Building Baseline Preprocessed Common Data Sets for Multiple Follow-on Data Mining Algorithms

Charles A. Fowler
cfowle1@students.towson.edu

Robert J. Hammell II
rhammell@towson.edu

Department of Computer and Information Sciences
Towson University
Towson, MD 21252, USA

Abstract

Organizations across all domains and of all sizes wrestle with the problem of "coping with information overload." They ingest more and more data, in new and varied formats every day, and struggle increasingly vigorously to find the nuggets of knowledge hidden within the vast amounts of information. Furthermore, due to the various and pervasive types of noise in the haystack of data, it is becoming exceedingly difficult to discern between shining false shards and the true needles of knowledge. In the grander scheme of our work we intend to demonstrate that, in the realm of offline network and computer forensic data mining, several data mining applications reporting to a hybrid intelligence/multi-agent systems based, overarching layer for interpretation and interpolation of the findings will yield more accurate results than any one data mining application acting on its own. In this paper we discuss steps required for generating suitable test data, and then take a look at initial results rendered by various data mining algorithms (classifiers, clusterers, associators – future work will include MapReduce job results).

Keywords: Intrusion Detection Systems, Hybrid Intelligence, Multi-Agent Systems, Data Mining, WEKA

1. INTRODUCTION

The "needles" of knowledge are in there, all the while, more and more "hay" collects in the fields of data, more than we can possibly sift through in a timely manner, and it grows more and more stale as the minutes whiz by. On our battlefields, national borders, college campuses, and cities we have more cameras, microphones, antennas, information capture and logging devices than ever before. Our ability to gather and ingest more types of data increases, while our capability to cull useful nuggets of actionable information from therein decreases (Gantz, J., Boyd, A., Dowling, S. 2009).

Data mining applications represented the first solid and effective attempts at combating the "coping with information overload problem." To an extent, traditional and innovative variations of data mining techniques have kept apace of the problem spaces to which they were tailored as long as the complexity of the data remained at a minimum. More often than not though, in most problem spaces, the solutions have fallen behind.

Typically, these knowledge discovery solutions were built and used in domains where data was collected for the specific purpose of harvesting

the knowledge contained therein. Not so in the Information Security realm. Despite the availability and ready acceptance of knowledge discovery through data mining in these other fields, the information security domain has not been so quick to implement it. As Markey points out, "Historically, data mining techniques have not been adopted in the IT security community. This lack of adoption has been for a number of reasons including the difficulty obtaining the necessary data and a lack of awareness about the techniques." (Markey & Atlasis, 2011) Suh notes other barriers to entry for data mining into the IT security field: "Simple, blind application of data-mining algorithms can lead to the discovery of meaningless and useless 'knowledge' from databases. Hence, data-mining systems have to be carefully customized to fit the individual needs of the intended users." (Suh, 2011) Customization and specialized domain knowledge mean more entries on the cost side of the ledger, often difficult to justify when things are working "well enough" already.

The larger scope of our research takes a look at the "data mining for Information Technology security" answers that are out there, measures the efficacy of a number of the traditional and newer generic and open source data mining techniques, and compares and contrasts them. This is done to set a baseline to measure their levels of effectiveness and see if there are any peaks or valleys across the spectrum when these technologies are applied in this specific domain.

Then we intend to look at whether or not accuracy can be improved by using them in a complementary fashion, governed by a hybrid intelligence-based "super layer". If each of several traditional data mining methodologies are likened to several blind men describing an elephant ("it feels like a wall!", "no, it feels like a hose!", "it feels like a tree trunk!") then this intelligent "over layer" would ingest each lower object's inputs, synthesizing them and creating a more comprehensive picture. Potentially, this layer would also provide a feedback loop to the underlying objects, tweaking their settings to increase their effectiveness. We hypothesize that several data mining algorithms acting in concert together will yield more nuggets of knowledge from a pool of unsifted data than will any single algorithm (single "blind man") on its own, no matter how well tweaked or crafted.

In this paper we outline the initial steps required to generate, clean, pre-process and prepare seeded test data, and begin the data reduction and projection steps. Additionally we compare and contrast results rendered by several data mining algorithms (e.g. classifying, clustering, associating). The remainder of this paper is organized as follows: Section 2 provides a background discussion on the information overload problem and intrusion detection systems. That is followed by a presentation of the research approach in Section 3 and preliminary results in Section 4. Finally, Section 5 discusses conclusions and future work.

2. BACKGROUND

Information Overload

The various efforts over the years focused on "coping with information overload" have gone by a number of names including: data pattern processing, information discovery, data mining and knowledge extraction. While the problem to be addressed in this research falls into the broad area of information overload, we have narrowed down the investigation to the specific domain area of information systems intrusion detection. Specifically, evidence of exploits or hacks (attempted or successful) represents the "needles" of information that we seek.

Abraham's (Grosan, Abraham, & Chis, 2006) diagram in Figure 1 shows a generic graphical view of the knowledge discovery process and he outlines (below) the following basic steps for its accomplishment:

1. Developing and understanding the application domain, the relevant prior knowledge, and identifying the goal of the "Knowledge Discovery in Databases" process.
2. Creating target data set.
3. Data cleaning and preprocessing: basic operations such as the removal of noise, handling missing data fields.
4. Data reduction and projection:
   a) Finding useful features to represent the data depending on the goal of the task.
   b) Using dimensionality reduction or transformation methods to reduce the effective number of variables under consideration or to find invariant representation of data
5. Matching the goals of the "Knowledge Discovery in Databases" process to a particular data mining method

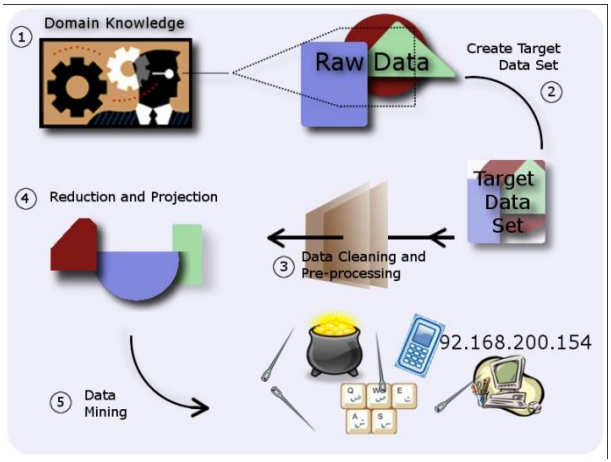

Figure 1. Steps of the knowledge discovery process, adapted from (Grosan, Abraham, & Chis, 2006)

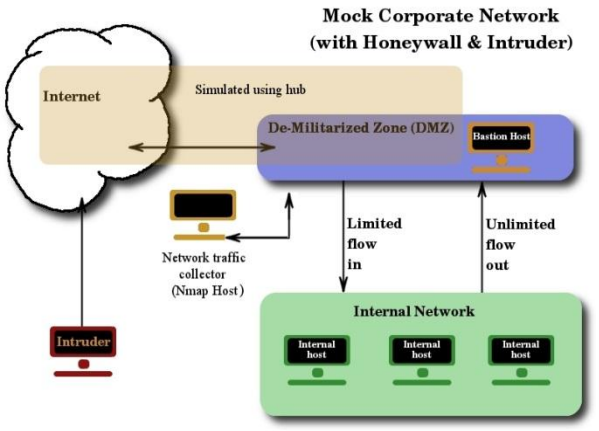

Figure 2. Lab network for generating seeded data

Intrusion Detection Systems (IDS)

As mentioned, this research is focused within the intrusion detection system (IDS) domain, which provides a rich environment wherein massive amounts of data can be easily generated in a controlled fashion. This will be accomplished through a lab network setup to mimic a real world corporate network attached to the Internet, with a DMZ, complete with an intruder conducting intrusions on the lab network, as shown in Figure 2.

Despite the best intentions of the creators of the early iterations of the "many to many" networks we have come to know as "The Internet," the world's malefactors would not be kept out, and they quickly began plying their age old practices in this new cyber world. (Settles & Rylander, 2002) In response to that, many intrusion detection systems have cropped up, hoping to catch and even prevent unauthorized access to networks.

In order to do their job effectively, these systems generate massive amounts of data, capturing every packet which traverses an internetwork gateway for collation (reconstruction of a communications stream, scattered by the nature of TCP/IP) and correlation (matching upstream and downstream packet flows, or "sides" of a network conversation). After the data have been collated and correlated, the intrusion detection system then begins the laborious task of sifting through the pile, looking for evidence of illegitimate traffic, employing a variety of techniques, most of them signature-based. (DeLoach, Oyenan, & Matson, 2008)

As a hopeful improvement on these systems, *intrusion prevention systems* (IPS) attempt to perform this analysis in real time, interacting with network guards to shutdown perceived unauthorized access. While our efforts currently only attack the offline data problem (IDS), it is expected to apply to real time data (IPS) in the future.

Intrusion detection systems generally fall into one of three categories (Pfleeger, 2003)

- NIDS - Network intrusion detection systems typically have a network port operating in promiscuous mode capturing every packet from the network segment to which they are connected.
- HIDS - Host intrusion detection systems are typically a software component installed on an individual host computer.
- DIDS - Distributed intrusion detection systems are a network of any combination of either host-based or network-based IDSes which are distributed in one network, or across network segments. The DIDS is differentiated by the presence of a centralized manager to which the distributed nodes report.

Each of these types of IDS may monitor and inspect one of several components of the information system/network to include (but not limited to): applications, file systems, log files, host to host(s) communication patterns, network flows and packet inspection. They typically accomplish these tasks by looking for changes in

behavior using either a signature based (pattern matching) or heuristic approach.

IDSes notoriously produce many false positives requiring human interaction or oversight to weed out the known innocuous events. (Pfleeger, 2003) Additionally, given the unique nature of every network and host on the network, heuristic IDSes can take a long time to train. (Pfleeger, 2003)

3. APPROACH

In this paper we report the phase of our research dealing with steps 1 through 4a of the bullets outlined by Abraham above (and spelled out in more detail below), which can be summed up as data gathering, pre-processing, and initial runs of the data through each of the selected data mining algorithms. We use this last step as an aid for data reduction and projection, as it provides a first shot and quick look at the data after it's been mined, validating (or invalidating) that the chosen features will be suitable for mining by machine intelligence.

These are exceedingly critical steps, and careful attention must be paid to each one to ensure that the follow on processes will be accurate and yield the desired results (i.e. the nuggets of knowledge that we ultimately seek). Successful performance of these steps requires domain knowledge (in our case, information security/intrusion detection), and the collection of meaningful data, which is "clean" and can be well understood by the algorithms for training, or has attributes suitable for meaningful interpretation. The importance of these pre-data-mining phases cannot be underestimated or done in a haphazard manner. Abraham's steps are elaborated upon below:

1. "Understanding of the application domain…" The previous section briefly outlined the development and understanding of the application domain, and the relevant prior knowledge. We identify the goal of knowledge discovery as finding the intruders, network traffic anomalies or malicious activity in the seeded data. This target changes for each set of data. Each data set may have (probably has) multiple instances of this.

2. "Creating target data set..." In this step, we generate and capture experimental intrusion detection data to serve as training sets within these labs via a host running Wireshark (Wireshark.org, 2012) sitting on the network. Other sources such as openpacket.org provide similar, suitable data. (OpenPacket, 2012) From this controlled data, concrete metrics can be drawn to enumerate the strengths and weaknesses of various existing solutions.
3. "Data cleaning and preprocessing…" This step consists of time consuming operations such as removing noise, handling missing data fields and the ensuring validity of the data. In addition to custom scripts written by the authors, some of the tools used for this step are AtoZ, GTVS and fullstats. (Moore & Zuev, 2005) Additional tasks in this step usually also include reformatting of data from whatever generated it into the various formats expected by the data mining programs. In our case this meant extracting relevant fields from pcap, to csv, to .arff format for Weka (Weka, 2011). An additional step of putting the data into a relational database management system can also be done.

4a. "Finding useful features to represent the data" At this stage of the process one either pares out data which is extraneous or adds in attributes (such as unique keys) or additional information not included in the collected data but known to the researcher. With one exception, to this point we have neither added nor deleted any columns in the data, leaving everything there to allow for better comparison and contrasting between the various algorithms. The exception is in the case of the Associator, Apriori, which "cannot handle numeric attributes!" (Weka, 2011) In this controlled environment, we happened to know the hostnames and so converted the IP addresses to text; however, "in the wild" this issue will need to be addressed before any solution is scalable.

Lab Environments

Several labs were created to accomplish this research. One lab network mimicked the DMZ of a real world corporate (internal) network, with a simulated Internet (external network) and intruders conducting cyber-attacks against the bastion host as shown in Figure 2. This network facilitates the creation of "seeded" sample intrusion detection data for training the data mining algorithms. Much freely available "seeded" sample data exist, as do volumes of real data with potential actual cyber intrusion attempts; however, having customizable known data with specifically interesting intrusions or certain types of exploits is required for a controlled (data mining) testing environment.

Components of the lab environment briefly described herein have been built and its construction and implementation have been recorded in exhaustive detail. These notes include step by step instructions for how to set up a small "internal network" of Linux and Windows based computers (i.e. corporate network), with a DMZ (de-militarized zone) and attendant hosts, an "external network" (i.e. the Internet), as well as hosts suitable for launching attacks. The documentation outlines how to choose and launch the attacks, and how to build a "glass box" framework around it all which will capture and package all of the interesting data. This "glass box" consists of a Linux based host running Wireshark sitting on the link between the DMZ and the (simulated) Internet serves as the capture point for all traffic coming in and out of the corporate network.

Experimental Data

The Intrusion Detection domain, one practical application of data mining, has several well known and painful shortcomings as previously outlined, and their measures of success or failure are easily quantified. This arena provides an excellent backdrop against which we can test our hypothesis. That being, in a nutshell, that multiple, and different data mining algorithms/technologies working in parallel, reporting to a higher multi-agent system/hybrid intelligence based layer, will yield much more knowledge from data, or will cope better with information overload than will any single algorithm on its own. The current research will be limited to the examination of offline data.

Test data were acquired in several ways. Training data (the subject of this paper) was generated by capturing simulated attacks within the lab, as well as downloading publicly available tcpdump/Wireshark (pcap) files. This is standardized output, whose formats are used extensively throughout the information security community, making the test data portable and easily accessible to any collaborators or follow on work. (Ramirez, Caswell, Rathaus, & Beale, 2005) Additionally, it is a trivial process to generate very large files suitable to the needs of this research, and it is also easy to seed the data with known "needles" or nuggets of information (within the scope of our domain, represented by exploits or hacks) to set up a control environment for experimentation and discrete measuring.

Data Mining

The open source datamining framework, WEKA was the tool we used for testing the traditional classify, cluster and association algorithms. Weka was chosen for a variety of reasons, not the least of which is because it is "well-suited for developing new machine learning schemes." (Revathi & Ramesh, 2011) It also enjoys widespread community support and has many algorithms from which to choose. We established a baseline of results for traditional data mining sweeps through data, to show how (at the very least) each one would quantify the same data set. Any measure of their effectiveness will be gauged in future research; the current goal is only to create, capture or otherwise accrue suitable intrusion detection data sets, ensure that they are properly pre-processed and that they can be successfully run through our program (Weka) without issues. Future efforts will also include running the data through a MapReduce based program. Of the three types of algorithms we report results (Appendix C) on the following.

- Classifying
  - UserClassifier - weak classifiers trees (a novel interactive decision tree classifier) (Revathi & Ramesh, 2011)
  - J48 Tree (Markey & Atlasis, 2011)
- Clustering – Cobweb, SimpleKMeans
- Association - Apriori

4. PRELIMINARY TRAINING SET RESULTS

A variety of typical hacker or penetration tester actions were initiated from the "intruder," directed at the "bastion host" and "internal host." (see Figure 2). These network activities were captured by the "network traffic collector/Wireshark host" and saved as pcap files. Pcap files are binary files and cannot be read directly into most data mining applications. Weka (our data mining platform of choice for this research) will accept *.csv files, *.arff files or a connection to a database. For this exercise we used tshark (Wireshark.org, 2012) to extract the interesting fields, creating *.csv output. The *.csv was opened in a spreadsheet program for checking and verification of the data.

ISSN: 2167-1508
v5 n2239

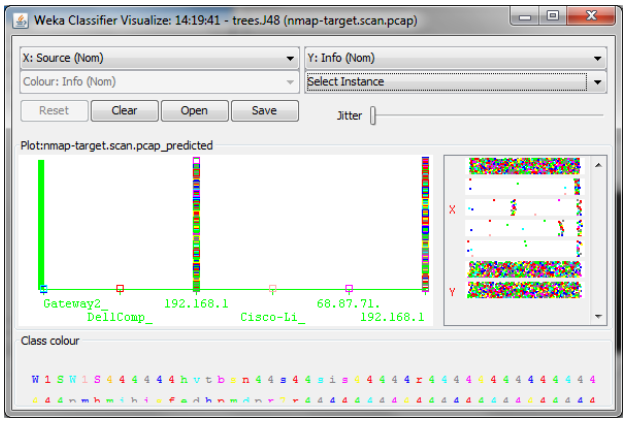

Figure 3. J48 Classifier results of nmap-target.scan

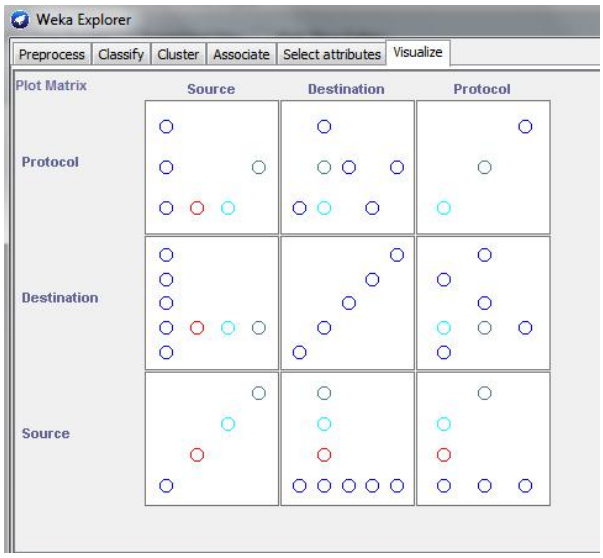

Figure 4. Associator - results of nmap-target.scan

As part of the preprocessing step, information can be injected into to the data which will be useful for the training of the data mining algorithm. Additionally outlying data points (garbage) can be eliminated if it will not serve the overall process.
To begin running this data through the algorithms spelled out above the .csv file (output converted from *.pcap of the nmap scan) was opened in Weka explorer, through the preprocessing tab. For purposes of these initial runs, we selected all of the attributes (6, in this case), except for Apriori which cannot accept numerical values. In that case, we changed what values we could to text (ip – names) and dropped the sequence attribute, which was spurious information for the associator. This adaptation left us with 3 attributes (Protocol, Destination & Source) containing only numeric data.

Figure 3 shows the results in Weka of a J48 Classifier training run on data acquired during an nmap scan of a target. The verbose output from each algorithms' run is included in appendix C. The results described herein were derived from a single nmap target scan, where a "hostile" host scanned a "victim" computer, probing for open/exploitable ports and it was performed in a relatively quiet and clean (network traffic wise) environment.

For brevity's sake, figures for only two of the five listed algorithms are inserted herein, however each algorithm (so far) yields expected visual results, and the Cobweb and SimpleKMeans clusterers' results are almost virtually identical to the classifiers'. For instance, Figure 3 shows two clear stacks, or relationships, between the "attacker" and "victim" hosts, where the attacker is a source for the victim (of the bulk of the traffic captured during the run) and vice versa, with other hosts' spurious/sporadic showing up as blips across the bottom.

The plot matrix (in Figure 4) generated by Weka's Apriori Associator shows a clear straight line correspondence (or association) whenever source/attacker and destination/victim hosts are plotted.

Given the relatively simple nature and limited actors of these exchanges, some of the mining results appeared visually similar, as one might suspect, for the algorithms tested. This in itself, is a useful observation, and it will be interesting to see in our future work how, or if, the results begin to vary as complexity and noise in test data increases.

Although, at this stage, both the clusterers and classifiers render visually similar results (and this is expected) each operates in different ways; classifiers are predictive, supervised machine learning, whereas clusterers are descriptive, unsupervised machine learning methods. (Aberer, 2012) Their approaches to mining data are completely different, allowing each to look for and find different things, a strength to be leveraged in ongoing research. These variances in methodologies will allow the proposed Hybrid Intelligence/Multi-Agent System (HI/MAS) overarching layer to possibly

play one algorithm off of the other, or at least provide multiple and varied insights into the data not offered by one algorithm alone. So far, the test results from the experimental data have fallen within expected boundaries. It remains to be seen whether or not more noisy, or real life data will cooperate as well.

The results outlined in this section serve as a baseline for future research and are relatively simple visually and textually (Figures 3 & 4 and Appendix C). The collected data were purposefully generated in such a fashion as to be devoid of as much noise as possible in order to create a "control set." In the future this will be compared against data captures either "from the wild," or from a controlled environment (Figure 2) wherein quantifiable noise has been injected. These initial results provide a foreshadowing of what a given exploit or "needle" may/should look like in each of the various types of data mining algorithms.

5. CONCLUSIONS AND FUTURE WORK

There is no question that the coping with information overload problem exists; there is also no question that current solutions are unsatisfactory and researchers are invoking unique and new ways to attack the problem (Moore & Zuev, 2005), (Habib, Sallam, & Badawy, 2008), (Settles & Rylander, 2002), (DeLoach, Oyenan, & Matson, 2008), (Zhang & Zulkermine, 2006). The increase in volume, complexity and lack of homogeneity of data only heightens current systems' painful shortcomings. Having said that, given the level of niche successes that various tools demonstrate, we do not feel it is necessary to completely re-invent the mousetrap. Rather, we believe that the current tools can be teased apart and re-woven together with recent developments in other domains to create a robust, unique and effective problem solving system.

Can the speed of data mining be increased? It should go without saying that adding an extra layer of processing on top of an already sluggish system will certainly not increase the speed. However, if the monolithic system can be recast into a new paradigm (Hybrid Intelligence/Multi-Agent System), wherein it is broken down and parceled out to simplified, speedy agents (offloading and parallelizing processing, memory and other resources), it is quite possible that overall speed may increase.

Additional factors for future consideration may be in the area of tradeoffs. For example, speed may be desired over accuracy or vice versa.

Can the accuracy of current data mining applications be enhanced? Apart from the question of speed, certainly more sets of eyes and ears are better than one, and the integration of diverse applications, playing to the strengths of each will most certainly yield more and hopefully larger nuggets of knowledge from the ever expanding volumes of data.

In the small picture, our current research focuses on comparing and contrasting the strengths and weaknesses of various data mining algorithms in the intrusion detection domain. On the larger landscape, this stage begins identifying, quantifying and qualifying viable tasks for these proposed agents to undertake. The preliminary results shown herein begin to reveal the lay of the foundation, hinting at what each tested component can bring to the table and how they may fit together in a future superstructure.

In addition to the re-iteration of previous tasks as required, future research will pick up the balance of the knowledge discovery tasks (tasks 4 and 5) laid out by Abraham as follows:

- "Data reduction and projection…" finding useful features to represent the data depending on the goal of the task. Using dimensionality reduction or transformation methods to reduce the effective number of variables under consideration or to find invariant representation of data. Mine the data using several conventional offline data mining tools, in a fashion that will generate well defined, repeatable and measurable results. An environment for testing various data mining applications. (WEKA, Hadoop!)

- "Matching the goals..." Our future tasks include simulating the Hybrid Intelligence/Multi-Agent Systems (HI/MAS) based layer fed by the underlying data mining processes, in an effort to prove that the algorithms working in tandem will complement each other, producing better output than one of them could by itself.

We will compare and contrast the results from each individual type of run, applying a weighting or valuation scheme on the results to assign levels of "success." Points in the weighting scheme include the obvious factors such as:

- number of events (intrusions, exploits, and the like) successfully found in the test data (higher "intrusion detection rate" or increased "diagnosis rate")
- repeatability of results
- system performance measurements (execution time, memory use, disk space use)

  These will then be fed into the hybrid intelligence/multi-agent system

In addition to running the data sets through the various algorithms used in this paper, we also intend to run them through a MapReduce based data mining package. These results would also be fed into the "over layer" for assessment to provide feedback and suggestions to tweak the other algorithms.

Beyond that we intend to exercise the hybrid intelligence/multi-agent system to demonstrate the concept. Additionally, it would be nice to have an open enough system so that new and unforeseen data types could be accommodated. It is also desired to move away from systems that are strictly signature based and reactive, but instead provide a system that can serve in a predictive, proactive role.

6. REFERENCES

Aberer, K. (2012). *Week 14 Datamining-Clustering-Classification-Wrap-up.pdf.* Retrieved from Distributed Information Systems Laboratory: http://lsirwww.epfl.ch/courses/dis/2007ws/lecture/week%2014%20Datamining-Clustering-Classification-Wrap-up.pdf

Basilyan, M. (2012). *Mike Basilyan*. Retrieved from http://mikebasilyan.wordpress.com/2010/02/02/weka-mysql-on-ubuntu/

DeLoach, Oyenan, & Matson. (2008). A capabilities-based model for adaptive organizations. *Journal of Autonomous Agents and Multiagent Systems Volume 16, no. 1*, 13-56.

Gantz, J., Boyd, A., Dowling, S. (2009). Cutting the Clutter: Tackling Information Overload at the Source. *Xerox White Paper*

Grosan, C., Abraham, A., & Chis, M. (2006). Swarm Intelligence in Data Mining. *Studies in Computational Intelligence*, pp. 1-20.

Habib, M., Sallam, A. E.-H., & Badawy, O. (2008). Quantitative Association Rule Mining Using a Hybrid PSO/ACO Algorithm (PSO/ACO-AR). *Arab Conference on Information Technology Proceedings.*

Markey, & Atlasis. (2011). *"Using Decision Tree Analysis for Intrusion Detection: A How-to Guide" .* Global Information Assurance Certification Paper.

Moore, A. W., & Zuev, D. (2005). Internet Traffic Classification Using Bayesian Analysis Techniques. *ACM SIGMETRICS.* Banff.

OpenPacket. (2012). Retrieved from Open Packet: http://www.openpacket.org

Pfleeger, P. &. (2003). *Security in Computing - Third Edition.* Upper Saddle, NJ.

Ramirez, G., Caswell, B., Rathaus, N., & Beale, J. (2005). *Nessus, Snort, & Ethereal Power Tools.* Rockland, MA: Syngress.

Revathi, M., & Ramesh, T. (2011). Network intrusion detection system using reduced dimensionality. *Indian Journal of Computer Science and Engineering Vol 2, no 1*.

Settles, M., & Rylander, B. (2002). Neural network learning using particle swarm optimizers. *Advances in Information Science and Soft Computing*, pp. 224-226.

Suh, S. C. (2011). *Practical Applications of Data Mining.* Jones and Bartlett Learning.

TCPDUMP (2012, June 8). TCPDUMP/LIBPCAP public repository. Retrieved from: http://www.tcpdump.org/

Weka. (2011, April 8). *Weka 3 - Data Mining with Open Source Machine Learning Software in Java.* Retrieved from Weka: http://www.cs.waikato.ac.nz/~ml/weka/gui_explorer.html

Weka 2 (2012, July 14). *Attribute-Relation File Format (ARFF).* Retrieved from Weka: http://www.cs.waikato.ac.nz/ml/weka/arff.html

Wireshark.org. (2012, April 1). *tshark - The Wireshark Network Analyzer.* Retrieved from The Wireshark Network Analyzer:

http://www.wireshark.org/docs/man-pages/tshark.html

Wireshark.org.2 (2012, April 1). *Wireshark - Frequently Asked Questions.* Retrieved from The Wireshark Network Analyzer: http://www.wireshark.org/faq.html#q1.1

Zhang, & Zulkermine. (2006). Anomaly based network intrusion detection with unsupervised outlier detection. *Communications, ICC*.

Appendix A

Installing Weka on Windows

Installing Weka on Windows

Download the installation files appropriate for your version of Windows from here:

http://www.cs.waikato.ac.nz/ml/weka/index_downloading.html

Run the installer.

To prevent "out of memory" java errors (in Windows) modify the maxheap line in RunWeka.ini to something larger than the default (512m) in our case. We changed it to maxheap=2048m.

Appendix B

Weka data mining notes

Running the data through various algorithms in Weka.

After starting Weka, select Explorer from the Weka GUI chooser. This brings up the Weka Explorer, ensure that the preprocess tab is selected. Click "open file…" and open up the interesting *.csv file. For purposes of this iteration of our research, in the Attributes section, we selected "All" (No., Time, Source, Destination, Protocol and info).

Running the classifiers.

Select the Classify tab, click on the "Choose" button and select the J48 algorithm from the "trees" folder. Accept the default values appended (-C 0.25 –M 2). Ensure that the Cross Validation button in Test Options is selected and Folds is set to 10. Then click "Start." Go get a cup of coffee.

Select the Classify tab, click on the "Choose" button and select the UserClassifier algorithm from the "trees" folder. Ensure that the Cross Validation button in Test Options is selected and Folds is set to 10. Then click "Start." Go get a cup of coffee.

Running the clusterers.

Select the Cluster tab, clik on the "Choose" button and select Cobweb. Accept the default values appended (-A 1.0 –C 0.0028209479177387815 –S 42). Ensure that the "Use training set" radio button is selected (for this run) and click "Start." Go get a cup of coffee.

Appendix C

Weka Data Mining Results

(Complete results (4,341 pages) available upon request)

Classifying Algorithms

J48 results Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2

Relation: nmap-target.scan.pcap

Instances: 2013

Attributes: 6

No.

Time

Source

Destination

Protocol

Info

Test mode:10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree

---------- snip ----------

UserClassifier results

Scheme:weka.classifiers.trees.UserClassifier
Relation: nmap-target.scan.pcap
Instances: 2013
Attributes: 6
No.
Time
Source
Destination
Protocol
Info
Test mode:10-fold cross-validation
=== Classifier model (full training set) ===
: N0 45138 > x11 [SYN] Seq=0 Win=1024 Len=0 MSS=1460(2013.0/2002.0)
Time taken to build model: 7493.74 seconds

---------- snip ----------

Clustering Algorithms

Cobweb results

Scheme:weka.clusterers.Cobweb -A 1.0 -C 0.0028209479177387815 -S 42

Relation: nmap-target.scan.pcap

Instances: 2013

Attributes: 6

No.

Time

Source

Destination

Protocol

Info

Test mode:evaluate on training data

---------- snip ----------

SimpleKMean results

Scheme:weka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10
Relation: nmap-target.scan.pcap
Instances: 2013
Attributes: 6
No.
Time
Source
Destination
Protocol
Info
Test mode:evaluate on training data

---------- snip ----------

Associator Algorithms

Apriori results

Scheme: weka.associations.FilteredAssociator -F "weka.filters.MultiFilter -F \"weka.filters.unsupervised.attribute.ReplaceMissingValues \"" -c -1 -W weka.associations.Apriori -- -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Relation: nmap-target.scan.pcap.no.numbers2-weka.filters.unsupervised.attribute.Remove-R1,5
Instances: 2013
Attributes: 3
Source
Destination
Protocol
=== Associator model (full training set) ===

FilteredAssociator using weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1 on data filtered through weka.filters.MultiFilter -F "weka.filters.unsupervised.attribute.ReplaceMissingValues "

---------- snip ----------

ISSN: 2167-1508
v5 n2239

Appendix D
Larger Figures

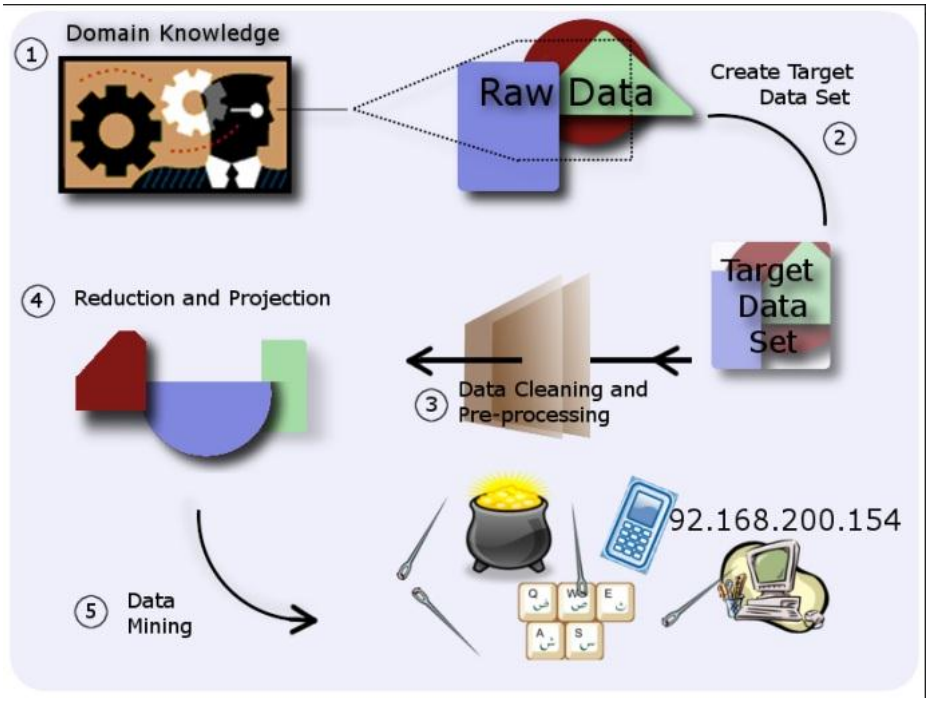

Figure1. Steps of the knowledge discovery process, adapted from (Grosan, Abraham, & Chis, 2006)

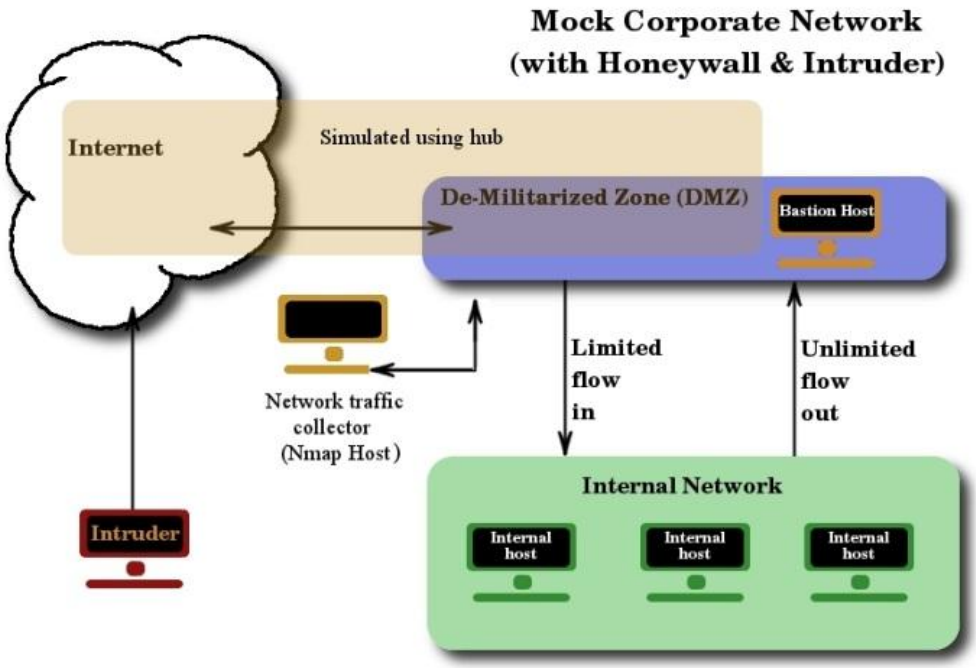

Figure 2. Lab network for generating seeded data

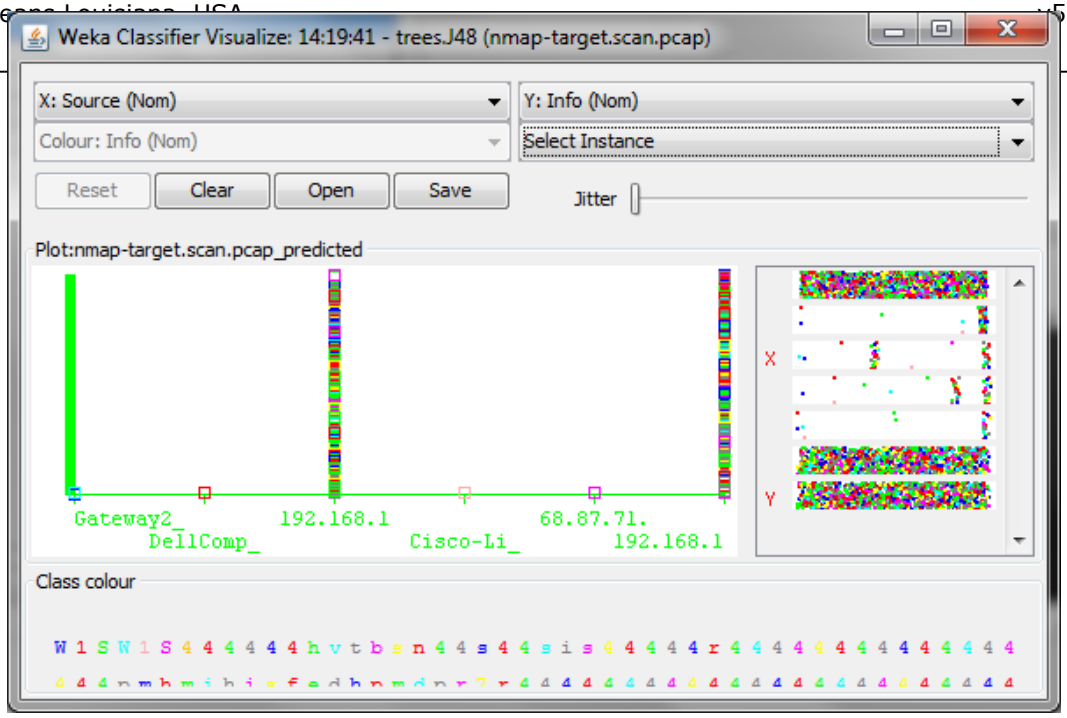

Figure 3. J48 Classifier results of nmap-target.scan

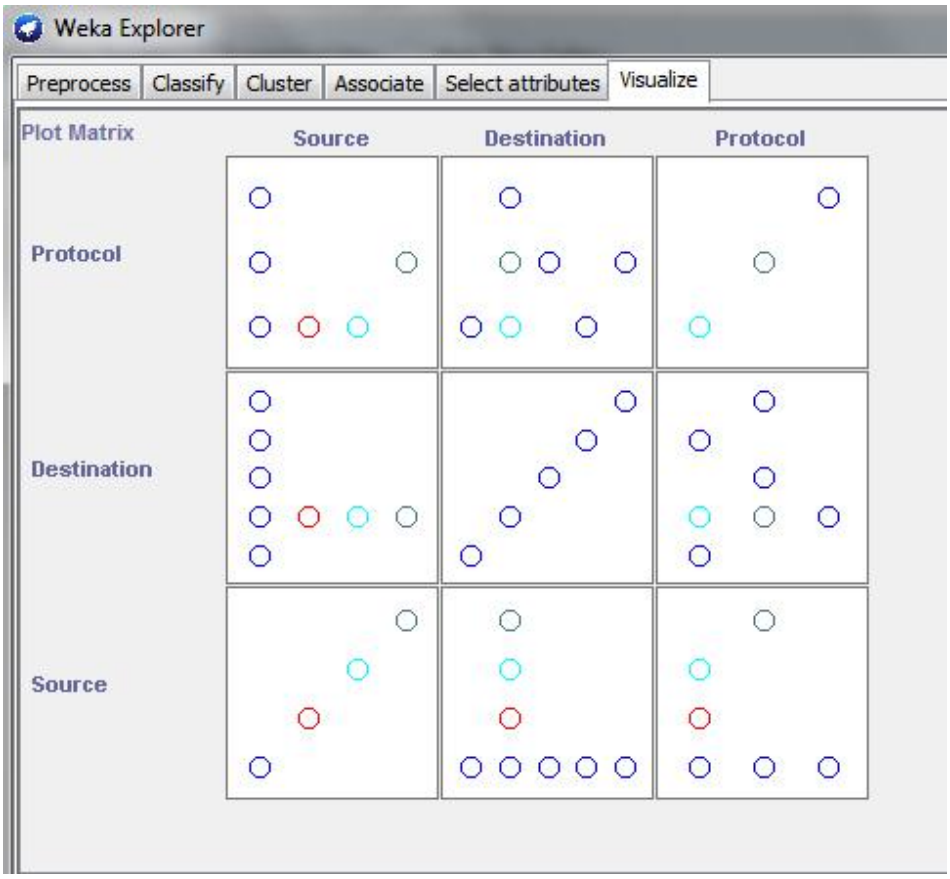

Figure 4. Associator - results of nmap-target.scan

Appendix E
Glossary

ARFF – "(Attribute-Relation File Format) file is an ASCII text file that describes a list of instances sharing a set of attributes." (Weka 2)

CSV – Comma Separated Values

PCAP - Packet Capture file format (TCPDUMP)

Tshark – "TShark is a network protocol analyzer" (Wireshark.org)

WEKA – "Weka is a collection of machine learning algorithms for data mining tasks." (Weka)

Wireshark – "Wireshark® is a network protocol analyzer. It lets you capture and interactively browse the traffic running on a computer network." (Wireshark.org.2)