Peter Hogya

ISS Project Part 3 : Results Analysis

Over the course of the project each implementation was evaluated using equations for their precision, accuracy, and recall. Additionally for the first part each screen capture includes the runtime to process the pcap file. In the first part of the project decision trees were used to differentiate between attacks and normal traffic. This included both a signature based implementation, where attacks were labeled individually, and an anomaly based implementation, where all attacks were labeled the same. In the second part of the project neural networks were used and both signature and anomaly based systems were implemented again in the same way. In both parts each implementation was evaluated at a theoretical stage where the precision, accuracy, and recall were calculated based on the algorithms ability to differentiate traffic in Weka. In the first part the implementation was also evaluated at a practical stage where the algorithm was implemented in Snort and the equations were calculated again to find a more realistic measure of success. I believe it was important to implement and test both of these stages where possible as applying an algorithm to a real world problem can be more complicated than it seems and is liable to prove less effective than expected. These complications can be seen in the neural network implementation where applying the results from Weka to Snort's rule syntax was found to be an exceedingly difficult and time consuming process. However this further highlighted the concern in the decision tree implementation that IDS rule syntaxes would greatly benefit from range based rules and support for inequalities.

As expected the performance for the decision tree implementations were higher in the theoretical stage than in the practical stage. All performance values were above 99.5% for both the signature and anomaly systems. The practical application in Snort yielded mostly very positive results. The W32/Sdbot Infected attacks had a slightly lower recognition rate than in Weka and the JexBoss Exploit attacks were not recognized at all, however they were marked by Snort as having bad checksums. This may have led Snort to dropping them before they even reached the rules meant to catch them. Aside from those two attacks the rest were recognized at a similar rate as in Weka and overall performance did not suffer tremendously, overall accuracy was at exactly 99%. The anomaly based implementation proved more difficult to evaluate as a single attack packet tended to trigger more than one alert. For example there were

only 692 Neutrino Exploit attack packets and 967 alerts went off for that set. One aspect that can be evaluated is the number of false positives. Since for one set the normal traffic file was run through it without any attacks then any alerts in that set are false positives. For that set 65 alerts went off as opposed to 1 alert in the signature based system. This increase in false positives is expected of an anomaly based system however.

For the neural network implementation only the theoretical stage was implemented. The algorithm was adjusted using several parameters to achieve better results for detecting each individual attack and all attacks in both a signature and anomaly based system. Some of these attacks were detected with perfect accuracy, W32/Sdbot Infected, Packet Injection, and the Malspam and Neutrino Exploit were also fairly high. The JexBoss Exploit attack again proved problematic with recall below 0.5. Some additional trimming at the data processing stage led to significantly reduced samples of the JexBoss Exploit attack which may have proved quite detrimental to the neural network's performance. More concerning was the performance for the full signature and anomaly based systems which had recall rates of 0.706 and 0.675 respectively. Clearly the added complexity of a neural network does not easily lead to improved results. It's possible the tremendous range of the IP address space in addition to the large range of computer ports was simply more difficult to model an equation using a neural network than simply feature ranges from decision trees. However the results for the perfectly detected attacks were very promising as none of the attacks were detected perfectly by the decision tree implementation. This illustrates the wide range of potential success with neural networks and highlights their high upper bound performance.

One of the problems faced with the neural network implementation was attempting to adapt the resulting equation into a ruleset in Snort. In the first part of the project I noted a similar problem and that Suricata has similar limitations. In both Snort and Suricata rules are generally assumed to be specific to a particular combination of packet features. A rule can only check for a specific packet length or ACK flag value. There are two features with exceptions to this, IP addresses which may include a range using CIDR addressing and further by comma separated addresses and port numbers which use a simple syntax that supports ranges and inequalities. This proves fairly problematic when attempting to apply the results of many machine learning algorithms. The ultimate goal of a machine learning algorithm is to take known points across the n-dimensional feature space and create a map of expectations for unknown points. This type of solution is not easily or effectively applied to a system that expects an explicit point for point solution. What could be done with a single equation is instead

done with hundreds or thousands of rules. Decision trees are a special case because they deal directly with ranges as an output rather than equations that use the features as variables. This means that building a ruleset from their output is simply a matter of automating rule creation to these ranges and hoping that the ranges are small enough that this does not take a tremendous amount of time. Luckily the attack space was magnitudes smaller than the normal traffic space in this project. However neural networks, support vector machines, and regression algorithms all generate equations that attempt to classify unknown data points. To generate a complete rule set would require either a complete traversal of the feature space, a monumentally large task given the range of IP addresses, or a comprehensive mathematical analysis of the behaviour of the resulting equation that would likely require a tremendous amount of human work. This problem could be very simply resolved by adding more functionality to the rule syntaxes of these IDSs that would allow for things like ranges and evaluation of packet features using math functions. Without such functionality many packet features have to be omitted from machine learning analysis and many algorithms cannot be feasibly implemented. Thus one significant improvement to IDS rule syntaxes would be supporting this kind of functionality.

I believe this type of approach to computer security and to problem solving in general is very effective. Utilizing large amounts of data to predict the outcomes of complicated systems is the purpose of machine learning tools and computer security is absolutely a complicated system with large amounts of potential data. However accessibility to that data can be more complicated and current tools are not particularly compatible with many machine learning algorithms. However Suricata now supports Lua scripting to deal with more complicated rule systems, this may circumvent many of the problems with this approach and Snort. The separation of theoretical and practical testing and evaluation is also very important. It provides an upper bound on your expected performance and your actual performance and identifies which area requires improvement. If the practical evaluation matches the results of the theoretical evaluation then the algorithm and data set used must be improved, otherwise the implementation can be improved to match those results. Ultimately using decision trees will almost certainly provide an effective ruleset more quickly and in a more easily implemented format than neural networks however the added complexity of neural networks means that once the proper tools are set up and parameters tuned appropriately they are likely to produce better results. The choice between them depends mostly on your time constraints and necessary accuracy.