Implementation of Genetic Algorithms into SNORT, a Network Intrusion Detection System

Master's Project Proposal

Brian E. Lavender

Abstract

I propose the integration of genetic algorithms (GA) into SNORT to enhance SNORT at performing Network Intrusion Detection (NID). SNORT is popular Network Intrusion Detection System (NIDS) tool that currently uses a custom rule based system to identify attacks. SNORT has a customizable structure for developing plugins as can be seen by the Statistical Packet Anomaly Detection Engine (SPADE) plugin. I have identified two research papers using genetic algorithms for NID, one by Ren Hui Guong and the other by Wei Li. Guong's paper describes the implementation of ideas proposed by Li. Their research and SNORT's popularity naturally lead to the extension of integrating genetic algorithms and their application methodologies into SNORT.

Background

SNORT (http://www.snort.org) has become a popular Network Intrusion Detection (NID). A search on the Google search engine for “SNORT” results in a set that exceeds 100,000. Its main focus is a rule based detection system for identifying malicious traffic.

SNORT started as the pet project by Marty Roesch in November of 1998. Originally, it was created to examine network traffic on his cable modem. Later, he began to develop rules for identifying different types of traffic and alerting on them. Today, Sourcefire maintains the free software version of SNORT and distributes rule sets to registered users. There have been other efforts to create rule sets such as the SNORT bleeding rules. Below is an example snort rule taken from the chat rules found in current snort rule snapshot snortrules-snapshot-2.8.tar.gz [SNORTrules].

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS
CodeRed v2 root.exe access"; flow:to_server, established;
uricontent:"/root.exe"; nocase; reference:url, www.cert.org/advisories/CA-
2001-19.html; classtype:web-application-attack; sid:1256; rev:8;)
```

It identifies the CodeRed worm [Kohlenberg] that wrecked havoc on the internet a few years ago. In order to develop this rule, an administrator trained in the SNORT rule syntax had to determine what traffic is not desirable, examine it for identifiable attributes, and then create the rule.

Beyond writing specific rules, SNORT supports a modularized architecture allowing developers to write customized plug-ins for it.

Artificial Intelligence, SNORT, and SPADE

The primary focus of SNORT isn't on Artificial Intelligence methods, but has focused on developing explicit rules by a team of experts. At the same time, various studies have been performed using soft based computing for Network Intrusion Detection including Fuzzy Logic, Artificial Neural Networks (ANN), Probabilistic Reasoning, and Genetic Algorithms [Farshchi]. James Hoagland wrote Statistical Packet Anomaly Detection Engine SPADE [Farshchi] taking advantage of the pluggable type architecture of SNORT. It monitors traffic and maintains a statistical probability table for IP addresses and port destinations. When a packet arrives, SPADE calculates an anomaly score for the packet. Anomalous traffic generally occurs with an attack or malicious traffic. SPADE operates regardless of the rule set and uses probabilistic analysis to do its job. Farschi [Farshchi] in his analysis of SPADE notes that while rule based analysis used by SNORT provides reliable results for detecting malicious traffic, it has two downsides. One, being that maintaining the rule sets can be a burden to the security

professional. Two, rule based methods have no way of identifying new attacks for which no rule is available. In addition, he points to other Artificial Intelligence techniques such as Artificial Immune System, Control Loop Measurement, and Data Mining as effective methods for identifying malicious traffic. SPADE supports the idea that other Artificial Intelligence techniques can be incorporated into SNORT.

Genetic Algorithms

The book *Artificial Intelligence, A Modern Approach[Norvig]* offers a detailed explanation of Genetic Algorithms. Genetic Algorithms follow the process listed below, which can also be seen in Illustration 1 [Pohlheim] .

1. Initialize population
2. Calculate fitness of population.
3. Perform selection. Roulette wheel is technique that randomly selects chromosomes giving proportional weight to chromosomes with higher fitness.
4. Perform crossover
5. Determine if mutation occurs.
6. Go back to step 2.
7. Quit

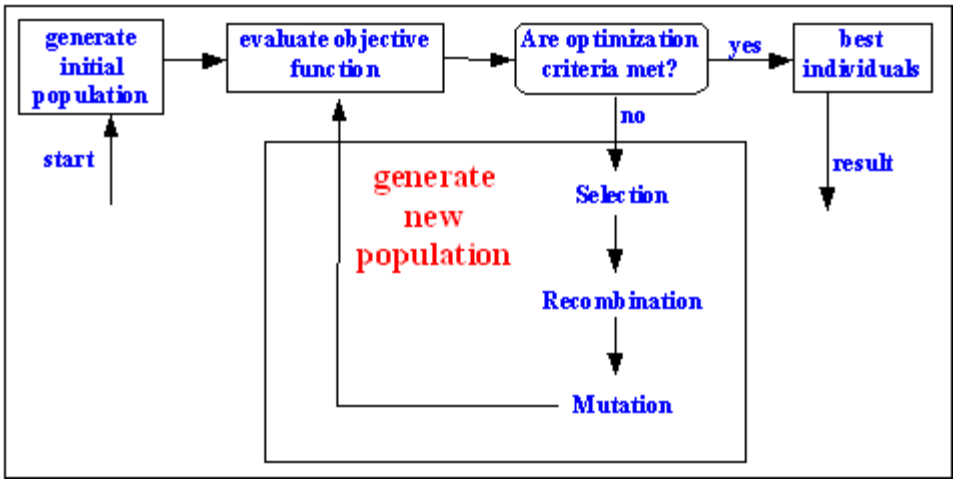

Illustration 1: Structure of a Simple Genetic Algorithm (Pohlheim)

The basic concepts of Genetic Algorithms are simple, yet the process of choosing the gene representation, a good fitness function, and even application of the recombination [Whitley] can be the key to successful use of Genetic Algorithms.

Previous Genetic Research for Network Intrusion Detection

Wei Li [Li] wrote a proposal for using GA in a NID and Ren Hui Guong [Guong] followed with his implementation. Li set the foundation for creating a system using Genetic Algorithms that analyzes DARPA datasets and Guong created an implementation using ECJ [ECLab] (A Java-based

Evolutionary Computation Research System). While Guong reported to have created an implementation, his group only provided pseudo code and class diagrams. No actual code was provided, yet it appears it would not be difficult to produce using the ECJ library. Li proposed using DARPA data sets [DARPA] from MIT Lincoln Laboratory for training and testing . These dataset records each contain 9 attributes (Starting Date, Starting Time, Source IP Address, Destination IP Address, Source Port Number, Destination Port Number, Duration, Protocol, and attack name). This DARPA training data is actually a result of test network traffic data, a Sun Microsystems Solaris and the use of Sun's Basic Security Module[Sun]. The datasets used in both papers were created in 1998. Today's attacks have changed with regard to rule based systems, but I believe the training data should still work well for developing a genetic algorithm implementation as genetic algorithms should adapt to the attacks.. In both Li's proposal and Guong's approach they create a fitness function and a chromosome type for the Genetic Algorithm.

Natural Extension of SNORT

Li points to future work in creating standard test data, yet Guong continued the implementation of Li's proposed methodologies. Farschi notes the need for more Artificial Intelligence techniques for use in intrusion detection. The SNORT tool has proven its self popular with the security community and SPADE provides evidence that Artificial Intelligence methodologies can be integrated into SNORT. The mentioned items naturally lead to the idea of better integrating genetic algorithms in SNORT.

Proposal

My proposal is to take the same Genetic Algorithm technique performed by Guong and proposed by Li and implement it in SNORT using the same DARPA datasets for testing. Using SNORT with genetic algorithms creates an unique situation. In both Li and Guong's work, they use the DARPA BSM format for training and testing. With SNORT, it reads raw network data, libpcap format[Kohlenberg]. For the genetic algorithm to work in training mode with SNORT, the test data and the training data have attributes and an indicator of what kind of record it is. The following two records shows normal traffic and the following record shows an rlogin attack in the DARPA BSM format.

Regular Traffic

127 01/23/1998 17:00:04 00:00:01 ftp-data 20 43550 192.168.1.30 192.168.0.40 0 -

rlogin attempt breakin

128 01/23/1998 17:00:05 00:00:14 rlogin 1022 513 192.168.1.30 192.168.0.20 1 rlogin

Since SNORT is rule based and has plugins, I will develop a program or plugin that will evaluate the DARPA training data using the same same attributes identified by Li and Guong and integrate the results into SNORT. Second, the testing part, I will write a module that will process the equivalent libpcap format DARPA data and determine how well the rules created from the training part correctly identify the equivalent malicious DARPA records in the test set.

Timeline

Reasearch and Development	1-2 months
Implementation	1-2 months
Training and Test	1 month

Writeup	1 month
Approval	3 weeks

Bibliography

SNORTrules: , Snort Rules, https://www.snort.org/snort-rules/ , Accessed August 29, 2009.

Kohlenberg: Toby Kohlenberg; Brian Caswell; Jay Beale; Andrew R Baker, Snort: IDS and IPS Toolkit, Syngress, 1-59749-099-7 2007, 180-183, 304-305 .

Farshchi: Farshchi, Jamil, Intrusion Detection FAQ: Statistical based approach to Intrusion Detection, http://www.sans.org/resources/idfaq/statistic_ids.php , Accessed August 29, 2009.

Norvig: Norvig, Peter. Russel, Stuart, Artificial Intelligence, A Modern Approach, Prentice Hall, 0137903952 2002, 116-119 .

Pohlheim: Pohlheim, Hartmut, “Genetic and Evolutionary Algorithms: Principles, Methods and Algorithms.”Genetic and Evolutionary Algorithm Toolbox, http://www.geatbx.com/docu/algindex.html , Accessed August 29, 2009.

Whitley: Whitley, Darrell, A Genetic Algorithm Tutorial, Accessed 1994 .

Li: Li, Wei, Using Genetic Algorithm for Network Detection, Accessed May 24-27, 2004 .

Guong: Ren Hui Gong, Mohammad Zulkernine, Purang Abolmaesumi, A Software Implementation of a Genetic Algorithm Based Approach to Network Intrusion Detection, Accessed 2005 .

ECLab: Luke, Sean et al., A Java-based Evolutionary Computation (ECJ) and Genetic Programming Research System, http://cs.gmu.edu/~eclab/projects/ecj/ , Accessed August 2009.

DARPA: , MIT Lincoln Laboratory, DARPA datasets, MIT,USA, http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html , Accessed August 29, 2009.

Sun: , Security Audit, http://www.sun.com/software/security/audit/ , Accessed August 29, 2009.

Kohlenberg: Toby Kohlenberg; Brian Caswell; Jay Beale; Andrew R Baker, Snort: IDS and IPS Toolkit, Syngress, 1-59749-099-7 2007, 180-183 .