantages for Radamsa:* Because of its general-purpose scope, using Radamsa would mean that generating new packets that need to meet a specific protocol would be difficult. [32]
* **Sulley:** Sulley is a fuzz testing framework that can, not only fuzz different kinds of files, but also “detects, tracks and categorizes detected faults [33].”
  + *Advantages for Sulley:* Sulley is meant to be used for extended periods of times, so that the user can send multiple fuzzed files, and come back at any time to see any errors or faults in the system. Sulley uses python scripts in blocks to create different kinds of data generation. Sulley can be used for network fuzzing and to monitor the network itself. [33]
  + *Disadvantages for Sulley:* It could potentially difficult to integrate with the system, especially if the system does not use python. The analyst might want to analyze everything on his own terms, which would make a lot of the features of this software useless. [33, 34]

# 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. We allude to our two preferred solutions.

## 4.1. Solution 1

For this solution the system will be implemented in Kali Linux. Following the logic that the client desires this operating system we would suggest using this operating system. The use of Kali Linux will reduce the flexibility of the tools we can implement and so we have decided to introduce a specific set of tools we believe were developed for systems like Kali Linux; these tools are: as main programming language C++ due to its versatility in implementing the functionality from object oriented programming languages, and still have the necessary “reach” to do operations as C typically does. As network sniffer we would choose to use t-shark due to its functionality in a command line fashion in comparison to Wireshark and contains a much richer pool of selectivity parameters or filters than TCPdump. In comparison with Scapy we believe performance of t-shark at execution is more efficient since Scapy, as does its parent programming language, Python, interprets the code of the user. Instead, Wireshark, where the API and the implementation are based on C, compiles and executes highly efficient version of our system. In addition, we have confirmed it is possible to add a functionality of Python code to our C++ platform, for further explanation please access reference [36].

As a packet replayer we will include TCPreplayer. Making use of our ability to embedded TCP’s API functionalities into our T-shark system we will attempt to provide the user with a much simpler version of a replayer, since, even though the functionalities Scapy provides to modify packets are much more elevated in scope (e.g. adding layers to the packets), TCPreplayer will provide the user in exchange for modifiability a faster and more compatible replayer to the entire system. Nevertheless, TCPreplayer will not lose its ability to modify layers 2, 3, and 4 of every packet, which can be another exploitable feature for our system. As an enqueueing tool we will suggest the use of NFQUEUE, because its functionality allows to access the packets that are intercepted by the rules applied to iptables, as well as it allows us to make a queue of packets for the analyst to interact with more easily. As a packet filtering software, we would suggest the use of iptables as it is included as part of most Linux systems, including Kali Linux, our operating system choose in this solution. Iptables also has the advantage that it is acting as a firewall for our system and that allows us to make specific decisions on the basis of multiple characteristics, like protocols, ip, etc. Finally, as fuzzer we would suggest including the American Fuzzy Lop, because the client is already accustomed to it, and secondly because the process can be trained to perform specific fuzzing procedures. Nevertheless, the would a complication to the system as it will be increasingly difficult for a medium level analyst to learn all the sophisticated techniques of the system.

## 4.2. Solution 2

For this solution the system will be implemented in Kali Linux. Following the logic that the client desires this operating system we would suggest using this operating system. The use of Kali Linux will reduce the flexibility of the tools we can implement and for this second solution we recommend the use of the following tools: as programming languages we suggest the implementation of Python, due to the increase support there is in open source communities to allow the system to be more maintainable and to a degree more practical than other options. As Network Sniffer we would suggest to use Scapy due to its direct relationship with the programming language we will based our system on and the “capacity” of the library to perform many tasks that even other combinations cannot perform. As in solution 1 we suggest the inclusion of NFQueue as the enqueuing tool of our system since it can access the kernel level of the operating system and manage packets with userspace determinations, or in other words, it allows an analyst and the software he or she is using to specify a dropping, forwarding, etc. procedures; Following a similar logic than in solution 1 we suggest the inclusion of C code in our Python system. This last can be done and shall more information be needed please refer to reference [36].

As part of this possible second solution we suggest using Scapy as a packet replayer due to its implicit decrease in necessity of integration of more software and explicitly because of other ranges of features like the capacity to add extra layer of information to the same packet. This last is with the intention to provide analysts using our software with a much richer variation of a packet replayer, and network sniffer. Finally, for the packet filtering we will suggest the use of nftables because it is a direct substitution to the already available iptables with the intention of speeding up the processes of packet filtering, while it includes the same functionalities and still the combination of functionalities with NFQueue is possible. In addition, we suggest the use of AFL because the client learning curve to get accustomed to the system is smaller, and due to the specific circumstances of the client, friendlier.

# 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.

The ease of use between both systems is quite similar due to the use of similar software solutions, nevertheless, solution 2 can be thought of being “easier” to use since many of the functionalities are implemented through Scapy. In terms of the staffing levels and training needed for the system to be “codeable” by our team we need to have similar training sessions where we can see experts intercepting, capturing, and replaying among other functionalities like applying/importing hooks to the system.

In terms of direct comparison of solutions, we have same implementation of operating systems, enqueueing tools, fuzzers and hardware configurations. The obvious distinction between the systems we proposed is the programming language we intend to use. In the case of solution 1 our system is expected to use C++ to keep functionalities that are familiar to the team, while implementing the tools preferred by the client. In solution 2 we base our system in Python to propose different solutions to those of the client. Overall, we also present these options as means to indicate that the implementation will be key to the efficiency of our system. An example of this, is the consideration in the velocity of execution between languages, because while C++/solution 1 will be compiled and executed, Python/solution 2 will be interpreted while executed, this represents a challenge for the development of our solutions as the programming implicit shortcoming (e.g. capacity of processing power) will determine how our system will perform on different scenarios.

In terms of network sniffers our solutions differ in that on one side we have a very extendable software like Scapy that can implement hundreds of different subroutines to make the system more reliable (e.g. capture failure recovery). While in solution 1 we implement a finished product that is available to be used from day zero of development. Our assessment is that these two options show an option of flexibility against an option probable, not definite reliability, even though both software solutions use the same libraries for their network sniffing operations. In relation to packet replayers and packet filters we present a direct comparison of two solutions that juxtapose newer options like Scapy and NFtables, which also represent a modern and platform, but that at best is in development. The other combination of solutions deals with TCPreplayer and iptables, which are tested, proven solutions that have been in development at least a decade. What we see of difference in these comparisons is a desire for the client to have a “single-use”/regular use of the system, or instead the client prefers to have the option to adapt and extend by paying the cost of reliability.

|  |  |  |
| --- | --- | --- |
|  | Solution 1 | Solution 2 |
| Operating System | Kali Linux 2018.2 | Python is considered to be more secured. |
| Programming Language | C++ | Python |
| Network Sniffer | T-shark | Scapy |
| Interceptor | NFQUEUE | NFQUEUE |
| Fuzzer | AFL | AFL |
| Packet Replayer | TCPreplayer | Scapy |
| Packet Filter | Iptables | NFtables |
| 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.  Higher number of dependencies due to increased number of use of different software solutions.  Increased learning gap to the average analysts due to complexity of fuzzing. | Graphical user interface to make it easy for user.  Less system dependencies due to lower number of modules, due to less use of different software solutions.  Increased learning gap to the average analysts due to complexity of fuzzing. |
| Staffing levels/Training | Team of five engineers must know/learn C/C++ since most of us have medium level skills on C/C++.  Team of five engineers must obtain training in basic procedures of the system like, but not limited to, capturing, modifying PCAP and replaying packets.  Staffing Levels:  Systems Architect  Systems Analyst  Designer  Lead Programmer | Team of five engineers must know/learn Python since most of us have beginner level comprehension of Python.  Team of five engineers must obtain training in basic procedures of the system like, but not limited to, capturing, modifying PCAP and replaying packets.  Staffing Levels:  Systems Architect  Systems Analyst  Designer  Lead Programmer |
| User preference | OS: Kali Linux  Programming Language: Python  Network Sniffer: T-shark  Enqueuing Tool: NFQueue  Fuzzer: AFL  Packet Replayer: Any  Packet Filter: Iptables | OS: Kali Linux  Programming Language: Python  Network Sniffer: T-shark  Enqueuing Tool: NFQueue  Fuzzer: AFL  Packet Replayer: Any  Packet Filter: Iptables |
| Security issues | Reliability is decreased because of more inclusion of software solutions, security is decreased because of potential increase of exploits in the system.  Authentication of users is not required in the system (listed due to potential harm to other systems). | Reliability is increased, maintainability is increased, security is still open to exploits because many solutions are still open sourced  Authentication of users is not required in the system (listed due to potential harm to other systems). |

# 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 first possible solution due to its close resemblance of the user preferences. We strongly suggest an assessment is done of both options thinking about the probable use of the solutions vs. current and fixed use of the solutions. Our assessment is that solution 1 is what our client is expecting to see implemented, but solution 2 is what our client might need in a scope from 4 to 6 years. We emphasize though, that our learning curve for solution 2 is greater than that of solution 1.

Whether is finally of interest to the client to have an open possibilities software is not 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 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] M. Boelen, “Differences between iptables and nftables explained,” *Linux Audit*, Jan. 18, 2016. [Online]. Available: <https://linux-audit.com/differences-between-iptables-and-nftables-explained>. [Accessed Oct. 14, 2018].

[22] F. Westphal, “What comes after ‘iptables’? Its successor, of course: ‘nftables’,” *redhat Developer*, Oct. 28, 2016. [Online]. Available: <https://developers.redhat.com/blog/2016/10/28/what-comes-after-iptables-its-successor-of-course-nftables>. [Accessed Oct. 14, 2018].

[23] R. Brash, “What are the disadvantages of iptables?” *quora.com*, May 16, 2016. [Online]. Available: <https://www.quora.com/What-are-the-disadvantages-of-iptables> . [Accessed Oct. 14, 2018].

[24] ziednamouchi, “Scapy Project,” *Cybrary.IT*, Jan. 1, 2017. [Online]. Available: <https://www.cybrary.it/0p3n/scapy-project>. [Accessed Oct. 14, 2018].

[25] Cisco Systems, “Packet Replay With Scapy,” *Cisco*, Aug. 29, 2018. [Online]. Available: <http://docwiki.cisco.com/wiki/Packet_Replay_With_Scapy>. [Accessed Oct. 14, 2018].

[26] Mat, “Scapy Sniffing with Custom Actions, Part 1,” thePacketGeek, Oct. 1, 2013. [Online]. Available: <https://thepacketgeek.com/scapy-sniffing-with-custom-actions-part-1>. [Accessed Oct. 14, 2018].

[27] R. Gupta, “Computer Networks | Traceroute,” *GeeksforGeeks.* [Online]. Available: https://www.geeksforgeeks.org/computer-networks-traceroute. [Accessed Oct. 14, 2018].

[28] P, Garul, “Fuzzing – Mutation vs. Generation,” Infosec Institute, Jan. 4, 2012. [Online]. Available: [https://resources.infosecinstitute.com/fuzzing-mutation-vs-generation/#gref](https://resources.infosecinstitute.com/fuzzing-mutation-vs-generation/). [Accessed Oct. 14, 2018].

[29] M. Zalewski, “American fuzzy lop (2.52b),” lcamtuf.coredump.cx. [Online]. Available: <http://lcamtuf.coredump.cx/afl>. [Accessed Oct. 14, 2018].

[30] J. Foote, “How to fuzz a server with American Fuzzy Lop,” fastly, Jul. 21, 2015. [Online]. Available: <https://www.fastly.com/blog/how-fuzz-server-american-fuzzy-lop>. [Accessed Oct. 14, 2018].

[31] H. Bock, “Network fuzzing with American fuzzy lop,” The Fuzzing Project, Oct. 27, 2015. [Online]. Available: <https://www.fastly.com/blog/how-fuzz-server-american-fuzzy-lop>. [Accessed Oct. 14, 2018].

[32] A. Helin, Radamsa. [Online]. Available: <https://gitlab.com/akihe/radamsa>. [Accessed Oct. 14, 2018].

[33] OpenRCE, Sulley. [Online]. Available: <https://github.com/OpenRCE/sulley>. [Accessed Oct. 14, 2018].

[34] P. Amini, A. Portnoy, “Fuzzing Sucks!” 2007. [Online]. Available: <https://github.com/OpenRCE/sulley/blob/master/docs/introducing_sulley.pdf>. [Accessed Oct. 14, 2018].

[35] M. Tsoukalos, “Using tshark to Watch and Inspect Network Traffic,” *Linux Journal,* Aug. 31, 2015. [Online]. Available: <https://www.linuxjournal.com/content/using-tshark-watch-and-inspect-network-traffic>. [Accessed Oct. 14, 2018].

[36] Python Software Foundation, *Python 2.7.15 documentation,* Python Software Foundation, Oct. 12, 2018. [Online]. Available: <https://docs.python.org/2/index.html>. [Accessed Oct. 14, 2018].

[37] UKEssays, “Cisco Prime and Wireshark Advantages and Disadvantages,” *ukessays.com,* Sep. 6, 2017. [Online]. Available: <https://www.ukessays.com/essays/information-technology/cisco-prime-wireshark-advantages-7229.php>. [Accessed Oct. 14, 2018].

[38] F. Klassen, “Tcpreplay - Pcap editing and replaying utilities,” *Tcpreplay.* [Online]. Available: <https://tcpreplay.appneta.com>. [Accessed Oct. 14, 2018].

[39] Trac Integrated SCM & Project Management, *Tcpreplay 3.x Online Manual,* Trac, 2012. [Online]. Available: <http://tcpreplay.synfin.net/wiki/manual>. [Accessed Oct. 14, 2018].