Project Evaluation

for

SnIPS Implementation and GUI

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

This document describes the insight of this project from design to implementation. The document starts with problem encountered, source lines of code, project duration, and then lessons learnt from this project.

2. Problem Encountered

2.1 Snort

This project is based on Snort intrusion detection system. Understanding the concept and how to run Snort is the first challenge that I had at the beginning stage. The documents for Snort are over hundreds of pages. I glanced over most of the documents and understand the framework of Snort IDS such as payloads and Snort rules.

2.2. SnIPS

Understanding the design and operation of SnIPS is the biggest challenge that I encountered during the first phase of MSE project. In order to fully understand the design of SnIPS, I need to fully understand the paper An empirical approach to modeling uncertainty in intrusion analysis written by Xinming Ou, S. Raj Rajagopalan, and Sakthiyuvaraja Sakthivelmurugan.

Operating SnIPS is another major issue. SnIPS is composed of XSB engine using python and shell script which I am not familiar with initially. I spent 2 weeks on shell script and operating SnIPS. In addition, MySQL database which stores Snort alerts is another challenge for me to get familiar with.

2.3. Parser

Due to the need of converting *.txt proof strengthened result to web-based with hyper links to trace back alerts, I need to build a parser to fulfill this need. I have written two kinds of parser. One is to create PHP web-based result which provides checkbox to extend alert description. The other one is to create JavaScript web-based result. This is the first time to experience how to write a parser and it took me a month to complete this parser.

2.4. Webpage Result

Two kinds of webpage result were developed during phase one and two. As mentioned above, one is PHP and the other is JavaScript version. The webpage result were build using PHP at the beginning due to language familiarity. However, during phase two, Dr. Simon and I decided to lessen the installation on our CIS server. This leads to more challenge because JavaScript is client-side programming language which is incapable of local file accessing. The solution for this was using Ajax but unfortunately Ajax was not capable of database connection. In a word, the webpage result switched back to PHP web-based at the end.
2.4. GUI Control Panel

Although I had experience in JavaSwing back in undergrad, I have not written
JavaSwing program since that. It took me around a month to understand needed
Swing Classes and corresponding ActionListeners.

3. Source Lines of Code

This project uses 3 kinds of programming languages: Java, JavaScript with Ajax, and PHP.
Table 1 depicts SLOC in detail of each language by using cloc-1.09 from cloc.sourceforge.net
website.

<table>
<thead>
<tr>
<th>Program Language</th>
<th>File</th>
<th>Blank</th>
<th>Comment</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>3</td>
<td>429</td>
<td>333</td>
<td>1765</td>
</tr>
<tr>
<td>PHP</td>
<td>5</td>
<td>177</td>
<td>54</td>
<td>455</td>
</tr>
<tr>
<td>JavaScript + Ajax</td>
<td>2</td>
<td>39</td>
<td>0</td>
<td>175</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>10</strong></td>
<td><strong>645</strong></td>
<td><strong>387</strong></td>
<td><strong>2395</strong></td>
</tr>
</tbody>
</table>

Table 1 Source Lines of Code

The total source lines of code estimated in Project Plan 2.0 was 2700 but the actual source
lines of code is 2395. This is because the webpage result switched back to original PHP web-
based which is server-side language and has more straightforward functions in terms of local
file accessing and database connection.

Based on table 1, it is obvious that Java language plays an important role in this project due to
Parser and GUI implementation. Although the webpage result is switched back to PHP,
written codes in JavaScript with Ajax are still taken into account.

4. Project Duration

Table 2 and 3 shows the estimated and actual start and end date of each phase.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Start Date</th>
<th>End Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase I</td>
<td>Sep 1st 2009</td>
<td>Dec. 3rd 2009</td>
</tr>
<tr>
<td>Phase II</td>
<td>Dec 30th 2009</td>
<td>Mar 30th 2010</td>
</tr>
<tr>
<td>Phase III</td>
<td>Apr 1st 2010</td>
<td>May 5th 2010</td>
</tr>
</tbody>
</table>

Table 2 Actual Start and End Date

<table>
<thead>
<tr>
<th>Phase</th>
<th>Start Date</th>
<th>End Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase I</td>
<td>Sep 1st 2009</td>
<td>Dec. 4th 2009</td>
</tr>
<tr>
<td>Phase II</td>
<td>Dec 21st 2009</td>
<td>Mar 16th 2010</td>
</tr>
<tr>
<td>Phase III</td>
<td>Mar 17th 2010</td>
<td>May 11th 2010</td>
</tr>
</tbody>
</table>

Table 3 Estimated Start and End Date
<table>
<thead>
<tr>
<th>Phase</th>
<th>Programming (min)</th>
<th>Documentation (min)</th>
<th>Meeting (min)</th>
<th>Reading (min)</th>
<th>Web (min)</th>
<th>Presentation (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase I</td>
<td>2005</td>
<td>2240</td>
<td>480</td>
<td>295</td>
<td>405</td>
<td>120</td>
</tr>
<tr>
<td>Phase II</td>
<td>3395</td>
<td>4925</td>
<td>375</td>
<td>0</td>
<td>70</td>
<td>195</td>
</tr>
<tr>
<td>Phase III</td>
<td>2110</td>
<td>2455</td>
<td>180</td>
<td>0</td>
<td>50</td>
<td>120</td>
</tr>
<tr>
<td>Total (hr)</td>
<td>7510 / 60 = 125.17</td>
<td>9620 / 60 = 160.33</td>
<td>1035 / 60 = 17.25</td>
<td>295 / 60 = 4.91</td>
<td>525 / 60 = 8.75</td>
<td>435 / 60 = 7.25</td>
</tr>
<tr>
<td>Total (hr)</td>
<td>323.66 hrs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4 Summation of Overall Project Duration

![Pie Chart](image1.png)

Figure 1 Overall Project Duration Percentage

![Pie Chart](image2.png)

Figure 2 Phase 1 Duration Percentage
5. Lesson Learnt

5.1 Software Management and Software Engineering Design

MSE Project is a great approach of experiencing the interrelationship with software management and software engineering design. Software management separates software timeline into phases and lists out the required tasks within every phase. On the other hand, software engineering design is the supplement to software
management. For example, during the first phase of software life cycle, Inception Phase, which captures the requirements from the users can use Use Case Diagram from software engineering design to capture user’s requirements in respect of user roles, scenarios, etc. During the second phase of life cycle, Elaboration Phase, Component, Deployment, Class, and Sequence Diagram can be very handy to assist the development of Architecture Design.

Software engineering design also provides patterns for mid-level designs. These patterns facilitate mid-level designs which lead to a great start. I adopted Command Pattern Structure for the GUI Control Panel. The class diagram of command pattern structure helps me to extend the design and consider the design in a more flexible point of view. Software engineering design not only decomposes design but also combines objects if it is needed. This happens when the process of decomposition becomes too deep and in turn combines similar parts for reducing the number of low-level modules or elements. Breaking modules into too many pieces may accidentally increase software complexity. I learned how to combine similar ActionListener classes into one class and simplify architecture design.

5.2 Architecture Design Flexibility

Another important lesson learnt in the production of MSE project is to have a flexible design. Ideally, 80 percent of requirements should be captured at the end of Elaboration Phase. However, users or stakeholders never stop asking for more and new features. It sure is a pain if the architecture of the software is not flexible enough to face the needs. I didn’t design the control panel flexible enough at the beginning stage. This leads me to spending lots of time modifying architecture in both software and documents. I decomposed modules into many pieces so that new module and replace old ones if it is required.

5.3 Software Prototypes

At the beginning of this project, the GUI control panel was simply to replace Linux commands in respect of operating SnIPS with graphical user interfaces such as buttons and textbox. During the process of phase II when I started implementing the functions of control panel, buttons are added to control panel. Then, more features are added such as list bar for selecting time frame and table for displaying the status of each button operation.

Control panel was considered to display web-based results for tracing back alerts to payloads. However, this will extend control panel to support browser functionality. Although everything will be displayed on control panel, due to time and exist PHP program that displays the web result with links to trace back Snort rules, descriptions, and payloads in the database, control panel is decided to focus on operating Snort and SnIPS and showing the status of Snort and SnIPS operations.

At the end of phase II, prototype 2.0 was developed with functions that can operate Snort and SnIPS. In other words, the basic goal of replacing Linux command with graphical user interface is fulfilled. However, new issues and requirements arise with the ability of control panel. For instance, prototype 2.0 is design under “synchronous” approach; that is, clicking a button will freeze the control panel till it is finish. The control panel can not operate Snort and SnIPS at the same time which definitely not expected to have. The final software production of this project resolves this issue by redesigning the control panel under “asynchronous” approach. This is done by spawning threads to in charge of each button operation.