(Check -)Network Traffic Proxy System

Version <1.3>

Document Control

Approval

The Guidance Team and the customer will approve this document.

Document Change Control

|  |  |
| --- | --- |
| Initial Release: | 1.0 |
| Current Release: | 1.3 |
| Indicator of Last Page in Document: |  |
| Date of Last Review: | 11/10/18 |
| Date of Next Review: | 11/21/18 |
| Target Date for Next Update: | 11/21/18 |

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

Christian Murga

Caesar Zapata

Software Team Members:

Oscar Galindo

Isai Gonzalez

Kevin Gonzalez

Julio De La Cruz

Alan Caldelas

Change Summary

The following table details changes made between versions of this document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modifier | Description |
| 1.0 | 11/9/18 | Oscar, Julio, Kevin, Alan, Isai | Oscar-Added information to section 1.2 and 3.2.1 and Document Control  Isai- Completed section 1.1 and continued update to section 1.5  Julio- Finished section 1.4 and descriptions for 3.2.2  Alan- Completed sections 2.3, 2.4, and 2.5  Kevin-Completed sections 2.1 and 2.2 |
| 1.1 | 11/12/18 | Oscar, Julio, Kevin, Alan, Isai | As a team we added requirements for each class in our class diagram. |
| 1.2 | 11/21/2018 | Oscar, Julio, Kevin, Alan, Isai | Julio- Revised section 3.1.1 hook collection and fields, added requirements for the hook interface  Isai- Revised section 1.1 and the packet class part of section 3.1.1. Added requirements for the main, edit packet, and fuzzer interfaces.  Kevin – Added the save & load requirements. Revisited entire section of 2 and added an introduction for the entire section. Revised section 2.1 & 2.2.  Oscar - Revised sections 3.2.2 and 3.1.1. Finished section 3.1.3, revised sections 1.2 and 3.2.1 |
| 1.3 | 11/21/18 | Oscar Galindo | Revised sections 1.2, 3.2.1, 3.2.2, 3.1.1. And finished sections 3.1.3. |

Table of Contents

[Document Control ii](#_Toc529618079)

[Approval ii](#_Toc529618080)

[Document Change Control ii](#_Toc529618081)

[Distribution List ii](#_Toc529618082)

[Change Summary ii](#_Toc529618083)

[1. Introduction 4](#_Toc529618084)

[1.1. Purpose and Intended Audience 4](#_Toc529618085)

[1.2. Scope of Product 4](#_Toc529618086)

[1.3. Definitions, Acronyms, and Abbreviations 4](#_Toc529618087)

[1.3.1. Definitions 4](#_Toc529618088)

[1.3.2. Acronyms 5](#_Toc529618089)

[1.3.3. Abbreviations 5](#_Toc529618090)

[1.4. Overview 5](#_Toc529618091)

[1.5. References 6](#_Toc529618092)

[2. General Description 6](#_Toc529618093)

[2.1. Product Perspective 6](#_Toc529618094)

[2.2. Product Features 6](#_Toc529618095)

[2.3. User Characteristics 6](#_Toc529618096)

[2.4. General Constraints 7](#_Toc529618097)

[2.5. Assumptions and Dependencies 7](#_Toc529618098)

[3. Specific Requirements 7](#_Toc529618099)

[3.1. External Interface Requirements 8](#_Toc529618100)

[3.1.1. User Interfaces 8](#_Toc529618101)

[3.1.2. Hardware Interfaces 8](#_Toc529618102)

[3.1.3. Software Interfaces 8](#_Toc529618103)

[3.1.4. Communications Interfaces 8](#_Toc529618104)

[3.2. Behavioral Requirements 8](#_Toc529618105)

[3.2.1. Same Class of User 8](#_Toc529618106)

[3.2.2. Related Real-world Objects 9](#_Toc529618107)

[3.2.3. Stimulus 9](#_Toc529618108)

[3.2.4. Related Features 10](#_Toc529618109)

[3.2.5. Other 10](#_Toc529618110)

[3.3. Non-behavioral Requirements 10](#_Toc529618111)

[3.3.1. Performance Requirements 10](#_Toc529618112)

[3.3.2. Qualitative Requirements 10](#_Toc529618113)

[3.3.3. Design and Implementation Constraints 11](#_Toc529618114)

[3.4. Other Requirements 11](#_Toc529618115)

[3.4.1. Database 11](#_Toc529618116)

[3.4.2. Operations 11](#_Toc529618117)

[3.4.3. Site Adaptation 11](#_Toc529618118)

[4. Appendix 13](#_Toc529618119)

# Introduction

## Purpose and Intended Audience

The purpose of the Software Requirements Specification (SRS) document is to state a clear and precise description of the functionality of the product that our team will be building. The SRS document will serve as an agreement/contract, between Team 5 and the client team, of the requirements for the system. The client team, according to [1], is made up of Dr. Jaime Acosta, Christian Aaron Murga, and Mr. Caesar Zapata, and they are the primary intended audience of this document. It will be up to them to decide whether the system described in this document is what they want or not. If this document is accepted by the client team, the requirements stated in this document will be used by our team to build the system.

## Scope of Product

This project intends to suffice the client requirements of providing support for multiple protocols in multiple layers, being able to provide functionalities to modify packet contents, provide management tools for the transmission of packets, and provide a sufficient capacity to the system to be able to, at any moment of handling 100 packets at least. In addition, hooks (script), PCAP support, interception of packets either with specific capture filters or in general, as well as modification of packets either when intercepting, or during no interception should be supported [2]. Our focus is to integrate different software solutions which are used by our clients to perform network-communication security assessments through the use of proxy behaviors and packet layer modification functionalities. Precisely, our focus in integrating the solutions clients will like to utilize is having a centralized GUI that allows analysts to summon the functionalities they require within the same application, rather than having to access a different number of applications at once.

The Network Traffic Proxy System (NTPS), as our project is called, is intended so that an analyst making use of the system can perform the necessary operations to suffice the aforementioned requirements. This involves packet interception, packet modification either manually or through randomization software by making use of fuzzing software, as well as allowing the automatization of this modifications through the use of imported hooks (scripts) collections. An analyst using the NTPS will also be able to modify packet communication rules of the terminal where the NTPS is run. In addition, an analyst using the NTPS will be able to drop (erase) packets, forward (send) packets, and import PCAP (Packet Capture) files into the system. Furthermore, an analyst will be able to apply capture filters with specific parameters of capture making use of and as a storage solution the system will implement a queue of packets that will store the intercepted packets and will inform the analyst about the packets intercepted by the system with a regular load of at least 100 packets.

Uses for the systems of the kind the client wishes to be constructed are assessment of network security by injecting erroneous and unexpected information, observing the effectiveness of encryption techniques applied to packet contents, and assess behavior of network-communication into a specific network by looking at the packet transfer between terminals. In addition, proprietary protocols, which are develop for specific, secure purposes can also be observed with this technology and improved by it.

## Definitions, Acronyms, and Abbreviations

### Definitions

This section lists the terms used in this document and their associated definitions.

Table 1: Definitions

|  |  |
| --- | --- |
| **TERM** | **DEFINITION** |
| Field | Units that define the meaning and effect |
| Layer | Specification fragments found within a packet that indicates among other things, where the packet is going, where it is coming from, etc. |
| Modern Laptop | 7th gen core i5 8GB of ram |
| NFQUEUE | Tool utilized with IP tables software that permits the user to store packets until the user decides what to do with them. |
| Open Systems  Interconnection reference model | Defines a hierarchical layer architecture that logically partitions the functions required to support system-to-system communication. [4] Network systems communicate using a layered architecture to pass information among nodes. Data at each layer must follow a protocol specification so that nodes involved will be able to send, receive, and interpret information. [2] |
| Packet | Electronic communication units that represent the fragmentation of data into transmittable pieces. |
| Protocol | A set of rules that govern the communication and transmission of data by giving structure to every layer. |
| Proxy Rule | Routing policy applied to the terminal where NTPS is ran. |
| PyPacker | Python parsing and creation library |
| Scapy | A powerful interactive packet manipulation program. It can forge or decode packets of a wide number of protocols, send them on the wire |
| Sniff | An action that refers to being able to inspect or intercept something sort of data. |
| TCPDump | A common packet analyzer that runs under the command line |
| Terminal | Computer where we the NTPS system is run. |
| Wireshark | A network packet analyzer [3] |

### Acronyms

This section lists the acronyms used in this document and their associated definitions.

Table 2: Acronyms

|  |  |
| --- | --- |
| **TERM** | **DEFINITION** |
| NTPS | Network Traffic Proxy System |
| SRS | Software Requirements Specification Document |
| GUI | Graphical User Interface |
| ARL | Army Research Lab |
| AFL | American Fuzzy Loop |
| PCAP | Packet Capture File |

### Abbreviations

This section lists the abbreviations used in this document and their associated definitions.

Table 3: Abbreviations

|  |  |
| --- | --- |
| **TERM** | **DEFINITION** |
| Etc. | Etcetera |
| M.S. | Master of Science Degree |
| Ph.D. | Doctoral Degree |

## Overview

The SRS is divided into three major section listed below.

Section 1 contains the purpose of this document, the scope, definitions and acronyms, an overview, and references. The two tables in this document will contain the definitions of ambiguous terms and acronyms.

Section 2 provides an overall description of our product, its features, user characterizes, general constraints, and assumptions and dependencies our team has discovered throughout the design process.

Section 3 describes all of our requirements in detail for each aspect of the system; external interface requirements, behavioral requirements, non-behavioral requirements, and the last main section for other requirements we list.

## References

[1] E. Tai, “Interviewee Biographies” Interoffice memorandum (Sep. 19, 2018).

[2] J. Acosta, C. Murga, C. Zapata, “Network Traffic Proxy System” (Sep. 5, 2018).

[3] Team 5, “Network Traffic Proxy System Interview Report,” Report, Nov. 4, 2018.

[4] R. Miller, “The OSI Model: An Overview”, in *SANS*Standards. SANS Institute, [Online document], Sep. 13, 2001. Available: Sans.org https://www.sans.org/reading-room/whitepapers/standards/osi-model-overview-543 [Accessed 9 Nov. 2018].

# General Description

This section will propose a general description of the Network Traffic Proxy System. This section will cover the product’s overview perspective, such as a summary of the system and the important need for the system. A use case diagram will be presented in order to show functionalities and their relationships among other features. Essential features will be a main focus in this section in order to demonstrate what the system can accomplish and show case the client’s requirements. Afterwards, a description of each user that will be interacting with the system will be provided. General constraints will cover about constraints that will affect the development of the product. Considering time limitations, hardware limitations, and software limitations. The last sub section will list each factor that will affect the requirements that was stated in the SRS. These are assumptions and dependencies that are being made by the development team. If these assumptions and dependencies are not met, then SRS will be needed to change work properly.

## Product Perspective

The NTPS is a software program designed to be a security system that would allow an analyst to sniff how computers are communicating between each other within a network traffic. The system shall provide an analyst the capability to analyze and test data within protocols that are within the network traffic from multiple layers of the Open Systems Interconnection reference model. The NTPS overall structure will provide ARL with functionalities that are crucial to security testing to identify any weaknesses in communication between network services. The system shall supply functions through a graphical interface such as setting proxy rules and filters to intercept certain packets to be dissected and modified through the interface itself. The NTPS shall work with other backend processing software in order to help achieve the system’s overall performance.

## Product Features

A use case diagram was created to show the relationships between each user and each external software to each functionality of the system. The use case diagram has two levels to it. The level 1 use case diagram shows the relationships between users/external software and features of the system. The level 2 use case diagram shows these relationships as well as any extended, included, and generalization relationships. The level 2 use case diagram will be used to give an overview of each actor, of each external software actor, and of each essential feature that is an importance of how the system functions to meet the requirements. The level 2 use case diagram can be found in the ‘Appendix’ section.

**Actors:**

***Analyst***: Intermediate to expert lever user that will use the system for the security purpose of intercepting network traffic and modifying its artifacts to direct network flow in order to test and analyze.

***Packet Replayer***: An external software mechanism that will be used by the system from the decision of the analyst in order to forward packets to their destinations.

***Fuzzer***: An external software mechanism that will be used by the system from the decision of the analyst in order to fuzz packets to generate different field values for a packet.

***Enqueuing******Tool***: An external backend software mechanism that will be used by the system from the decision of the analyst to queue packets within the system.

***Network Sniffer***: An external software mechanism that will be used by the system to intercept packets from the network traffic in order to be analyzed.

***Packet******Filter***: An external software mechanism that will be used by the system to filter network traffic to intercept certain packets only.

**Use Cases:**

The main features of the system can be simplified into 2 main features, in which the first main feature is about intercepting packets from the network traffic. The second main feature is the ability to modify packets. Breaking these 2 main features down a bit, the intercepting feature includes sub features that assist in helping to intercept packets, such as set proxy settings, setting filters to intercept specific packets, applying hooks to packets, and being able to view and queue intercepted packets within the interface of the system. For modifying packets, sub features such as being able to view different formats of a dissected packet, manually changing the values of fields within a packet, fuzzing a packet, sending or dropping a packet, and being able save and load PCAP files. Some features were not modeled as use cases since they can be considered steps or atomic actions.

***Intercept Packet***: A required feature or function that needs to be able to allow the analyst to toggle on to intercept packets.

***Set Proxy Setting***: A required feature or function that needs to be able to allow the analyst to toggle on iptables to allow the system to intercept packets.

***Manage Hooks***: A required feature or function that needs to be able allow the analyst to load hooks, save hooks into collections, and delete hooks.

***View Packets***: A required feature or function that needs to be able allow the analyst to view intercepted packets and interact with the packets.

***Modify Packets***: A required feature or function that needs to be able to allow the analyst to modify packets that are either intercepted or loaded as in a PCAP file.

***Fuzz Packet***: A required feature or function that needs to be able allow the analyst to fuzz packets that are either intercepted or loaded as in a PCAP file.

***Send Packets***: A required feature or function that needs to be able to allow the analyst to forward packets from the system to their destinations.

## User Characteristics

The user of the system is going to be assumed to be an intermediate to expert analyst of Network Security Systems.

Intermediate User: This user would have a background in Computer Science with a few years of experience in networking technologies [3] and have an understanding on how nodes communicate. The user also should have some technical use of tools used in Networking such as WireShark, tcpdump, etc. This user should be able to do all basic functions that relate to filtering packets, interpreting python hook at a decent level, and the basics of dissecting packets and interpreting the basic structure of common packets. This user should be using a modern laptop [1]. This user should be using this system to analyze packets that are being communicated between systems for basic security testing of a Network Communication System [1] by getting between a communicating node with another while traffic is live.

Expert User: This user would have a background in Computer Science, with a M.S. or Ph.D., with a lot of experience in Cybersecurity in testing network systems, such as evaluating and identifying weaknesses that exist in communications systems. This user is proficient in using all common security tools such as using a dissector and are able can write their own. This user can specify different protocols and what is required for the connection to take place. [3] This user will be using this system for Security Testing in Network Systems that will be changing and sending packets to test for vulnerabilities in a Network System [2] by getting between a communicating node with another while traffic is live [3].

## General Constraints

* The development team is constrained to only about two semesters of the school year
* The system will be running on Kali Linux [3].
* The limitations of the hardware are that we assume that we are using a modern laptop [2].
* The dissector can handle encrypted packets [3]

## Assumptions and Dependencies

* The hardware will have a functioning network card that can receive packets
* The system will interact with AFL for fuzzing to interact with our system
* This system will interact with ARL Hooks
* Written in Python [3]
* Hooks that are for encrypted packets [3]
* Hooks that trigger for certain conditions met by the filter [3]
* This system will depend ARL having a dissector to use from the following:

1. Wireshark [3]
2. Scapy [3]

* The system will be interacting a packet replayer to forward packets from our system [3]

# Specific Requirements

<< Provides detail that is sufficient to allow the design of a system that will satisfy the requirements. >>

## External Interface Requirements

In this section, Team 5 will describe the requirements of the interfaces of our system. The section will have four subsections, User Interfaces, Hardware Interfaces, Software Interfaces, and Communication Interfaces. The specifications of requirements are all contained in the corresponding subsection. There may also be graphic representations of the interface(s) for the sake of clarity.

### User Interfaces

1. Each window shall disable any GUI elements that are not available for use under the current conditions of the system.
2. Each window, as shown in Figure 1, shall include following in its top section:
   1. An “Exit” button designed as an orange rectangle with a black “X” in the center.
   2. A “minimize” button designed as a white rectangle with a black “—“ in the center.
   3. The name of the system, “Network Traffic Proxy”, and the name of the feature in use, if applicable.
   4. A white rectangular button with the word “File” written on it in black.
   5. A white rectangular button with the word “View” written on it in black.
   6. A white rectangular button with the word “Help” written on it in black.

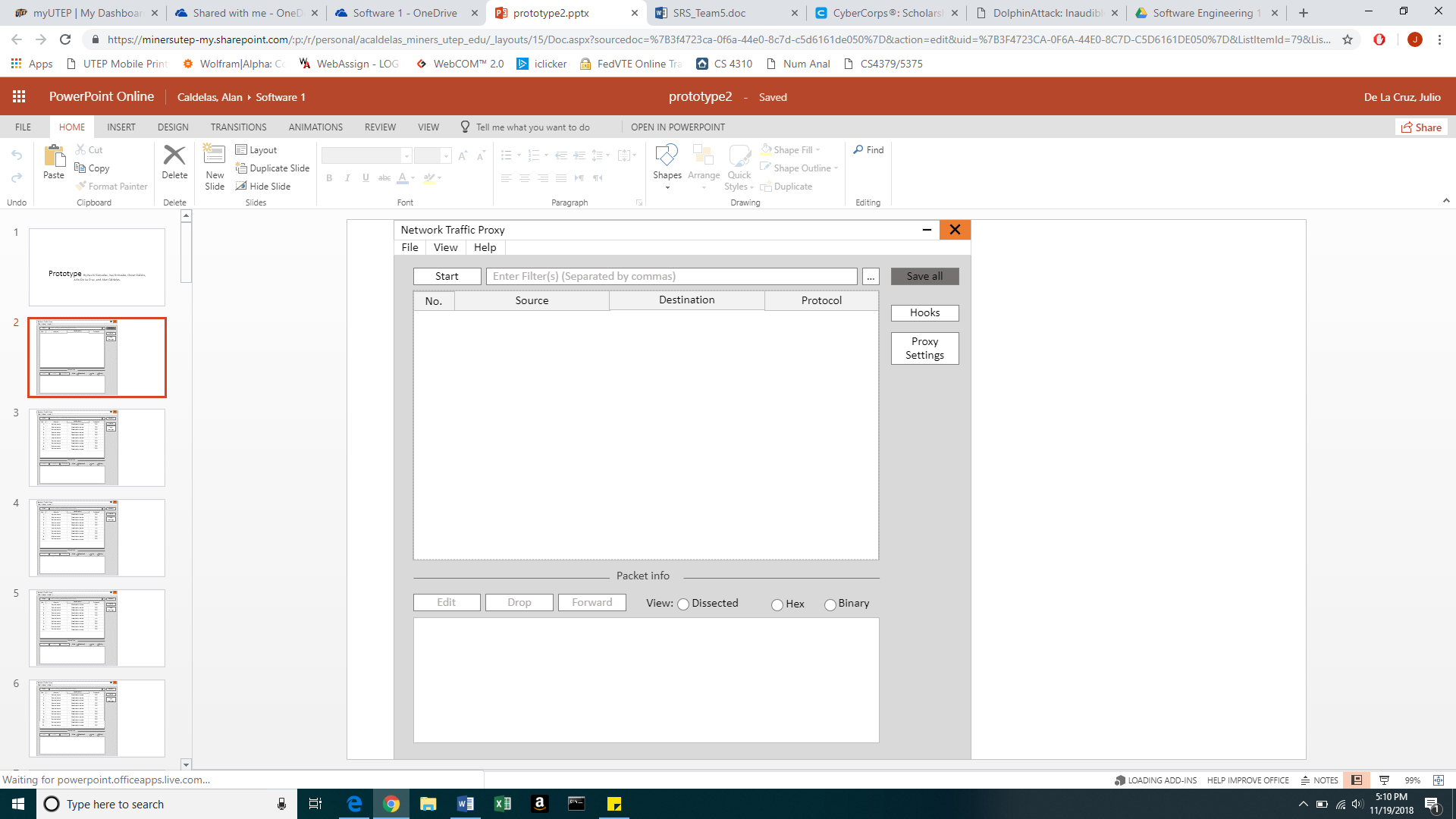


Figure 1: General windows top section

1. As shown in Figure 2, the “Home Screen Interface” window shall include the following:
2. A button labeled as “Intercept”.
3. A button labeled as “Enable Proxy”.
4. A text box labeled as “Enter Filter(s) (Separated by commas)”.
5. A button labeled as “…”.
6. A button labeled as “Save All”.
7. A button labeled as “Hooks”.
8. A rectangular bpanel that will display buttons corresponding to packets that have been intercepted by our system.
9. Buttons labeled with the number, source, destination, and protocol of a specific intercepted packet.
10. A line labeled “Packet Info” that splits the window into two sections.
11. A button labeled as “Edit”.
12. A button labeled as “Drop”.
13. A button labeled as “Forward”.
14. A section labeled as “View:” that has circular buttons labeled as “Dissected”, “Hex”, and “Binary”, respectively.
15. A rectangular panel that will display all of the information of each layer in a selected packet.

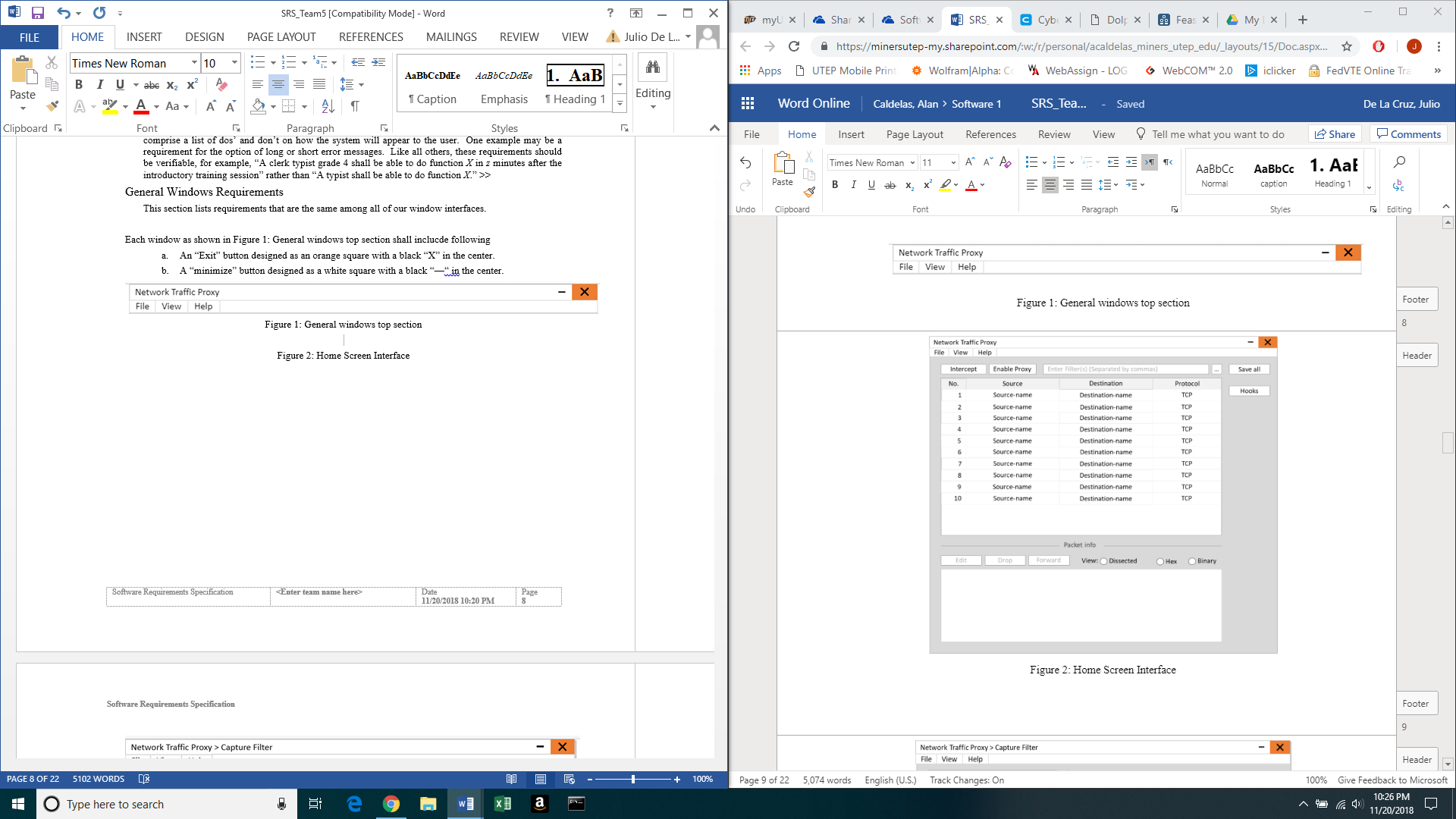


Figure 2: Home Screen Interface

1. As shown in Figure 3, the “Capture Filter Interface” window shall include the following:
   1. A field labeled as “Relation.”
   2. A field labeled as “Grammar.”
   3. A table with buttons labeled “Fields” with other fields “Destination”, “Protocol”, “Source.”
   4. A table with labels “Protocol” and “Port” with well-known protocols with the corresponding ports to apply to the filters.
   5. A toolbar field labeled as “Enter Filter(s) (Separated by commas)”.
   6. A button “Apply.”
   7. A table “Applied filters” with a table of current filters.
   8. A button “Remove.”
   9. A button “Clear All.”
   10. A button “Close.”.

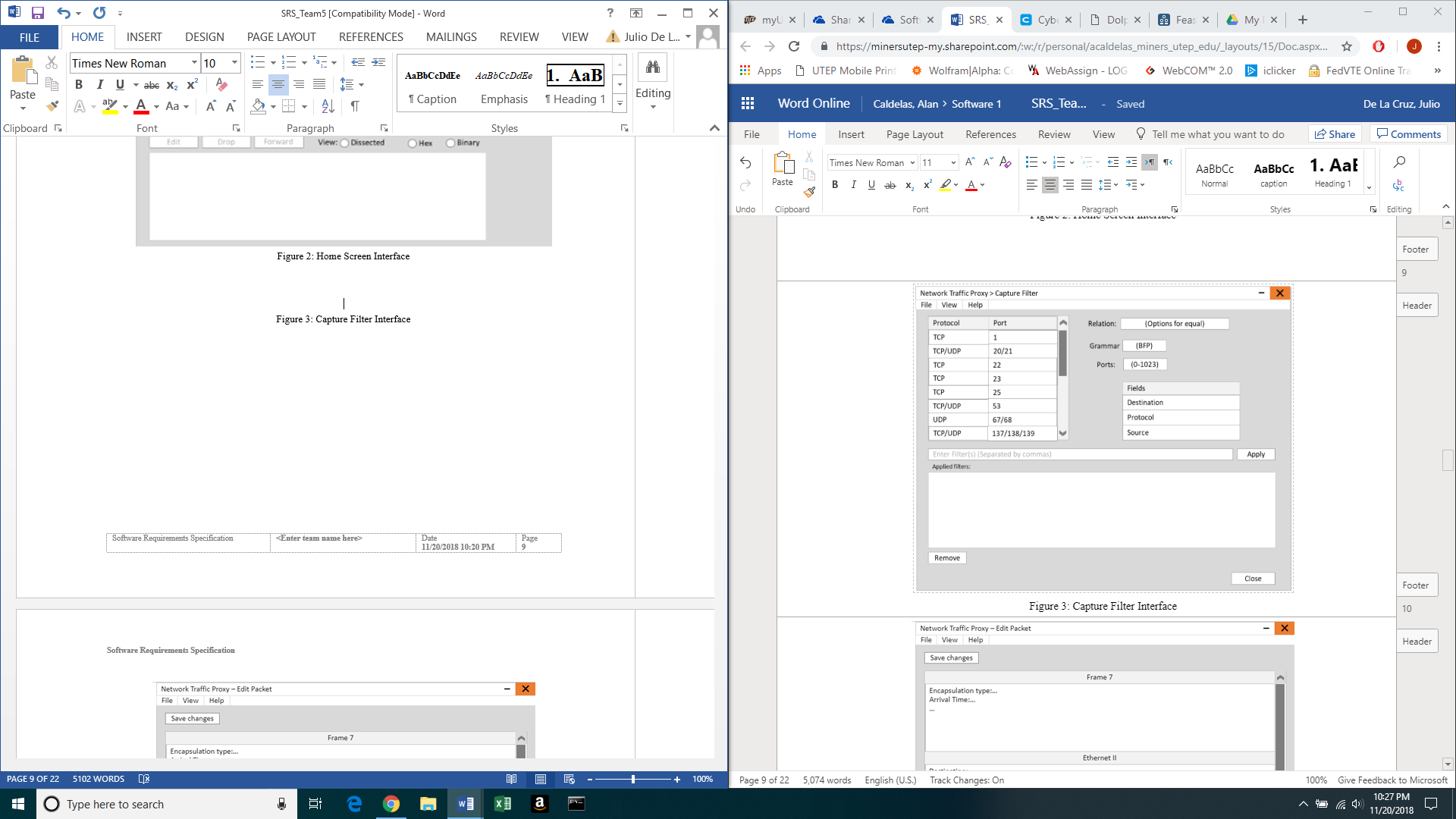


Figure 3: Capture Filter Interface

1. As shown in Figure 4, the “Edit Packet Interface” window shall include the following:
2. A button labeled as “Save Changes”.
3. A rectangular panel with the information of the selected packet, separated by layer, as well as a scroll bar if needed.
4. A button labeled as “Fuzz…”.
5. A button labeled as “Cancel”.
6. A button labeled as “Forward”.
7. A button labeled as “Drop”.
8. A button labeled as “Close”.

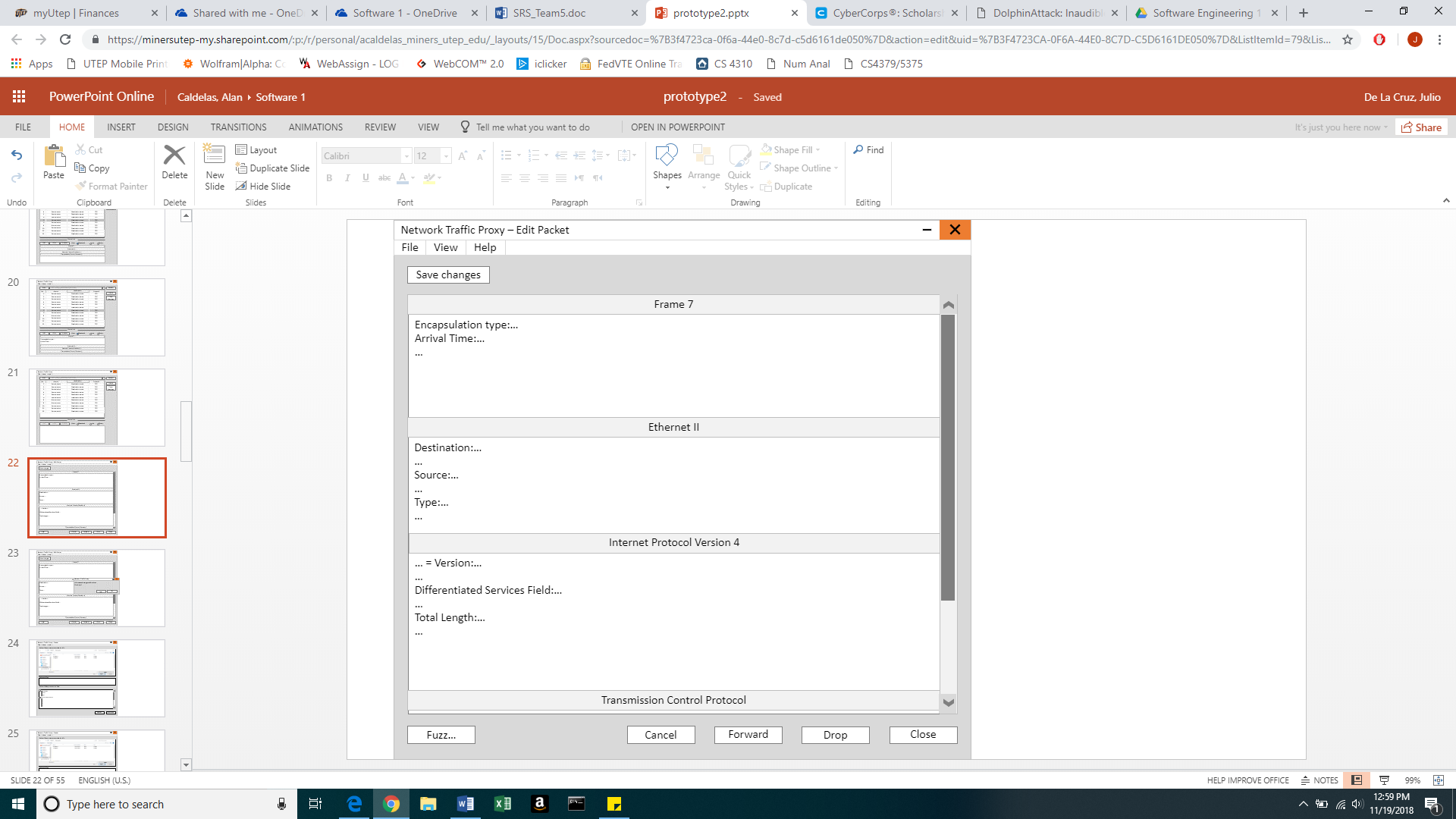


Figure 4: Edit Packet Interface

1. As shown in Figure 5, the “Unsaved Changes Confirmation Window” shall include the following:
2. Text display that says “All unsaved changes will be lost. Continue?”
3. A button labeled as “Yes”.
4. A button labeled as “No”.

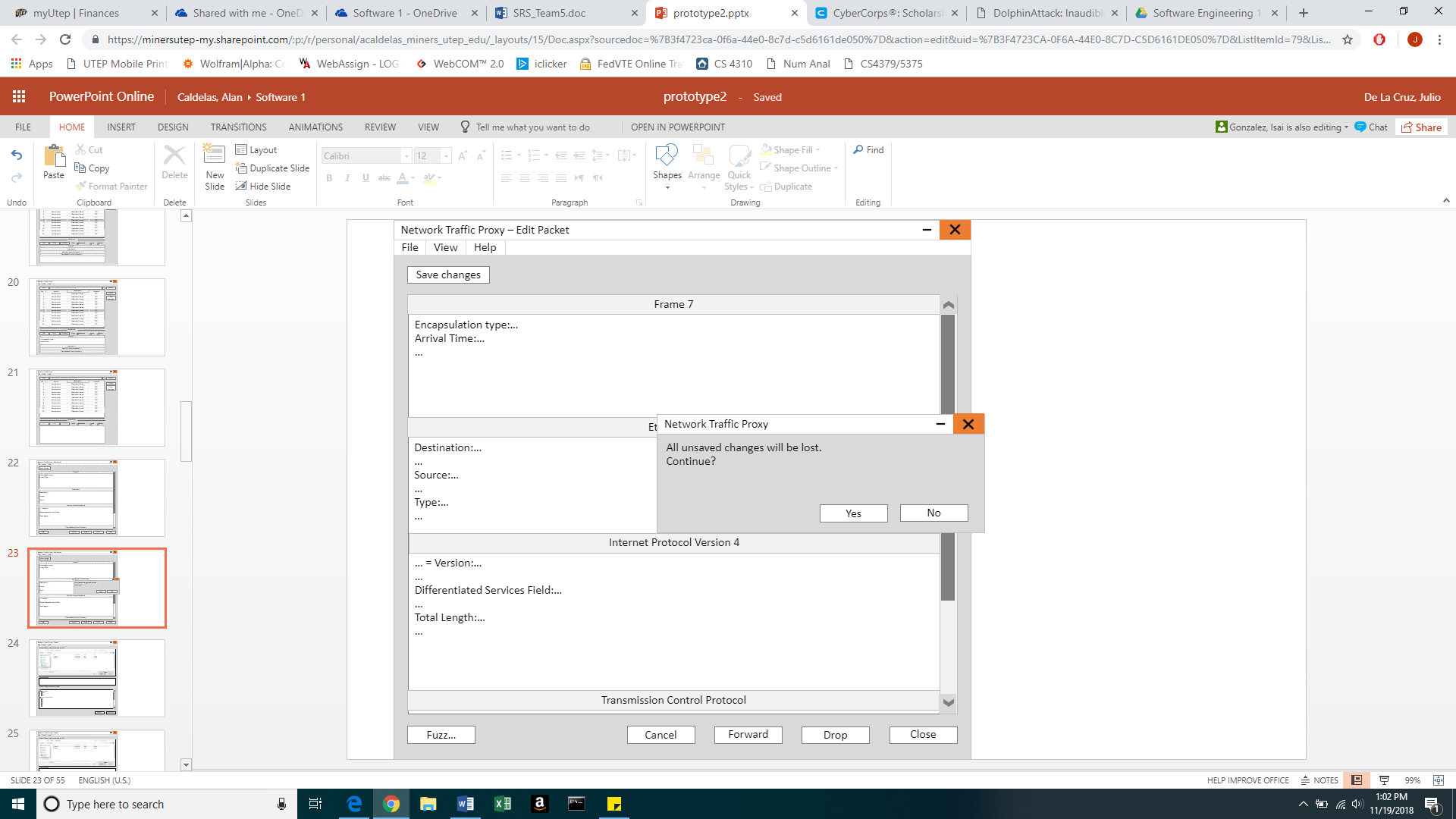


Figure 5: Unsaved Changes Confirmation Window

1. As shown in Figure 6.1, the “Fuzzer Interface 1” window shall include the following:
2. A panel labeled as “Select files to use as samples for AFL”, that will contain the directories and files of the local system.
3. A panel labeled as “Selected files:”, that will contain files that have been selected by the user.
4. A panel labeled as “Select fields you want to fuzz:”, that contains the names of the fields that are available to be fuzzed.
5. Check boxes next to the names of each of the available fields.
6. A button labeled as “Cancel”.
7. A button labeled as “Continue”.

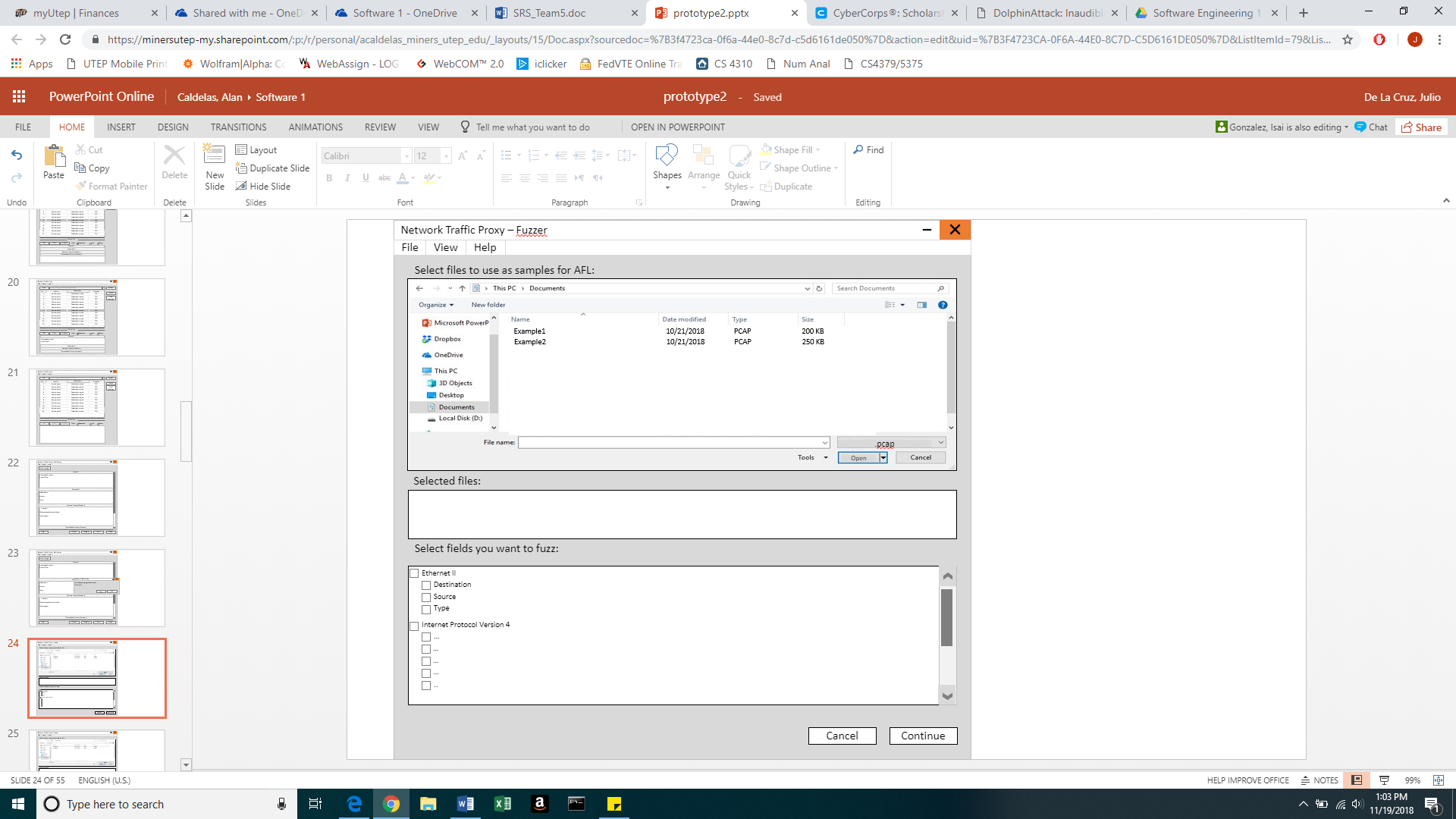


Figure 6.1: Fuzzer Interface 1

1. As shown in Figure 6.2, the “Fuzzer Interface 2” window shall include the following:
2. A panel labeled as “Select where to save fuzzed packets:”, that will contain the directories and files of the local system.
3. A text input box labeled as “File name:”, next to a dropdown menu with, at least, the option “.pcap”.
4. A button labeled as “Back”.
5. A button labeled as “Begin Fuzz”.

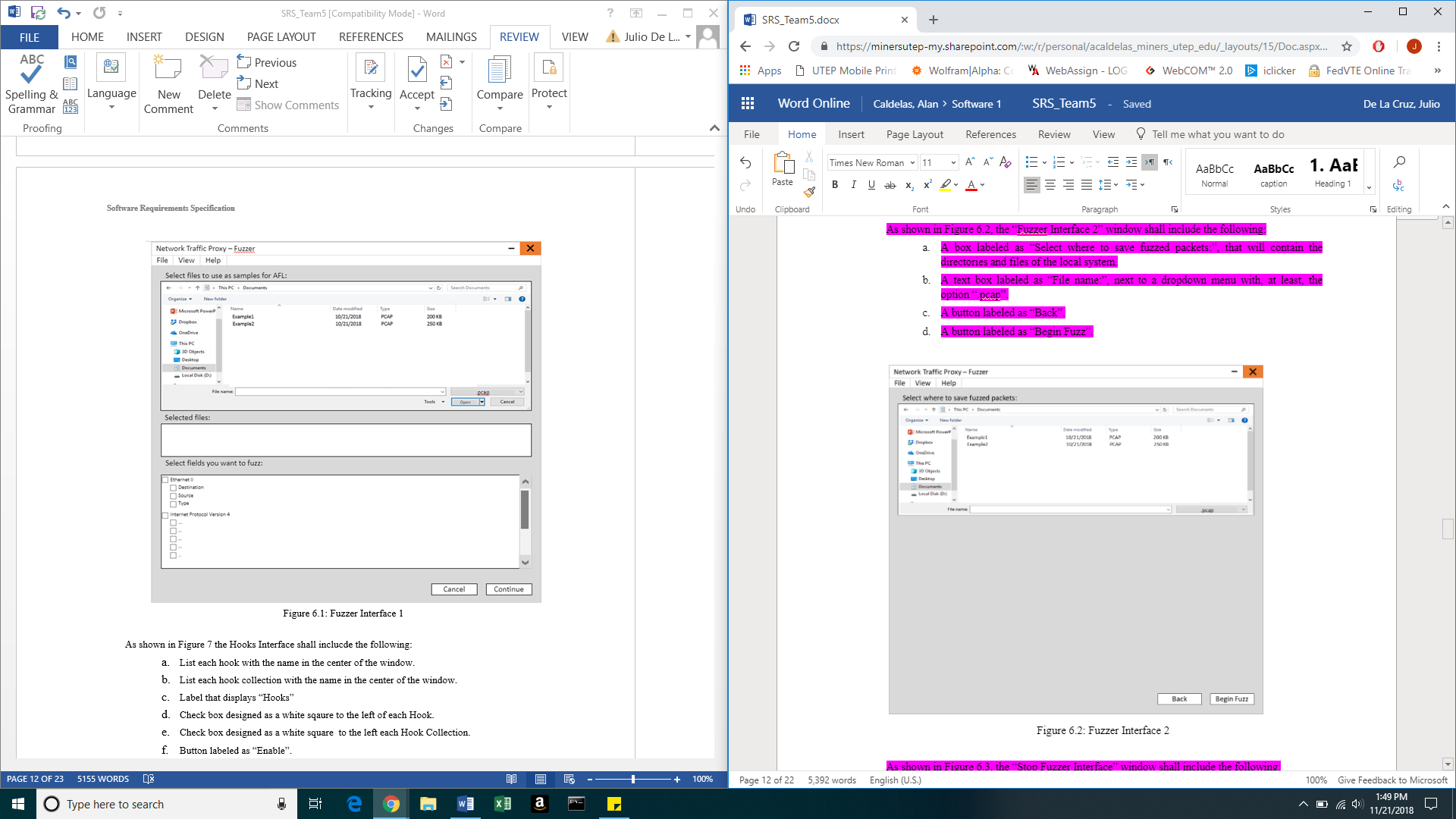


Figure 6.2: Fuzzer Interface 2

1. As shown in Figure 6.3, the “Stop Fuzzer Interface” window shall include the following:
   1. A message that states “Fuzzing in progress…”.
   2. A button labeled as “Stop”.

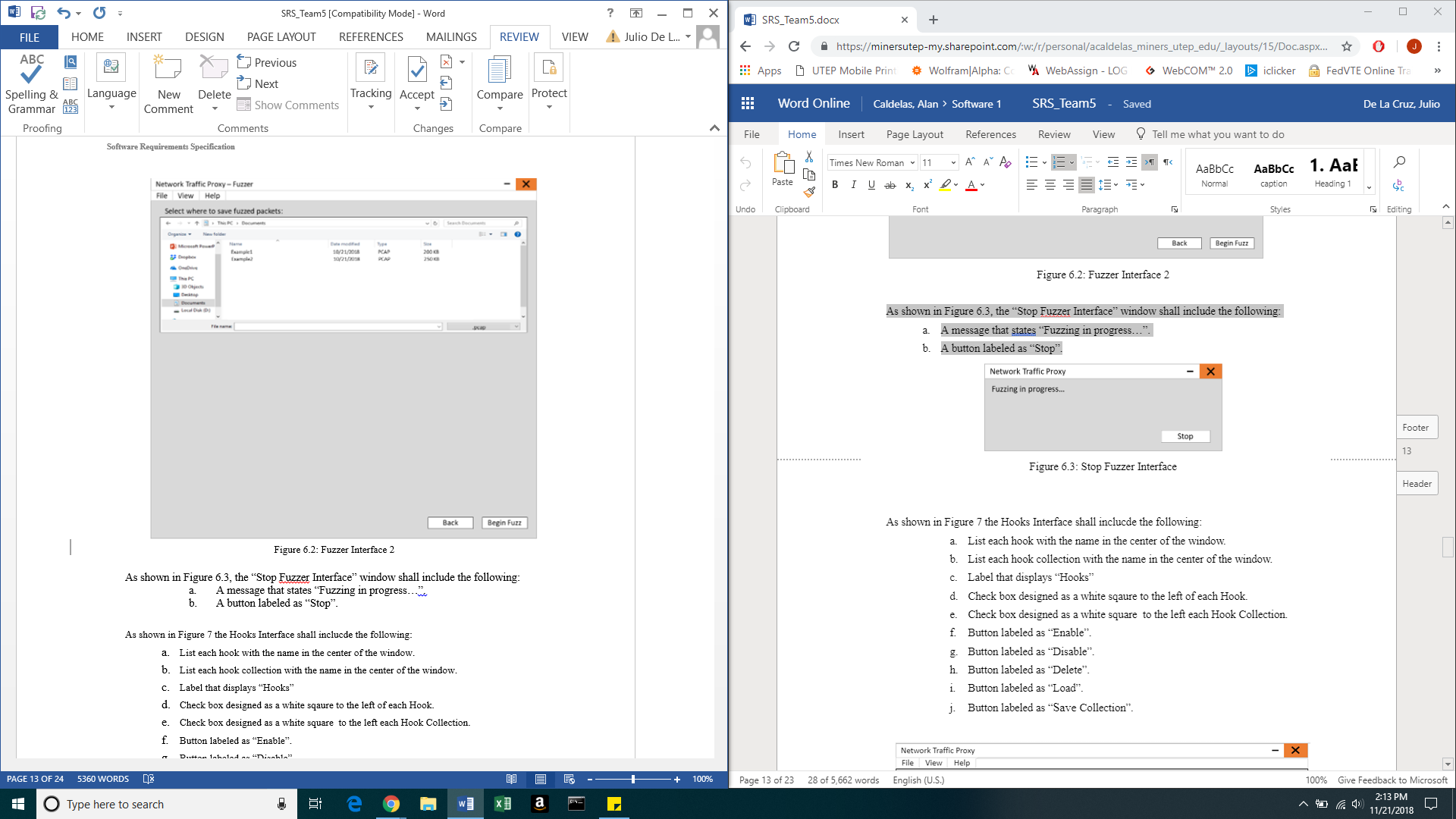


Figure 6.3: Stop Fuzzer Interface

1. As shown in Figure 7 the Hooks Interface shall inclucde the following:
   1. List each hook with the name in the center of the window.
   2. List each hook collection with the name in the center of the window.
   3. Label that displays “Hooks”
   4. Check box designed as a white sqaure to the left of each Hook.
   5. Check box designed as a white square to the left each Hook Collection.
   6. Button labeled as “Enable”.
   7. Button labeled as “Disable”.
   8. Button labeled as “Delete”.
   9. Button labeled as “Load”.
   10. Button labeled as “Save Collection”.

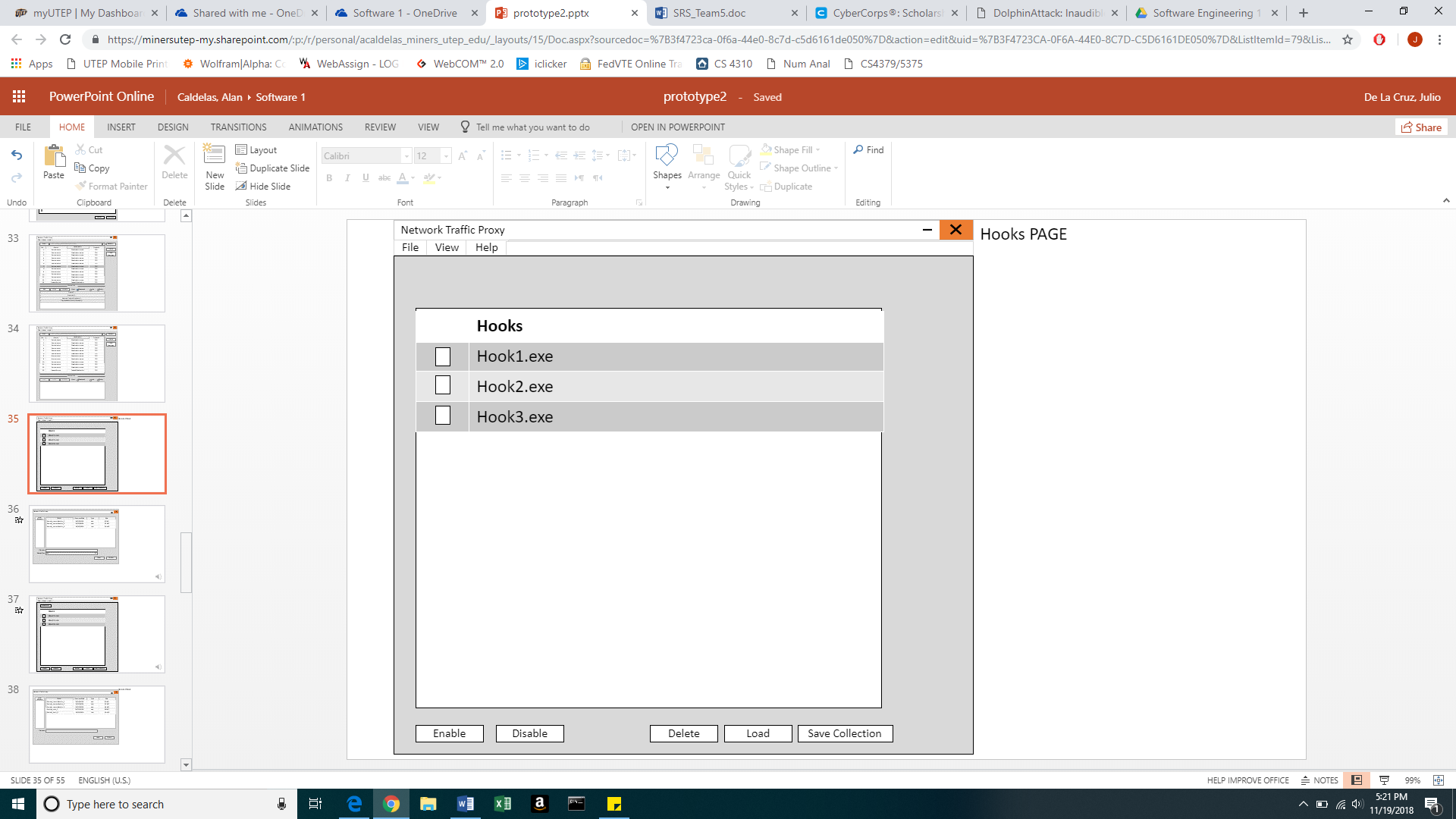


Figure 7: Hook Interface

1. As shown in Figure 8, the ‘Save Hook Collection’ interface shall include the following:
2. Interactive path file panel located on the top left hand side.
3. Interactive view panel with 4 columns inside that are selectable and displays each hook collection, the date when it was modified, the type of the file, and the memory of each file that have been previously saved before.
4. Label that displays “File Name”.
5. Text input box for entering a file name or entering a new file name.
6. Label that displays “Save as Type”.
7. Drop menu that offers a file type to be selected.
8. Button labeled as “Save”.
9. Button labeled as “Cancel”.

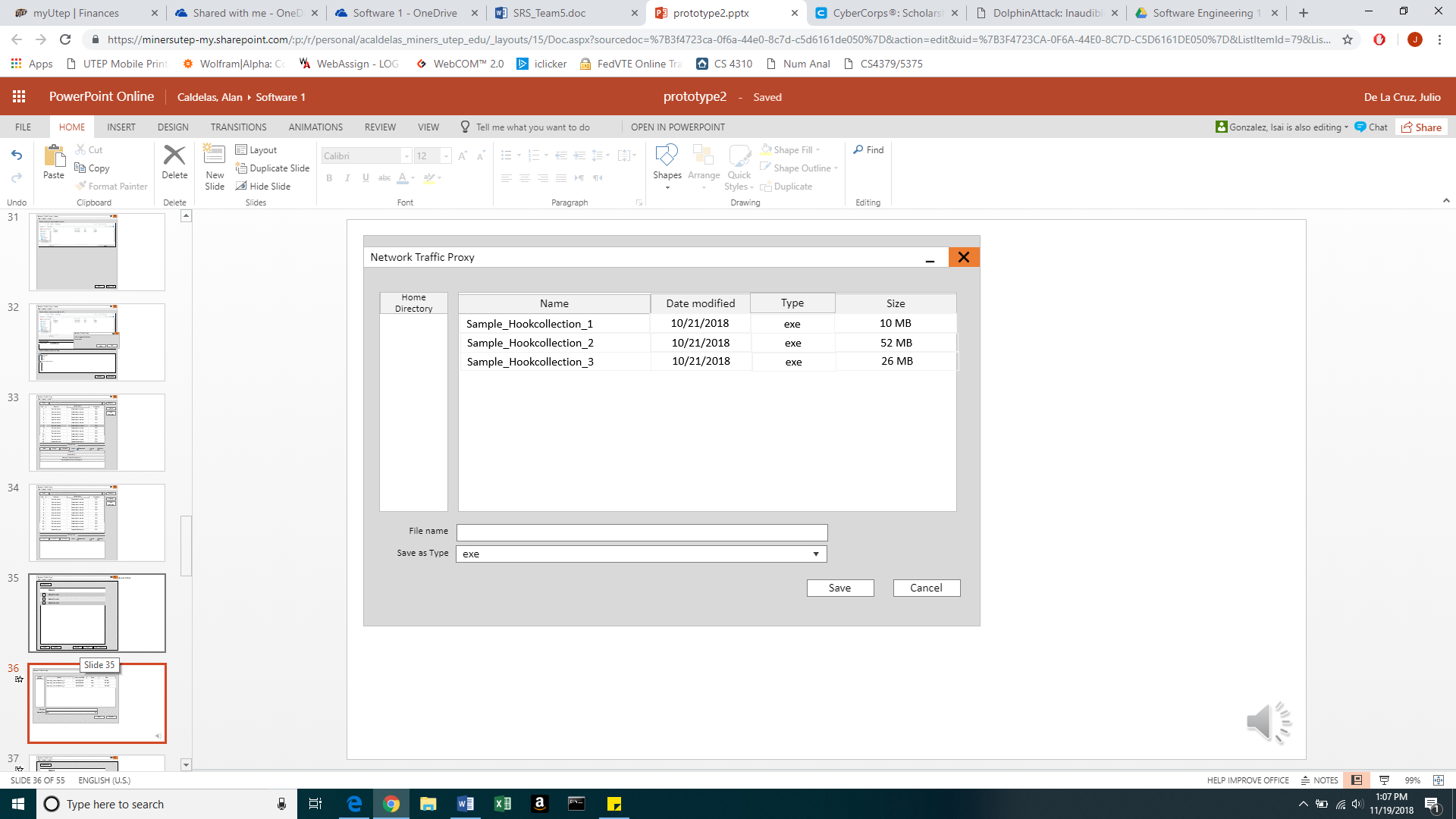


Figure 8: Hook Save Collection Interface

1. As shown in Figure 9, the Load Hook Collection’ interface shall include the following:
2. Interactive path file panel located on the top left hand side.
3. Interactive view panel with 4 columns inside that are selectable and displays each hook collection, the date when it was modified, the type of the file, and the memory of each file that have been previously saved before.
4. Label that displays “File Name”.
5. Text input box for entering a file name or entering a new file name.
6. Button labeled as “Load”.
7. Button labeled as “Cancel”.

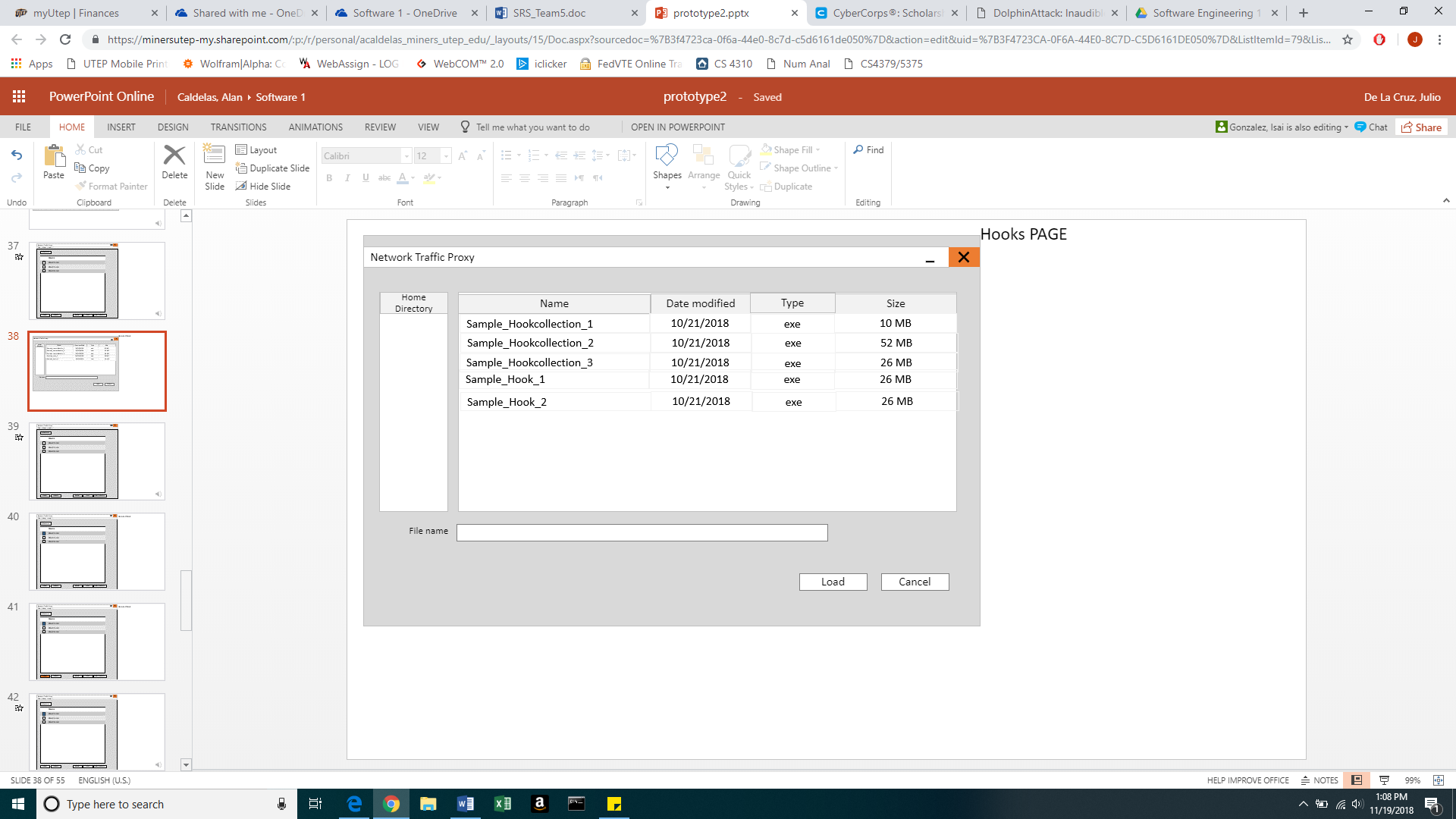


Figure 9: Hook Load Interface

### Hardware Interfaces

There are no Hardware Interfaces requirements for this system.

### Software Interfaces

* Scapy:
* Name: Scapy
* Mnemonic: Scapy
* Version Number: 2.4.0
* Source: <https://pypi.org/project/scapy/>
* API’s:

* IP-tables (Chains described and Tables (Filtering))
* Name: IP-tables
* Mnemonic:  IP-tabled
* Version Number: 1.08
* Rule used: iptables -A INPUT NFQUEUE –queue-num 0

* NFQUEUE
* Name: libnetfilter\_queue project
* Mnemonic: NFQUEUE
* Version Number: 2017-Nov-13: libnetfilter\_queue-1.0.3
* Source: <https://netfilter.org/projects/libnetfilter_queue/>
* Net Filter
* Name: netfilter.org project
* Mnemonic: netfilter
* Version Number: 2.6.14
* Source: <https://netfilter.org/projects/libnetfilter>

### Communications Interfaces

There are no Communications Interfaces requirements for this system.

## Behavioral Requirements

### Same Class of User

1. The system shall have one level of access privileges as shown in Table 4.

**Table 4: Access Privileges**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Access Hierarchy** | **Set proxy settings** | **Impleme-ntation of fuzzing** | **Modification of packets** | **Interception and filtering** | **Hook Impleme-ntation** | **PCAP**  **Functio-nalities** | **Drop/For-ward Packets** |
| Analyst | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

### Related Real-world Objects

Our class diagram for the NTPS, displayed in the Appendix, models the classes of the system and their relationships. This section will describe each class and have a set of requirements related to the class and objectives of the specific fields and field types that compose our objects.

Hook Collection class:

1. The system shall provide the functionality of displaying a list of hook collections for the analysts to select from.
2. The system shall allow the analysts to add hook(s) to a hook collection.
3. The system shall allow the analysts to remove hook(s) from a hook collection.
4. The system shall allow the analysts to modify the order of hook(s) to be executed in the hook collection.
5. The system shall allow the analysts to disable and enable a selected hook collection.
6. The system shall compose Hook Collections of at least one or more hooks.
7. The system shall link the lifetime of a Hook collection to the lifetime of its corresponding hooks.
8. The system shall apply a Hook Collection to zero or many packets.
9. The system shall represent the Hook Collection class with the attributes presented in Table 5..

**Table 5: Hook Collection Class**

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Values and Constraints** |
| Name | String | Name of hook collection |
| Description | String | Description of hook collection |
| hookList | List | Link to all hooks in list |

Field class:

1. The system shall display to the analysts the Field in the selected display format
2. The system shall provide the functionality for the analysts to edit the contents of the field.
3. The system shall represent the Field class with the attributes presented in Table 6.

**Table 6: Field Class**

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Values and Contraints** |
| Name | String | Name of field |
| Type | String | Type of field |
| displayFormat | String | Which display the field is being viewed |
| Mask | String | The type of mask the field has. |

Hook class:

1. The system shall provide the functionality to enable any hook.
2. The system shall provide the functionality to disable any hook.
3. The system shall provide the functionality to import hooks into the system.
4. The system shall compose hook collections of at least one hook.
5. The system shall represent the Hook class with the attributes shown in Table 7.

**Table 7: Hook Class**

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Values and Constraints** |
| name | String | Name of Hook |
| script | String | Path of import. |
| sequencingNumber | Integer | Priority level with 1 being the highest and the highest positive number found as the lowest priority. |
| enabled | Boolean | Boolean that indicates current state as active or not active. |

Layer Class:

1. The system shall represent the layers found inside of every Hook
2. The Layer class shall contain a number ranging from 1 to many of Field class instances as fields that represent the architecture of every Layer class instance.
3. The system shall represent the Layer class with the attributes shown in Table 8.

**Table 8: Layer Class**

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Values and Contraints** |
| name | string | Contains the name of the layer |

Packet Class:

1. The system shall provide the functionality of changing the display mode of a packet between hexadecimal, binary, and decoded-type.
2. The system shall provide the functionality of sending a packet through the network.
3. The system shall provide the functionality of dropping the packet from the system and the network.
4. The system shall provide the functionality of editing the dissected fields of a packet.
5. The system shall provide the functionality of fuzzing specific fields of a packet.
6. The system shall have the ability to send the packets that have been created by the fuzzer.
7. The system shall provide the functionality of saving the fuzzed packets as a PCAP file.
8. The system shall link the lifetime of a Packet to its corresponding Layers.
9. The system shall represent the Packet Class with the attributes shown in Table 9..

**Table 9: Packet Class**

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Values and Constraints** |
| viewFormat | Integer | Number that corresponds to display mode |

Proxy Rule Class:

1. The system shall provide the functionality to toggle on proxy rules.
2. The system shall provide the functionality to toggle off proxy rules.
3. The system shall represent the Proxy Rule Class with the attributes shown in Table 10..

**Table 10: Proxy Rule Class**

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Values and Constraints** |
| Output |  |  |
| numberOfPackets | Integer | Number of packets |

Capture Filter Class:

1. The system shall provide the functionality of enabling the selected Capture Filters before intercepting Packets.
2. The system shall provide the functionality to disable selected filters.
3. The system shall represent the Capture filter Class with the attributes shown in Table 11.

**Table 11: Capture Filter Class**

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Values and Constraints** |
| Protocol | string | The protocol of a packet to filter |
| Ips | Double | The ip address of a packet to filter |
| port | Double | The port of a packet to filter |

PCAP Class:

1. The system shall display the file names of the PCAP class instances.
2. The system shall allow the analyst to save packets from the queue to a new or an already existing PCAP file.
3. The system shall link the lifetime of a PCAP file to the packets it encompasses.
4. The system shall provide the functionality to delete PCAP files.
5. The system shall provide the functionality to load PCAP files to the Packet class.
6. The system shall compose PCAP files of at least one or many packets.
7. The system shall represent the PCAP Class with the attributes shown in Table 12.

**Table 12: PCAP Class**

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Values and Constraints** |
| name | string | The name of the PCAP file |

Queue Class:

1. The system shall provide the functionality to display the number of packets in the Queue.
2. The system shall place intercepted packets into the Queue.
3. The system shall provide the functionality to pop packets from the Queue.
4. The system shall provide the functionality to push packets into the Queue.
5. The system shall provide the functionality to peek packets in the Queue.
6. The system shall provide the functionality to select any packet within the Queue.
7. The system shall represent the Queue Class with the attributes shown in Table 13.

**Table 13: Queue Class**

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Values and Contraints** |
| Size | Integer | The size of the queue, constraint depends on the size the user specifies for the queue |
| numPackets | Integer | The number of packets in the queue. |

### Stimulus

<< Some systems (e.g., real-time systems) can be best organized by describing their functions in terms of stimuli. For example, the functions of an automatic aircraft landing system may be organized into sections that include the following loss of power, wind shear, and sudden change in roll. The state diagram, event diagram, or other dynamic model is included in this section. >>

Examples:

SRS103 If a workspace file is not specified in the Metafile table for the scene being opened, the system shall set the parameters for each window as specified in Table F-1 in Appendix F to their corresponding default values when the scene is opened.

SRS104 If a workspace file is loaded when a scene is first opened, the system shall set the default values to those specified in the file.  >>

### Related Features

<< A feature is an *externally* desired service provided by the system that may require a sequence of inputs to affect the desirable result. For example, in a telephone system, features include local call, call forwarding, and conference call. Each feature is generally described in a sequence of stimulus-response pairs (in such a case, Section 3.2.3 could be changed to a subsection of this section. The use cases form the outline of this section. >>

### Other

<< This section should present requirements that define responses to situations that do not fit in the other sections. Possible requirements could be overflow conditions, error handling, and recovery. >>

## Non-behavioral Requirements

### Performance Requirements

<< This subsection should specify both the static and the dynamic numerical requirements placed on the software or on human interaction with the software as a whole. Static numerical requirements may include:

1. the number of terminals to be supported,
2. the number of simultaneous users to be supported, and/or
3. the amount and type of information to be handled.

Dynamic numerical requirements may include, for example, the numbers of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions. All of these requirements should be stated in **measurable** terms, e.g.,

*“95% of the transactions shall be processed in less than 1 second.”*

rather than,

“*An operator shall not have to wait for the transaction to complete.” >>*

### Qualitative Requirements

<< There are a number of attributes of software that can serve as requirements. It is important that required attributes be specified so that their achievement can be objectively verified. >>

#### Availability

<< This should specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart. >>

#### Security

<< This should specify the factors that will protect the software from accidental or malicious access, use, modification, destruction, or disclosure. Specific requirements in this area could include the need to:

1. utilize certain cryptographical techniques,
2. keep specific log or history data sets,
3. assign certain functions to different modules,
4. restrict communications between some areas of the program, and/or
5. check data integrity for critical variables. >>

#### Maintainability

<< This should specify attributes of software that relate to the ease of maintenance of the software itself. There may be some requirement, for example, for certain modularity, interfaces, or complexity. Requirements should not be placed here just because they are thought to be good design practices. >>

#### Portability

<< This should specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems. This may include the following:

1. percentage of components with host-dependent code,
2. percentage of code that is host dependent,
3. use of a proven portable language,
4. use of a particular compiler or language subset, and
5. use of a particular operating system . >>

### Design and Implementation Constraints

<< This should specify design constraints that can be imposed by other standards or hardware limitations. This may include standards compliance, which state requirements derived from existing standards, or regulations (e.g., report format, data naming, accounting procedures). This section details constraints that directly affect design and implementation. For example:

*“The software shall be developed in Java 1.2”* or

*“The number of entries in the Log Table shall not exceed 500 entries.”* >>

## Other Requirements

### Database

<< This section should specify the logical requirements for any information that is to be placed into a database. This may include:

1. types of information used by various functions,
2. frequency of use,
3. accessing capabilities,
4. data entities and their relationships,
5. integrity constraints, and
6. data retention requirements. >>

### Operations

<< This section should specify the normal and special operations required by the user, such as:

1. the various modes of operation in the user organization, e.g., user-initiated operations,
2. periods of interactive operations and periods of unattended operations,
3. data processing support functions, and
4. backup and recovery operations. >>

### Site Adaptation

<< This section could be used to:

1. define the requirements for any data or initialization sequences that are specific to a given site, mission, or operational mode, for example, grid values, or safety limits, and
2. specify the site or mission-related features that should be modified to adapt the software to a particular installation. >>

# Appendix

