CS424: Network Security

**A Novel Network Security Protection System**

Zaoxing Liu Chang Liu

zaoxing@jhu.edu chang.liu@jhu.edu

Department of Computer Science

Johns Hopkins University

04/20/2012

Abstract

As a de facto standard for IDS/IPS, Snort platform is combing the benefits of signature, protocol, and anomaly based detection. Since Snort platform includes various security technologies of IPS and IDS, it is difficult for most of security software users to configure the snort and make the most of it. In our new system, we modified and recompiled the original Snort platform by identifying the “active” rules in the rule set. Then, our modified algorithm of choosing and executing rules is optimized. Through obtaining the necessary information from the lower platform, our UI friendly web application can implement the security network monitoring, alerting, analysis, and feedback.

In the final test, our system’s rule execution performance is more efficient by 6%~21%. By using our system, users can get excellent experience in protecting the network security.

I. INTRODUCTION

***A. Background***

Any attempt tried to gain unauthorized access to a computer or disrupt the availability of a service/resource is termed as an intrusion. Intrusion Detection System[1][2] (IDS) refers to a software or a system built to detect intrusions and Intrusion Prevention System (IPS) is mainly focus on monitoring network or system activities for malicious behaviors. In general, detection mechanism used by IDS can be classified into two major categories.

* 1) *Signature based detection*: Models built from well-known attack types, i.e., from already known attack patterns.
* 2) *Anomaly based detection*: Modeled using normal traffic and deviation from this profile is considered anomalous.

As a de facto standard for IDS/IPS[2], Snort platform is combing the benefits of signature, protocol, and anomaly based detection. Snort’s open-source feature provides our developers an opportunity that we can build our new system based on the Snort. Since Snort platform includes various security technologies of IPS and IDS, it is difficult for most of security software users to configure the snort and make the most of it. Totally, the snort platform can be configured to provide service in three core modes:

• *Sniffer mode*, which simply reads the packets off of the network and displays them for users in a continuous stream on the console.

• *Packet Logger mode*, which logs the packets to disk.

• *Network Intrusion Detection System (NIDS) mode*, the most complex and conﬁgurable conﬁguration, which allows Snort to analyze network trafﬁc for matches against user-deﬁned rule set and performs several actions based upon what it sees.

***B. Prior Work***

The snort[4] project started in 1998 and it has become one the most respectful open source soft wares worldwide through the development of more than 10 years. Some snort’s protocols and mechanisms, such as simple packet management, simple TCP/UDP rules matching, are out of date. Although snort has a simple GUI interface for users, this interface is not user-friendly and cannot directly represent the data. In 2004, the Basic Analysis and Security Engine (BASE) released its candidate for 1.0, which includes a fully functional user authentication system and a setup program. This release also includes support for the new Flow-Portscan preprocessor. Thus

***C. Our work***

Our goal is to modify the original snort platform and build a new integrated system, which combines the snort platform and an UI friendly IPS/IDS front-end with features of monitoring, alerting, analysis, and feedback. The main problem of this system is that we will strength the snort’s fundamental monitoring and alerting functions, optimize the performance of snort’s rule matching mechanism and discover the potential network attacks by analyzing the threat level of each different network connection. Totally, the difficulty and creativity of implementing this system is to establish a new rule matching and processing mechanism on snort’s *Network Intrusion Detection System mode,* sniffer necessary data from the snort sensor, and represent the potential threats to users by data analysis.

II. SYSTEM DESIGN AND IMPLEMENTATION

***A. Modified Rule Matching Mechanism for Snort Platform***

In order to improve the running performance[5] of snort platform, we have three different methods:

a. *Reduce the amount of network traffics, which are unnecessary to monitor:* Choose those network traffics have attacking possibilities.

b. *Rules optimization:* Try to do some optimizations on rule matching, rule processing, and rule execution.

c. *Use the better algorithm of mode matching*: Try some methods to improve the performance of snort’s mode matching.

Our new mechanism for rule matching is based on choosing the “active” rules, which means these rule are efficient and have possibilities to execute. After completing the original rule set by the user, our new system will create a new rule set that only includes active rules. Thus, the core problem in this section is distinguishing “active” rules and “inactive” rules.

First of all, we add a property, *OptGroupNode*, for every rule-node in snort. The structure of *OptGroupNode* contains three values, which are flags, a pointer to the right, and a pointer to the down. Please refer the sample code below and see the Figure 2.1.

typedef struct OptGroupNode

{

int flags;

struct OptGroupNode \*right;

/\* ptr to the next OptGroupNode in the list \*/

OptTreeNode \*down;

/\* list of rule options to associate with this rule node \*/

} OptGroupNode



Figure 2.1 Regular Tree Schematic Drawing after Introducing OptGroupNode

Second, we build up an index for every rule in the rule set. To an original snort platform, the main function for parsing the rules is *ParseRulesFile( )* and this function can produce a pointer, *OptIndexNode* to the property of the rule-node.

Typedef struct OptIndexNode

{

int sum; /\* mathc’s frequentness\*/

struct OptTreeNode\*otn\_addr; /\* point the OTN node\*/

struct OptIndexNode\*next; /\* point to the next OIN node\*/

struct OptIndexNode\*former; /\* point to the former OIN node\*/

} OptIndexNode



Table 2.2 Regular Tree Schematic Drawing After Introducing OptIndexNode

Totally, we build a rule-tree to distinguish the “active” rules and “inactive” rule. A reduced-size rule set plays a crucial in improving the performance of rule execution. In the report, we called the new snort platform, *the modified snort*.

***B. The Solution system***

The main structure of the integrated system:

Front-End

Function Logic

Modified Snort

Figure 2.3 The structure for the system

In the Figure 2.3, we use the modified snort as a reliable network traffic sensor and fundamental service provider. We can find the basic data flow procedure in Figure 2.4.

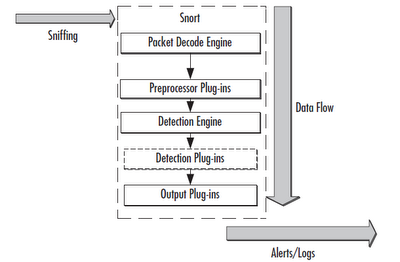


Figure 2.4 Data Flow Chart for Snort Platform

Different from the original snort, the modified snort can own a better execution performance in *Detection Engine* in the Figure 2.4. After recompiling the source code of the snort, we conducted an experimental test for the performance improvement and got the result. Please see the Figure 2.5 below.

Our performance test[3] is based on the input from *DARPA intrusion Detection Evaluation Data Sets*, provided by the MIT Lincon Lab. The average time of matching a rule in modified Detection Engine of Snort has reduced by 0.06195 ms (6% to 21%), which means that the improvement is significant.

Figure 2.5 The Contrast Chart of Snort Performance Test

Through the *Output Plug-in* in Figure 2.4, the upper application can obtain the data easily. In general, the upper applications can get results from the *Output Plug-in*.

|  |  |
| --- | --- |
| Data Type | Description |
| Packet Info | Including time stamp, IP header information, TCP header, source IP and destination IP. |
| Signature Info | Including the signature in each allowed connection |
| Rule Execution Info | Including the modified rule set and the rule matching for each packet. |
| Behavior Classification | Including each connection’s behavior information and it can be used to classify the connection. |

Table2.6 Table for the Descriptions of Different Data Types

***C. High-Level Description of Implementation***

In this project, we implemented an integrated system combing a modified Snort platform and a feature-rich front-end web application. This web application is scripted in Ruby on Rails platform. Our implementation is able to monitor and analyze the flow of the network as well as the security of the network. With Snort as a sensor, we could define our own rules to identify the risky activities such as illegal login attempt to the server or Denial of Service attack and then display and notify those risky activities to the user in the front-end.

Our tool has the ability to perform real-time traffic analysis and packet logging on TCP and IP networks. At the same time, the tool can produce some analysis charts and graphics to present the network traffic and network attacks, such as SYN Floods, DNS Spoofing attacks, and buffer overflow in the system, which will make it easy for the system administrator to figure out the potential security vulnerabilities in their system. For the demo of our application, please refer to the Appendix A.

***D. The front-end and function design***

By collecting and analyzing the necessary data from the lower platform, the front-end system can implement some core functions, which is shown in Table 2.7.

|  |  |
| --- | --- |
| Function | How to Implement |
| Threat Level | Based on the classification of behavior, the front-end system will label each connect a threat level, including risky, safe, medium. |
| Alerting | According to the Threat Level function, our system is able to identify the threaten connections and alert users by flag, figure and email. |
| Data Analysis | Based on our new rule set with “active” rules only, the system can collect the information of source IP, destination IP, port, payout, and other connection information. Then, we design a database to store these data. |
| Statistics | The system implements the statistical functions by adding the packages of ImageMagick and Wkhtmltopdf. |

Table 2.7 The Implementation for the Functions in the System

III. RESULT AND CONCLUSION

**A. Final results**

We have tested our implementation by performing several test cases, all the test cases had been successfully performed and passed, this means our design and implementation is correct. The *Table 3.1* showed the attack tests we had made and the result of these test. We obtain the normal network traffic by using the system normally, thus no specific test is performed.

For details of how we perform a test case and how our application handles such a test, please refer to the Appendix A.

|  |  |  |
| --- | --- | --- |
| Test Case | Output | Result |
| Invalid Login Attempt | The system successfully filtered out this kind of attempt and displayed it in the system. | PASS |
| Denial of Service Attack | The system successfully detected, blocked and displayed it in the system. | PASS |
| ARP Attack | The system successfully detected, blocked and displayed it in the system. | PASS |
| Policy Violation | The system successfully detected and displayed it in the system | PASS |
| Unauthorized Root Access | The system successfully detected and displayed it in the system | PASS |

Table 3.1

**B. Final conclusion**

Our system is a functional and UI friendly application that makes simple network security monitoring, alerting, and analysis possible. The system could recognize and analyze unexpected access to the network by effectively modifying the snort platform. Our own front-end application could produce analysis charts and graphics to both current and past network traffic. A network or system administrator could easily gain all the information on the network by using the user-friendly graphic interface.

Our system could also alert any usual network activities to help the administrator find the potential vulnerabilities in the current network environment. Also, the modified rule matching algorithm makes our new system performs better than the original Snort platform.

IIII. APPENDIX

1. **A brief application demo**

In this section, we will demo the basic function of our web application.

On the home page, our application will show a chart of the total number of security alerts during the past day, as showed in figure A.1.

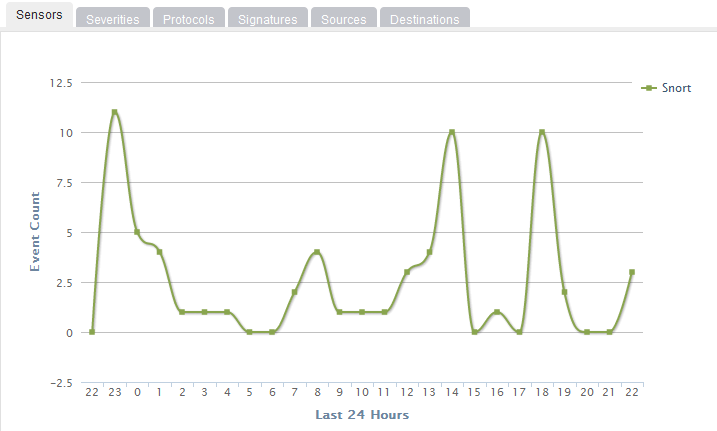


Figure A.1 Chart of total number of security alert

We can shift to each tab to get the detailed summary in different fields. For example, the Protocol tab will show all kinds of the protocols that had been used during the 24 hours, as showed in figure A.2.

In the Events page, we could see the detailed events information instead of the summary on the front page. Figure A.3 shows how the detailed events list looks like. The orange tag with number 2 means this activity may not necessary be a secure or a risky event, but administrator should pay attention to it. Other kind of tag includes red tag with number 1 and green tag with number 3, a red tag means the event may be an attacking attempt while green tag means such an event is a normal and safe event. With the different color we defined in our application, a system administrator could quickly tell if an activity is risky.

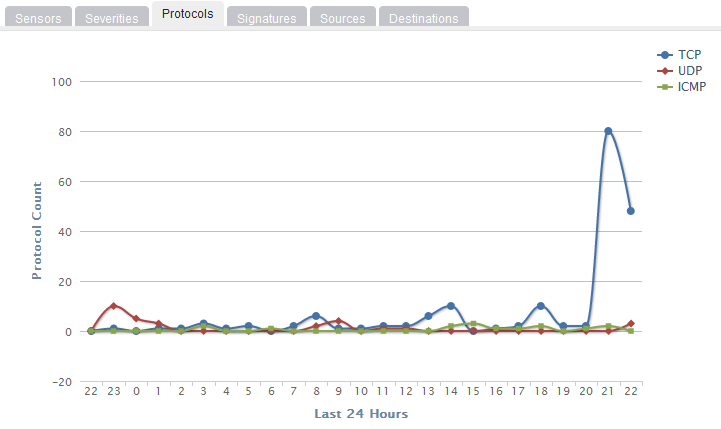


Figure A.2 Chart of different protocols

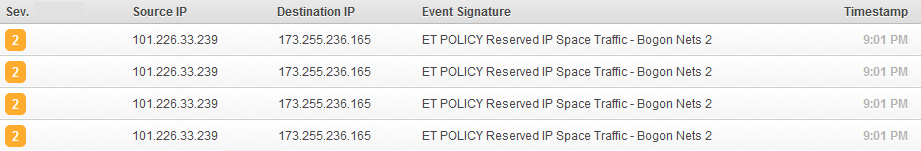


Figure A.3 Detailed events list

We can obtain further information of each record by click on the record, a new page with details of that record will show up, as in the figure A.4. The detailed information includes IP Header information, Signature information, TCP header information, Payload, etc. These information will be helpful when the administrator want know more about a certain records.

We would also like to include an example of how our system will record a simple attack attempt in our demo. If a remote user trying to login to the server by guessing the password of a certain user, this should be considered as the password guessing attack. Our application will identify this kind of attack and assign it with the red tag, which means this kind of login attempt may be an attack to the system and the system administrator should pay attention to it. A typical record of this kind is showed as figure A5.

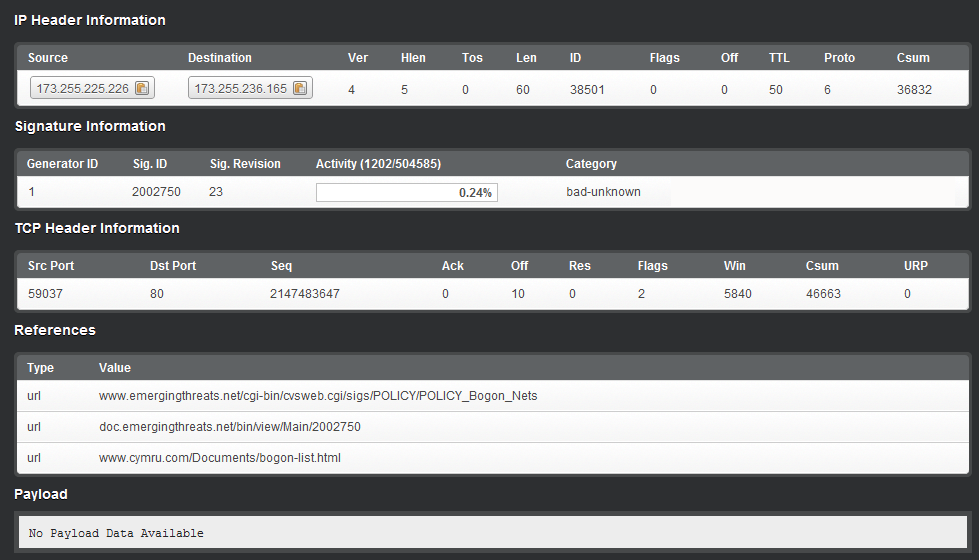


Figure A.4 Detail information of an event



Figure A.5 Records of bad login attempts

REFERENCE

[1] Rangadurai Karthick R., Hattiwale, Vipul P., Ravindran Balaraman, Adaptive Network Intrusion Detection System using a Hybrid Approach, 978-1-4673-0298-2/12/$31.00 c 2012 IEEE.

[2] Man Zhao, Jing Zhai, and Zhouqian He, Intrusion Detection System Based on Support Vector Machine Active Learning and Data Fusion, ISICA 2010, LNCS 6382, pp. 272–279, 2010.

[3] David J. Day, Benjamin M. Burns, A Performance Analysis of Snort and Suricata Network Intrusion Detection and Prevention Engines, ISBN: 978-1-61208-116-8, IARIA, 2011.

[4] Martin Roesch, Snort-Lightweight Intrusion Detection for Networks, 1999 LISA XIII- November 7-12, 1999-Seattle,WA.

[5] C. Jason Coit, Stuart Staniford, Joseph McAleney, Toward Faster String Matching For Intrusion Detection, 0-7695-1212-7/01 2001 IEEE