Vision Document 2.0

for

SnIPS Implementation and GUI

Tsung-Hsi Wu
CIS 895 MSE Project
Department of Computing and Information Science
Kansas State University
1. Introduction

This section explains the motivation, overview of SnIPS, key terms and definitions, and references used for the production of this vision document.

1.1. Motivation

The output of SnIPS is generated by applying SnIPS reasoning engine which gets alerts from Snort intrusion detection system. However, in order to fulfill the curiosity of system administrator in terms of what causes the alerts in Snort, the functionality of finding what payload triggers the snort rule which leads to generating alerts is needed. Moreover, the current version of SnIPS operates under Linux command mode which implies that a GUI interface for SnIPS is required in order to operate SnIPS handily.

1.2. SnIPS

SnIPS works by mapping a Snort alert into a logic predicate describing the condition a user really cares about (e.g. machine compromised), along with a tag indicating the strength of the belief (possible, likely, or certain). The tagged conditions are reasoned about together and beliefs with strong corroborative evidential support are distinguished from those with only mediocre evidence, yielding high-confidence attack traces from the Snort alerts. It can handle Snort alerts coming from multiple sources to detect multi-stage attacks in a network.

1.3. Terms and Definitions

Reasoning Engine – An engine consists of observation correspondence and internal models. This engine is implemented in Prolog and it is able to give answers to simple Prolog queries.

Snort - A widely used network-based intrusion detection system (IDS). It works by comparing a network packet with a set of pre-defined signatures (Snort rules) which specify certain patterns often associated with malicious activities.

System Administrator - A person employed to maintain and operate a computer system and/or network

Graphical User Interface (GUI) - A type of user interface item that allows people to interact with programs in more ways than typing.

1.4. References


2. Project Overview

The project overview provides the structure of SnIPS, goals, and purpose for extending the functionality of SnIPS.

2.1. Introduction

SnIPS is a reasoning system for combining low-level IDS alerts to produce high level attack traces. The current implementation is a command line utility.

![SnIPS Architecture Diagram]

Figure 1: SnIPS Architecture

Figure 1 depicts the architecture of SnIPS which is a reasoning system. According to *An empirical approach to modeling uncertainty in intrusion analysis* paper, the top two modules, observation correspondence and internal model are specified in Datalog. The raw observations are pre-processed and the distilled results are converted to Datalog tuples as input to the reasoning system. The reasoning engine is itself implemented in Prolog.

2.2. Goal

There are two goals for this project. The first goal is to find the correspond payload of the alert that triggers the snort rules in order to capture the malicious behavior occurred from SnIPS results. The second goal is to generate a GUI for SnIPS and substitute for command-based operation. Figure 2 shows which part of SnIPS architecture will be extend to GUI based approach in this project.
2.2. Purpose

The purpose of this project is to increase the functionality of SnIPS and create a friendly user interface in terms of usability for system administrator. The project will provide thorough payload in respect of snort rules that correspond with the output from the reasoning engine and reduce the time of finding the actual malicious and compromised attacks happened throughout the network.

3. Project Requirements

This section points out system administrator and user in demand requirements in respect of SnIPS GUI. The level of demand of each requirement is categorized as Critical, Non-Critical, or Future.

The first part of project requirements describes the requirements of overall structure regarding SnIPS function and GUI. The second part indicates the requirements of SnIPS GUI based on Use Case diagram.

3.1. SnIPS function and GUI framework

Description

The most significant notion of tackling network security issues is to solve the problem after network attack. Although network security theories are reliable, issues will arise in practice which leads to requiring an extendible project.
SnIPS with graphical user interface (GUI) can be seemed as Server and Client architecture. SnIPS provides all the functions with regard to initiating snort, fetching alert, and execution of pre-processing and reasoning. These first hand functions are done by Sakthiyuvanraj Sakthivelmurugan. SnIPS GUI is simply a friendly user interface for system administrator and user to operate SnIPS functions instantly by clicking buttons instead of typing commands.

**SR1. Framework Requirements**

SR1.1. (Critical) SnIPS under Linux system and graphical user interface (GUI) must be extendible. This implies adding new functions and updating or deleting existing functions.

SR1.2. (Critical) All functions in SnIPS GUI corresponds with functions in SnIPS under Linux system.

SR1.3. (Critical) SnIPS GUI should fulfill the functions provided in SnIPS under Linux system excluding the functions from reasoning engine in SnIPS.

3.2. Use Case – SnIPS GUI Component

![SnIPS GUI Component Use Case](image)

**Description**

Figure 3 is SnIPS GUI component use case. It depicts the capability of control panel GUI in user’s point of view.
**SR2. Use Case 3.2.1. – Operate Snort**

SR2.1. (Critical) User can operate control panel GUI to start Snort intrusion detection system to generate alerts. This is reported as an error.

SR2.2. (Critical) User can operate control panel GUI to stop Snort intrusion detection system. This is reported as an error.

SR2.3. (Critical) User can operate control panel GUI to configure Snort intrusion detection system such as using different ports for SR.2.1. This is reported as an error.

SR2.4. (Critical) User can operate control panel GUI to update Snort rules. SnIPS GUI can update latest Snort rules by providing an Import Snort Rules directory menu to import latest Snort rules downloaded from Snort website. This is reported as a warning.

SR2.5. (Critical) The status of SR.2.1 to SR.2.4 will be shown on SnIPS GUI. This is reported as an error.

**SR3. Use Case 3.2.2. – Operate SnIPS**

SR3.1. (Critical) User can operate control panel GUI to run pre-processing function for SnIPS based on SR.3.4 time frame. This is reported as an error.

SR3.2. (Critical) User can operate control panel GUI to run summary function for SnIPS based on SR.3.4 time frame. This is reported as an error.

SR3.3. (Critical) User can operate control panel GUI to run trace function for SnIPS based on SR.3.4 time frame. This is reported as an error.

SR3.4. (Critical) User can operate control panel GUI to set the time frame for SR.3.1 ~ 3.3. This is reported as an error.

SR3.5. (Critical) Control Panel GUI should provide a button that runs SR.3.1 ~ 3.3 after the time frame from SR.3.4 is set and display a pop-up message showing that SR.3.6 and 3.7 are available. This is reported as an error.

SR3.6. (Critical) SnIPS GUI is able to fetch alerts from MySQL Database or raw alert files. The output of the alert is shown in web-based by the browser. This is reported as an error.

SR3.7. (Critical) All alerts need to be fetched based on time frame. For instance, 1:00pm ~ 3:00pm, Jan-01-2010 ~ Jan-02-2010 and etc. This is reported as a warning.

SR3.8. (Critical) SnIPS GUI will provide a button linking to text-based output website for tracing alerts back to snort rule or payload. This is reported as an error.

SR3.9. (Non-Critical) SnIPS GUI will provide a button linking to graphic-based output website for tracing alerts back according to time line. This is reported as an error.

SR3.10. (Critical) The status of SR.3.1 to SR.3.4 will be shown on control panel GUI. This is reported as an error.
3.3. Use Case – Trace Output Webpage

Description

The trace output webpage not only display the result of proof strengthening but also include links to trace back the original alerts which consist of payload that triggers snort rules. Moreover, trace output webpage provides a link that describes the snort rule.

Figure 4: Trace Output Website Use Case

SR4. Use Case 3.3.1. – Show Proof Strengthening Result

SR4.1. (Critical) Trace output webpage provides the result of proof strengthening. This is reported as an error.

SR4.2. (Critical) The result of proof strengthening on trace output webpage should be aligned properly. This is reported as a warning.

SR5. Use Case 3.3.2. – Trace Back Alerts

SR5.1. (Critical) Trace output webpage provides links to display primitively summarized alerts for each result. This helps the system administrator perceive what kind of attack the system encounters. This is reported as an error.

SR6. Use Case 3.3.3. – Trace Back Alert Payload

SR6.1. (Critical) Trace output webpage provides links to display the payload of the alerts that triggers a particular snort rule. This is reported as an error.
SR7. Use Case 3.3.4. – Show Snort Rule and Description

SR7.1. (Critical) Trace output webpage provides links to display the snort rules that have been triggered by the alerts. This is reported as an error.

SR7.2. (Critical) Trace output webpage provides links to display the descriptions for the result in SR9.1. This is reported as an error.

4. Assumptions

a) Users are aware of the procedure of using SnIPS GUI.
b) JVM 1.3.1 or later version must be installed on the machine.

5. Constraints

The GUI control panel developed in this project is restricted to Linux 2.6 or above operating system. This is because the server that contains reasoning engine requires SSH connect remotely. The client's Linux-Based operating system needs set up the connectionless SSH configuration at the first place.

6. Environment

a) JDK 1.5 – available at http://sun.java.com
c) MYSQL 4.0.13-nt – available at http://www.mysql.com/
d) Web browser support XMLHttpRequest. Eg: Firefox, Google Chrome.
e) Linux 2.6 or above operating system.