Network Traffic Proxy System

Software Requirements Specification

Version <1.0>

11/12/18

Document Control

Approval

The Guidance Team and the customer will approve this document.

Document Change Control

|  |  |
| --- | --- |
| Initial Release: | 1.0 |
| Current Release: | 1.0 |
| Indicator of Last Page in Document: | ∎ |
| Date of Last Review: | 11/9/18 |
| Date of Next Review: | 11/10/18 |
| Target Date for Next Update: | 11/10/18 |

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 Galindo | Added the information to the document control section and finished section 1.2. and 3.2.1 |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Document Control II](#_Toc529387839)

[Approval II](#_Toc529387840)

[Document Change Control II](#_Toc529387841)

[Distribution List II](#_Toc529387842)

[Change Summary II](#_Toc529387843)

[1. Introduction 4](#_Toc529387844)

[1.1. Purpose and Intended Audience 4](#_Toc529387845)

[1.2. Scope of Product 4](#_Toc529387846)

[1.3. Definitions, Acronyms, and Abbreviations 4](#_Toc529387847)

[1.3.1. Definitions 4](#_Toc529387848)

[1.3.2. Acronyms 4](#_Toc529387849)

[1.3.3. Abbreviations 4](#_Toc529387850)

[1.4. Overview 5](#_Toc529387851)

[1.5. References 5](#_Toc529387852)

[2. General Description 5](#_Toc529387853)

[2.1. Product Perspective 5](#_Toc529387854)

[2.2. Product Features 5](#_Toc529387855)

[2.3. User Characteristics 5](#_Toc529387856)

[2.4. General Constraints 5](#_Toc529387857)

[2.5. Assumptions and Dependencies 6](#_Toc529387858)

[3. Specific Requirements 6](#_Toc529387859)

[3.1. External Interface Requirements 6](#_Toc529387860)

[3.1.1. User Interfaces 6](#_Toc529387861)

[3.1.2. Hardware Interfaces 7](#_Toc529387862)

[3.1.3. Software Interfaces 7](#_Toc529387863)

[3.1.4. Communications Interfaces 7](#_Toc529387864)

[3.2. Behavioral Requirements 7](#_Toc529387865)

[3.2.1. Same Class of User 7](#_Toc529387866)

[3.2.2. Related Real-world Objects 7](#_Toc529387867)

[3.2.3. Stimulus 8](#_Toc529387868)

[3.2.4. Related Features 8](#_Toc529387869)

[3.2.5. Other 8](#_Toc529387870)

[3.3. Non-behavioral Requirements 8](#_Toc529387871)

[3.3.1. Performance Requirements 8](#_Toc529387872)

[3.3.2. Qualitative Requirements 8](#_Toc529387873)

[3.3.3. Design and Implementation Constraints 9](#_Toc529387874)

[3.4. Other Requirements 9](#_Toc529387875)

[3.4.1. Database 9](#_Toc529387876)

[3.4.2. Operations 10](#_Toc529387877)

[3.4.3. Site Adaptation 10](#_Toc529387878)

# Introduction

## Purpose and Intended Audience

<< State the purpose, intentions, and primary intended audience of the SRS. >>

The purpose of the Software Specifications Documents is to specify the product that our team will ultimately be building. This document is not meant to explain how it will be built, but what will be built. This is intended to explain to the client team, what to expect from the built system. What is described in this document will be what Team 5 will build and it is up to the client team to decide whether this is what they want or not. 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.

## Scope of Product

This project intends 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. In addition, our focus is 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 sufficient capacity to the system to be able to handle at any moment at least 100 packets whether the interception of packets is activated or not. 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].

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 randomization 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, 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 either interception is activated or not.

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

<< Provide a list of terms and definitions that may be required or be helpful to the intended audience. >>

Table 1: <Caption>

|  |  |
| --- | --- |
| **TERM** | **DEFINITION** |
| Active Scene | The scene that is currently displayed. |
| Aimpoint | The place on the target that the missile will try to fly towards and hit; the aimpoint is determined by calculating the centroid of a group of pixels that passed the target detection criteria or threshold. |
| Aspect Angle | The angle at which the vehicle is viewed relative to the front of the vehicle; for instance, 0º aspect is looking straight down the front of the vehicle, 90º is looking at the left side of the vehicle, 180º is looking at the back of the vehicle, and 270º is looking at the right side of the vehicle. |
| Terminal | Computer where we the Network Traffic Proxy System is run. |

### Acronyms

<< Provide a list of used acronyms and their associated definitions. >>

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

Table 2: <Caption>

|  |  |
| --- | --- |
| **TERM** | **DEFINITION** |
| NTPS | Network Traffic Proxy System |
| SRS | Software Requirements Specification |

### Abbreviations

<< Provide a list of used abbreviations and their associated definitions. >>

Table 3: <Caption>

|  |  |
| --- | --- |
| **TERM** | **DEFINITION** |
| e.g. | For example |

## Overview

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).

# General Description

## Product Perspective

<< Provide a description of this product, where it fits into the Company’s overall structure, and what functionality is provided by this product. This section places the product to be developed in perspective with other related products. If the product is independent and self-contained, this should be stated; if it is a component of a larger system, the relationship to the larger system should be stated and the interfaces defined. >>

## Product Features

<< The main features of the software should be summarized and organized in a way that makes it understandable to the customer or the reader. Use graphical methods to show the different functions and their relationships. Give an overview of the essential features that are important to this product. Use case diagram (Level 2) and descriptions should be included in this section. >>

## User Characteristics

<< This section should identify each type of user of the system (by function, location, type of device), the number in each group, and the nature of their use of the system. State any user characteristics that may be influential or significant in the structure of the product. Include characteristics such as educational level, experience and technical expertise. >>

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 and in Security analysis of Computer Science. The use also should have some technical use of tools used in Networking such as WireShark, netstat, 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 (RDD reference). 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.

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 Scapy for packet manipulation, Wireshark for network analysis, using Hooks to modify packets. 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.

## General Constraints

<< State any known constraints. Explain factors that constrain the options of the development team including organizational factors, hardware limitations or safety and security considerations. These may include constraints such as:

*“The system will be accessible in the field.”* or

*“The system will not be accessible to unauthorized users.”* Note: the term “unauthorized users” should be defined in Section 1.3.1 of the SRS)*.*

These constraints are more general and may be elaborated other places in the SRS. >>

Constraints of the development team include time, knowledge, …

Time: The development team is constrained to only about two semesters of the school year. This team is also restricted by other factors that can affect how the development team works together, such as outside work, other course work that requires a large amount of time and family/medical related issues that may cause a member to be gone for a portion of the project duration.

Knowledge: The development team is comprised of senior undergraduates with various experience in different subjects. The project requires a heavy amount of knowledge of Networking based systems.

Hardware limitations:

## Assumptions and Dependencies

<< This subsection should list each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software. They are assumptions and dependencies that if changed, will affect the requirements of the SRS. For example, an assumption might be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS will then have to change accordingly. >>

Examples:

The development team has made the following assumptions [AC3]:

* The source code will be provided to ARL/SLAD.
* ARL/SLAD will provide the correct MATLAB signatures for the following:
  + *aimpoint and detection* module,
  + *obscurants* module, and
  + *change resolution* module.
* ARL/SLAD will provide unclassified mockups of the modules listed above for the purpose of testing and integration. >>

# Specific Requirements

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

## External Interface Requirements

<< This section contains the specification of requirements for interfaces among different components and their external capabilities, including all its users, both human and other systems. The characteristics of interfaces to systems under development, or future systems, should also be included. Any inter-dependencies or constraints associated with interfaces should also be identified (e.g., communication protocols, special devices, standards, fixed formats). Each interface may represent a bi-directional flow of orientation. Use a graphic representation of the interfaces when appropriate for the sake of clarity. >>

### User Interfaces

<< Characteristics of each interface between the software product and its users. This should specify the following:

1. The *logical characteristics of each interface between the software product and its users.* This includes those configuration characteristics (e.g. required screen formats, page or window layouts, content of any reports or menus, or availability of programmable function keys) necessary to accomplish the software requirements.
2. *All the aspects of optimizing the interface with the person who must use the system.* This may simply comprise a list of do’s and don’t on how the system will appear to the user. One example may be a requirement for the option of long or short error messages. Like all others, these requirements should be verifiable, for example, “A clerk typist grade 4 shall be able to do function *X* in *z* minutes after the introductory training session” rather than “A typist shall be able to do function *X*.” >>

Example:

SRS3 The system shall gray-out and disable any visible GUI elements that are not applicable to the context of the functionality being performed. See Appendix F for settings.

SRS5 Hover text shall be displayed in sentence case.  >>

### Hardware Interfaces

<< Characteristics of each interface between the software product and the hardware components of the system, e.g., configuration characteristics (ports, instruction sets, devices to be supported). >>

### Software Interfaces

<< This should specify the use of other required software products (for example, a data management system, an operating system, or a mathematical package), and interfaces with other application systems (for example, the linkage between an accounts receivable system and a general ledger system). For each required software product, the following should be provided:

1. name,
2. mnemonic,
3. specification number,
4. version number, and
5. source. >>

### Communications Interfaces

<< This should specify the various interfaces to communications such as local network protocols, etc. >>

## Behavioral Requirements

### Same Class of User

The system shall have one, universal level of access privileges.

Analyst access privilege level shall be unrestricted and should provide access to all functionalities within the system.

Table 1: Access Privileges

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Access Hierarchy** | **Set proxy settings** | **Impleme-ntation of fuzzing** | **Modific-ation 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 Network Traffic Proxy System (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 system.

Hook class:

The Hook class shall contain two methods, X and Y to support disabling and enabling operations for every instance of the Hook class.

The Hook class shall contain a field that enumerates the priority number of every Hook Class instance within a collection, with the highest priority level being equal to 1 and the lowest being equal to the highest number greater than 1 found in the collection.

The Hook class shall contain a field X that stores the name of every Hook class instance.

The Hook class shall contain a method X to perform the loading hook operation from the terminal where the system (NTPS) is running.

Layer Class:

The Layer class shall contain a name field X that shall contain the name-description for every Layer class instance.

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.

### 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. >>

∎