IDDM: Intrusion Detection using Data Mining Techniques

Tamas Abraham

IDDM: Intrusion Detection using Data Mining Techniques

Tamas Abraham

Information Technology Division
Electronics and Surveillance Research Laboratory

DSTO-GD-0286

ABSTRACT

The IDDM project aims to determine the feasibility and effectiveness of data mining techniques in real-time intrusion detection and produce solutions for this purpose.

Traditionally, data mining is designed to operate on large off-line data sets. Previous attempts to apply the discipline in real-time environments met with varying success. In this paper, we overview earlier attempts to employ data mining principles in intrusion detection and present a possible system architecture for this purpose. As a consequence, we show that by combining data mining algorithms with agent technologies, near real-time operation may be attained.

20010720 076

RELEASE LIMITATION

Approved for public release

```
┌─────────────────────────────────┐
│ D E P A R T M E N T   O F   D E F E N C E │ DSTO
│ DEFENCE SCIENCE & TECHNOLOGY ORGANISATION │
└─────────────────────────────────┘
```

AQ FOI-10-2022

IDDM: Intrusion Detection using Data Mining Techniques

Executive Summary

The IDDM project investigates the potential use of the data mining paradigm in near real-time intrusion detection in order to develop techniques for the defence of computing networks.

To protect networks, intrusion detection systems aim to recognise attacks with two primary requirements: high detection and low false alarm rate. As attacks manifest themselves in two categories, those that are known and those that have not been seen previously, it is imperative that good descriptions of existing attacks as well as normal network behaviour are available. Data mining is recognised as a useful tool for extracting regularities in data and thus has been the target of some investigations for its use in intrusion detection. The IDDM project focuses on the use of data mining in the latter context, by producing descriptions of network data and using this information for deviation analysis. A number of existing technologies are available for this purpose, some of which are evaluated as part of the project.

As part of the investigative process, this report also details an architecture which is designed to accommodate the deviation analysis process. This process is performed by meta-mining techniques that characterise change between network data descriptions produced at different times. When detecting large deviations between descriptions, the system can produce appropriate alarm notices.

The outcomes of the IDDM project are hence the abilities to characterise network data and to detect variations in these characteristics over time. Combining this capability with tools that either recognise existing attack patterns or operate similarly to IDDM, it strengthens the ability of intrusion detection professionals to recognise and potentially react to unwanted violations to network operations.

Authors

Dr Tamas Abraham
Information Technology Division

Tamas Abraham is a research scientist in the Advanced Computer Capabilities branch in the DSTO. He is a researcher in the Crashcart project. Dr Abraham holds a Ph.D. in Computer Science from the University of South Australia, Adelaide.

Contents

1. Introduction

As global networking becomes more widespread, the number of unwanted violations to its normal operations are on the increase, be that for personal, commercial or military advantage. Intrusion detection, a rapidly maturing field, deals with the real time detection of such activities. It attempts to identify existing attack patterns and recognise new intrusion methods, employing methods from sciences such as mathematics, statistics and machine learning. Data mining, generally perceived to be a tool to discover unknown regularities in data, also lends itself to this task. In particular, it promises to help in the detection of previously unseen attacks by establishing sets of commonly observed regularities in network data. These sets can be compared to current traffic for deviation analysis. Data mining techniques, however, are traditionally employed on large amounts of off-line data. It therefore remains to be seen how well they are able to support ID systems commonly required to operate in real time.

The object of this paper is to investigate the feasibility and effectiveness of a variety of data mining techniques within intrusion detection. Without aiming to be complete, we attempt to look at a range of knowledge discovery methods and their potential use in detecting intrusions. The paper is sub-divided as follows. The remainder of this section overviews existing solutions that combine the two paradigms. Section 2 introduces our project and details some well-known data mining techniques and their potential contribution to intrusion detection. Section 3 discusses an architecture implementing some of the ideas presented in Section 2, whilst taking advantage of existing available infrastructure. Section 4 details the implementation of a particular technique and its place within the project. Section 5 summarises results of experiments performed to test our meta-mining technique. Section 6 presents some brief conclusions whilst the Appendix shows an overview of existing algorithms.

1.1 Previous Related Work

Some interest has already been expressed in data mining as an aid to intrusion detection. There are, however, differences in the approaches taken in the use of mining techniques. The two examples presented below illustrate these differences and provide an insight into the potentially diverse application of data mining for intrusion detection purposes.

1.1.1 Columbia University

A pioneering research activity that emphasises data mining as the leading paradigm in intrusion detection was MADAM ID, part of the larger JAM Project at the Computer Science Department of Columbia University, lead by Salvatore Stolfo [1].

The original idea behind JAM (Java Agents for Meta-Learning) was to use data mining techniques to correlate knowledge derived from separate, heterogeneous data sets into a rule-set capable of providing a general description of an environment comprising these sets. This work lead to the further use of data mining techniques to build better models for intrusion detection by analysing audit data using associations

and frequent episodes, and utilising the resulting rules when constructing ID classifiers. The results from this effort are collectively labelled as MADAM ID (Mining Audit Data for Automated Models for Intrusion Detection). The project has received a number of distinctions from peers, such as achieving high scores at the DARPA 1998 Offline Intrusion Detection Evaluation and being awarded Best Paper in Applied Research at KDD 99 [2].

As an extension to the above achievements, Columbia University is currently coordinating a project to create a complete intrusion detection system operating real-time. The project, tentatively named Project IDS, is in its initial stages with further details available on the Internet [3]. Below is a short list of some of their goals:

❑ *Real-time* operation of the IDS and adaptive anomaly training.
❑ *Coverage* of entire attack space including both *anomaly* and *misuse* detection and *explanation* of attacks.
❑ *Automatic feature construction* from data collected from *multiple sources*.
❑ *Cost based algorithms* to decide if attacks are worth handling.
❑ *Link analysis* to detect complex attacks possibly spread over space and time.

The accompanying intrusion detection system architecture is heavily modularised. A number of components interoperate to make deductions on network and system attacks. Accordingly, the implementation is divided into sub-projects:

❑ HoBIDS – Host Based IDS which monitors specific hosts by analysing binaries and process logs (for example, BSM logs on Solaris machines). It uses *rules* supplied by rule generating components to detect attacks in a cost sensitive manner. It supplies information to a data warehouse for off-line analysis.
❑ HAUNT – A network based IDS analysing packets and detecting attacks based on *rules* specified in N-code of the NFR system.
❑ AMG – Adaptive Model Generation builds and updates a model of the behaviour of a system in real time using sensor data. It uses a probability-based *data mining* algorithm in order to do so. Both models and formatted sensor data are stored in a data warehouse, with XML representation of data and models.
❑ DIDS – The Distributed IDS System coordinates the activity of host and network based IDS and based on reports received makes decisions on attacks and generates alarms. It applies *activity association* (for multiple reports arriving at the same time) and *link analysis* (for temporally spread reports) to information received from other components. It also contains an *explanation engine* to produce natural language statements about observed attacks.
❑ MEF – The Malicious program E-mail Filter monitors e-mail attachments and stops viruses from spreading when one is found. It employs *machine learning* and *data mining* techniques for detection, and uses an *adaptive* generalisation model to discover malicious attachments and contains a reporting module for when one is found.
❑ DWARF – Data Warehousing for IDS centralises data collected from IDS systems to facilitate the learning of models for intrusion detection. Several components in Project IDS utilise and add to the contents of the data warehouse.
❑ FWRAP – File System Wrappers monitor writes to the file system to detect attacks.
❑ ASIDS – The Advanced Sensors Project supplies data to a data warehouse for learning.

❑ IDSMODELS – A collection of algorithms to be employed in different components of the IDS. They include techniques such as *Sparse Markov Trees, probabilistic graphs* for sequences of system calls, *clustering, probabilistic anomaly detection, classification, boosting, link analysis, co-training, frequent episodes* and *associations*.

1.1.2 Iowa State University

A different approach is taken by Iowa State University in their design of an intrusion detection system. Data mining techniques are incorporated into an agent based architecture at high level in order to correlate information collected by low-level agents [4]. The low-level agents are designed to propagate information upwards as well as sharing information on their designated level. The expected benefits of this design is the potential to recognise attacks spread over both space and time, localise their origin and co-ordinate a suitable response. The system also illustrates a successful combination of agent technologies with data mining.

At least some of the work in future intrusion detection approaches is likely to proceed in a direction similar to that demonstrated by the works above. Data mining hence may become a prominent component in large-scale IDS warranting continued examination of other, potentially useful techniques.

2. Using Data Mining in Intrusion Detection: The IDDM Project

The success of earlier attempts of the use of data mining in intrusion detection, as outlined in the previous section, suggests that a review of available techniques can help in identifying additional ones worth investigating.

2.1 Project Summary

The IDDM project (Intrusion Detection using Data Mining) aims to explore data mining as a supporting paradigm in extending intrusion detection capabilities. Our interest is to *re-use, augment and expand* on previous works as required and introduce new principles from data mining that are considered good candidates for this purpose. Rather than concentrating on the use of a particular technique in a certain application instance, we intend to explore multiple uses for any given data mining principle in a variety of ways.

2.1.1 Fitting into Existing Frameworks

The IDDM project is part of Crashcart, an internally developed ID toolkit at DSTO [5]. It can therefore both rely on its front-end data gathering functionality and needs to retain interoperability with this toolkit. It may also be tied in with other Defence projects designed for protecting network security, particularly Shapes Vector [6]. This should provide additional tools already available in those projects for use within IDDM (for example, alarms, visualisation paradigms and other analytical features). There is also a possible requirement on IDDM outcomes to provide interaction with these external projects.

2.1.2 General Goals

As data mining normally excels at observing general regularities, the predominant goal of the IDDM project is that of anomaly detection in network data at a relatively high level, and the delegation of suspected problems to dedicated applications. The expected benefits of such an approach are an increased ability to include large distributed data sets in the detection process. In addition, a reduction in false alarms is more likely as a result of delegating anomalies to more specialised low-level detection applications.

2.1.3 Process

The initial stages of IDDM include the building of a data mining library that is a subset of algorithms already in use for intrusion detection. Each algorithm is tested and finetuned as it is developed, and fit into the larger IDDM model.

2.2 IDDM Details

Part of the purpose of this document is to overview existing techniques that may be able to be employed in the mining of network data to detect intrusions. This section presents some of the major techniques currently in use and discusses platforms for their application.

2.2.1 Techniques

A number of data mining techniques can be used in intrusion detection, each with its own particular advantage. The following categorisation lists some of the techniques and the purposes for which they may be employed. In later sections, further discussion is provided for techniques investigated within the IDDM framework.

- Characterisation/Generalisation – Produces a general description of the nature of the data. It can be used for deviation analysis if adequate difference is present in data content.
- Classification – Creates a categorisation of data records. It could be used to detect individual attacks, but as described by previous experiments in the literature, it is prone to produce a high false alarm rate. This problem may be alleviated by applying fine-tuning techniques such as boosting [7].
- Association - Describes relationships within data records. Detection of irregularities may occur when many records exhibit previously unseen relationships.
- Frequent Episodes - Describes relationships in the data stream by recognising records that occur together. For example, an attack may produce a very typical sequence of records (similar to system call traces used in the Computer Immunology project [8]). This technique may produce results for distributed attacks or attacks with arbitrary noise inserted within.
- Clustering – Groups records that exhibit similar characteristics according to some pre-defined metrics. It can be used for general analysis similar to classification, or for detecting outliers that may or may not represent attacks.
- Incremental updates - Keeps rule sets up-to-date to better reflect the current regularities in the data.

❑ Meta-rules – Provides a description of changes within rule sets over time. It can be used for trend analysis. It is probably not very useful for detecting individual attacks, but can serve as a base of comparison when new, unusual rules appear. For example, if a trend shows a steady increase in some network activity over time, then a sudden increase in another type of activity may be suspicious, but not necessarily an increase in the former.

2.2.2 Data Elements To Mine

The major contributor to the development of successful mining techniques is the clear definition of the network data features that will be used in the creation of initial rule sets. It is possible to add to or remove from a feature set later, and most techniques will correctly operate with these modifications. However, it is likely that the rule sets resulting from these changes will be different to those produced earlier, and this will affect processes performing trend analysis.

There are different levels at which feature sets may be defined:

❑ Packet - Interesting details at this level are characteristics such as service type, addresses, any special flags and so on.
❑ Connection - Some of the same characteristics apply at this level as for packets, with new ones being added such as duration, amount of data transferred and so on.
❑ Connection content – The set of features required at this level are currently not clearly defined, as relatively few mining approaches, such as text analysis, are candidates for analysis at this level. The most likely data to be useful from connection content is header information from Internet protocols for web browsing or e-mail.

2.3 Association Rules for IDDM

Associations rule mining is one of the most popular techniques within data mining and thus has been one of the first methods to be trialed in intrusion detection. Both university projects mentioned in Section 1 have investigated and successfully utilised it, and it features in our IDDM project. Its use in IDDM differs from previous attempts in that it is mainly employed as a tool to generate observations about network traffic for further processing, both at packet and at connection level. That is, association mining acts as a sensor supplying rules as source data for higher level processing, such as meta-mining. Further details on this role as a data gatherer can be found in Section 4, where the association mining algorithm is incorporated into an agent.

2.3.1 Technique

The association mining algorithm, as originally described in [9], finds relationships within data in the form

$$A \Rightarrow B \ (s\%, c\%),$$

where A and B are sets of (binary) attributes occurring together in a database. The percentages $s\%$ and $c\%$ express support and confidence for a rule described with the

above formula. More precisely, the support of a rule is the percentage of records containing both A and B, while the confidence of a rule is the percentage of records containing A that also contain B. Usually, a pre-existing data set may be converted into another containing only binary attributes so that association mining can proceed.

Both support and confidence are user-defined variables that may be dynamically adjusted during mining in order to obtain a desired final rule set. Association mining, however, has its drawbacks:

❑ It works best when the number of attributes are large and records are sparsely populated, that is, only a few attributes are set within each record. In the unusual scenario where most or all attributes are set in every record, the final rule set may contain far more rules than there are records.

❑ No rules are generated for attributes that are not frequent enough, that is, for those that do not reach the desired support.

❑ It is often difficult to achieve a balance between the number of final rules and their importance. Setting the required support and/or confidence levels high will result in a smaller number of rules, but they are also likely to be ones that seem too obvious to be useful observations. Lowering the percentages increases the chance of finding more interesting rules, whilst also increasing the number of rules, thus making it more difficult to analyse the final rule set.

2.4 Meta-Rules for IDDM

Meta mining is a concept that derives rules from several rule sets collected over a period [10]. Each rule set is treated as a snapshot description of the data at a given time. A meta-rule set relates any two given sets by describing rules that expired, changed, remained unchanged or appeared new. In theory, it is possible to create meta-rules of any type on any existing rule set types with some restrictions. Thus, association meta-characteristic rules are just as possible as a combination of meta-meta-rules at an arbitrary level of abstraction. In the IDDM project, meta-tests were performed on association rules generated on network packets captured at separate time intervals. The results are outlined in Section 5.

2.5 Characteristic Rules for IDDM

The purpose of characteristic rules is to provide a general description of a data stream in order to facilitate evaluation of changes over time. Within the IDDM project, they are used for monitoring changes over time, in fashion largely similar to the use of association rules.

2.5.1 Technique

The original data characterisation technique, as described by [11], has three main aspects:

❑ Removal of unusable attributes
❑ Concept ascension
❑ Data compacting

The first of these aspects is not relevant within the IDDM framework, as rule generation is preceded with a transformation of net data into a relational database. This means that no unnecessary/uncompactable attributes are stored. Usually, attributes are removed only if they contain too many values that cannot be replaced by higher level concepts, such as addresses. After transformation, every record in the database represents a rule in itself with a support count of one for the rule.

Concept ascension is the replacement of attribute values with higher level concepts. They are commonly controlled by thresholds, for example, the maximum number of concepts allowed for a given attribute. The high and low level concepts are organised into hierarchies, which are usually designed by experts possessing substantial domain knowledge. Alternatively, concept hierarchies may be generated by automatic means to avoid this often expensive and time-consuming step [12].

Data compacting means the deletion of duplicate entries from the relational table. This includes an initial parsing of the database to remove duplicates from the transformation of raw data into the database. As attribute values are replaced with concepts, more and more records will contain the same attribute values. When a particular duplicate is removed, a count for the original is increased to express the support the given concept ascended record has. Concept ascension and data compression continues until the original rule set is reduced to a desirable size.

Some of the thresholds that can be useful for characterisation are:

❑ maximum number of concepts allowed for each attribute
❑ maximum number of generalised records in the final table
❑ a percentage above which an attribute value is not ascended when the rest is (for example, if 5% of port numbers are a certain value, we may retain this value while replacing other, less significant values with higher concepts)

2.5.2 Benefits

Applying characterisation to a data set has obvious benefits from the intrusion detection viewpoint. However, some of the benefits come at a price. Below is a short description of some of the advantages and disadvantages of employing characteristic rules within an ID project:

Advantages
❑ Rules are high level, that is, the final descriptions are easily understandable.
❑ Employs concept hierarchies that are user manageable.
❑ Reduces data space to desired compactness without loss (as opposed to association rules where a minimum support needs to be met, otherwise some attribute values are not included in the rule generation process).

Disadvantages
❑ Strictly off-line, it needs a suitably large initial data set and the algorithm is relatively complex and may hence be slow.
❑ Needs a relational database to store transformed data stream information.
❑ Not suited to pick up specific events in the data unless the concept hierarchies and algorithm allow for it.
❑ Not suited for recognising specific new attacks (see previous point).

2.5.3 Example rule sets

To provide a basis for comparison with association rules, a simple characteristic rule mining algorithm has been implemented in the IDDM project. In contrast to associations as discussed in later sections, this algorithm does not yet form an integral part of the project. Figure 1 below illustrates some of the rules that can currently be produced. Tests were performed on several smaller sets of data generated on a local network with machines simulating use of some of the more often used network protocols. The example below contains mainly UDP and Web traffic. Note the small number of rules produced in the final set. Each rule is expressed as a combination of packet type (ICMP, TCP, UDT and so on), a size concept, and source and destination port concepts where applicable. Certain well-known port numbers are placed high into the conceptual hierarchies in order to be keep their values as long as possible in the rules. For example, port 8080, designated as PROXY, did not get replaced by a higher level concept while the other port numbers have been generalised into the concept OTHER (describing high port numbers above 1024). As there were less than 10 different port numbers for UDP traffic, they did not get replaced at all, which shows in the final rule set.

```
— Rule generation started
DDD Date: Tue Jun  6 09:51:08 2000
RRR Recordcount: 1018
MMM Maximum_Concepts_Used: 10
    I     60                          ( 0.002947 or    3/   1018)
    O     TINY                        ( 0.133595 or  136/   1018)
    O     SMALL                       ( 0.076621 or   78/   1018)
    T     SMALL PROXY OTHER           ( 0.049116 or   50/   1018)
    T     TINY    OTHER PROXY         ( 0.182711 or  186/   1018)
    T     TINY    PROXY OTHER         ( 0.084479 or   86/   1018)
    T     ETHER PROXY OTHER           ( 0.125737 or  128/   1018)
    T     SMALL OTHER PROXY           ( 0.014735 or   15/   1018)
    U     TINY   43425  3000          ( 0.270138 or  275/   1018)
    U     SMALL 43425   3000          ( 0.020629 or   21/   1018)
    U     SMALL 138     138           ( 0.004912 or    5/   1018)
    U     TINY   137    137           ( 0.020629 or   21/   1018)
    U     TINY   799    2049          ( 0.000982 or    1/   1018)
    U     TINY   2049   799           ( 0.000982 or    1/   1018)
    U     TINY   138    138           ( 0.009823 or   10/   1018)
    U     TINY   1025   111           ( 0.000982 or    1/   1018)
    U     TINY   4496   1604          ( 0.000982 or    1/   1018)
*** Rule generation completed
```

Figure 1: Characteristic network rules

2.6 Clustering for IDDM

Clustering is another possible technique that may yield results in the intrusion detection context. As more and more intrusions are now spread over time and space, these techniques may be able to discover such attempts by correlating seemingly independent network activities. Clustering groups data according to their "similarity", and this can be used to identify outliers not fitting into expected

patterns (new attack types) or existing attacks previously identified to belong to particular groups.

An existing example for the use of clustering for intrusion detection purposes can be found in [13]. The paper presents an implementation of a one-pass clustering technique that classifies normalised network connection data into clusters according to a distance metric. Working on the assumptions that data with the same classification are close to each other according to this metric, and that attacks constitute only a small percentage of the data, it then labels clusters with relatively small number of elements as anomalous. The algorithm was tested on the KDD99 intrusion data set [14] and was moderately successful in detecting intrusions while keeping the false alarm rate down. Although the results are not outstanding, the advantages of this technique are obvious: it does not require prior knowledge of data contents as long as the assumptions above are satisfied. It is also capable of retraining itself on-the-fly by creating new clusters as they appear and potentially re-classifying existing ones as their relative membership increases/decreases. An additional suggested use of this technique is to employ it as a sensor in a larger ID system. Possible intrusions can be reported as they happen to a higher level decision making module for handling. This module may receive information from other sensors and can treat alarms from the clustering algorithm as only one of several sources.

3. An IDDM Architecture

This section discusses an architecture that has a number of desired properties to satisfy the requirements for a modern intrusion detection tool. The ideas presented here are not unlike those promoted by systems described in Section 1, and include the following attributes:

- The architecture makes use of the data mining paradigm to help in the detection of intrusions.
- It facilitates operation in real-time or near-real time.
- It maintains a central data and rule storage warehouse.
- It is adaptive, that is, it constantly updates its rule sets to better fit the current environment.
- It interoperates with and incorporates existing tools available internally.

Some of above specifications require further discussion, which is provided in the remainder of the section. Figure 2 below contains a possible architectural design satisfying these conditions.

3.1 Details of the Architectural Design

As outlined earlier, intrusion detection concentrates on recognising two types of possible attacks, those that have been encountered before, and previously unseen ones. Although other detection tools employing data mining techniques often attempt to capture both kinds of intrusions, the IDDM project focuses more on recognising new attack types, or anomalies as they are often referred to. This is more attune to the spirit of data mining emphasising the discovery of previously unseen regularities in large data sets. A great number of tools already exist that cater for the detection of existing attacks with varying degrees of success, and one or more of

these tools can interoperate with IDDM outcomes to cover the full range of attack scenarios.
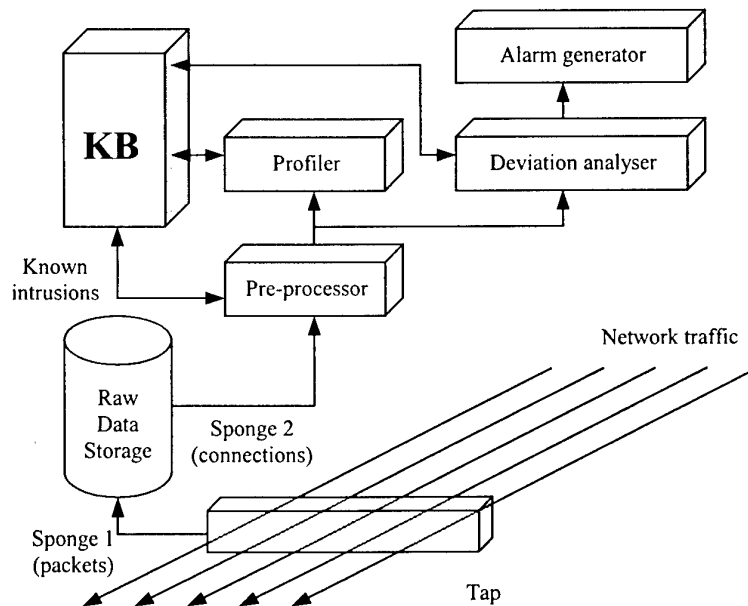

Figure 2: A possible IDDM architecture

The components of the architecture of Figure 2 are designed to satisfy the requirements of the IDDM project. They are as follows:

□ The *Tap* is where network data is collected. It is simply an interface capable of capturing information flowing by (such as a network card on a machine). The location of the tap determines the localisation of intrusion detection. For example, anomalies may be observed on a single machine, a network segment, or a gateway. For our IDDM project , the information gathering at the tap can be done with the aid of an existing Crashcart application (Sponge).

□ The *Raw Data Storage* stores collected network data. Typically, it is a set of hard drives where an application dumps information passing through the tap, usually according to some filtering requirements.

□ The *Pre-processor* handles the conversion of raw packet or connection data into a format that mining algorithms can utilise and may store the results in the knowledge base. It can perform a range of duties, such as additional filtering, noise elimination, and include third party detection tools that recognise known attack patterns.

□ The *Knowledge Base* stores rules produced by mining and any additional information used in the mining process. It may also hold information for the pre-processor, such as patterns for recognising attacks and conversion templates.

□ The *Profiler* is responsible for generating snapshot rule sets to be used for deviation analysis. It can be triggered automatically based on time of day or the amount of pre-processed data available.

❑ The *Deviation Analyser* examines rule sets in the knowledge base and creates a description of differences by meta-learning. The results are stored in the knowledge base for further reference. If necessary, it signals the alarm generator. A strategy for invoking the deviation analyser could be periodic queries to the knowledge base for the availability of new profiles. Alternatively, the profiler may signal the analyser when a new profile is deposited to the knowledge base.

❑ The *Alarm Generator* is responsible for notifying the administrator when the deviation analyser reports unusual behaviour in the network stream. This can take the form of e-mails, console alerts, log entries, or even be a component of a visualisation tool.

3.2 Considerations

One of the benefits of our approach is that deployment of the system does not require a-priori background knowledge. Data collection and rule generation commences on live data at the time of installation and trends may be observed as early as required. It is likely that in the initial operating stages there will be a larger than usual number of false alarms, until a more stable description of normal network data becomes available. Note that if the individual machine/network segment to be protected is under attack during installation, then alarms will be triggered when the environment resumes its normal state. It is therefore important to investigate the cause of even these early alarms.

A requirement on the architecture is to allow for real or near real-time operation. This can be the case for detecting known attacks by third party packages. Data can be pushed through to the pre-processor stage in real-time. Profiling (and therefore, deviation analysis), however, requires a certain amount of input data to form a description. The rule base is continually evolving – it is adaptive - to reflect the current behaviour of the system being monitored and is flexible to adjustments (for example, the user may be able to force an intermediate profiling step). Nevertheless, it may take some time before an appropriately large set of pre-processed data is available and profiling can be triggered. Therefore, trend analysis performed under the IDDM project is best described as near real-time.

The alarm generator, whilst performing its purpose, is only one tool of possibly many that relates information to the user. It may be necessary to allow inspection of the knowledge base contents to gain further insight into the rule collection. As rules are often cryptic, this not only means the need for a good visualiser but potentially a translator component to easily interpret the knowledge base contents. It may also be desirable to receive regular updates on the current state of the monitored environment, such as changes that did not warrant alarms, and reports on evolving trends.

A sensitive issue within any system monitoring trends is finding the correct balance between updates. In our architecture, updates are suggested to be performed based on a combination of time of day and input data set size. Using an initial periodic cycle and a fixed size, these parameters may be adjusted automatically or by user intervention, based on, for example, system performance and fluctuations in rule set stability. Automation forms an important part of IDDM. Generally, the higher the degree of automation the system achieves, the less the need for supervision by experts or interaction with the user. This is usually desirable, because of, for

example, its cost-cutting properties. On the other hand, full automation removes the human element from the analysis, which at the current state of software intelligence is possibly ill advised.

The architecture of Figure 2 describes componentry catering for a single data source. It is conceivable to expand it to include several sources, for example, by inserting several taps with supporting components up to the profiler level. Deviation analysis may be performed by a single component or multiple ones (but not necessarily as many as there are taps), whilst keeping a single knowledge base for storage. A model like this could possibly enable detection of multi-system distributed attacks, but likely to require additional higher level components to co-ordinate the operation and analyse the outputs of current analytical components.

4. Agent Tools

The architectural design of Section 3 allows for the periodic invocation of programs that analyse bulk sets of data as they become available. These programs are well suited to take the form of agents whose functionality is to extract higher level knowledge from (possibly pre-filtered and formatted) raw data. A common characteristic of these programs is the ability to produce a variety of different outputs according to user-controlled thresholds. As they can be put into operation immediately when an appropriate amount of data is obtained, results are also available near real-time. From an intrusion detection perspective, this functionality is desirable, because in combination with other tools it may either help to detect new attacks or prevent the escalation of attacks in progress.

4.1 Mining Network Packets

The network packet agent is a program based on the association rule mining paradigm. Its aim is to profile raw network packets by observing commonly occurring associations within individual packets. This information can then be used by further analytical programs (such as meta-miners) to detect unusual behaviour in the network stream. It needs to be reiterated that association mining is not well suited to identify infrequently occurring patterns, that is, patterns with low support. The control parameters of the mining process therefore play an important role on the amount of data that will not be influencing the resulting rules.

4.1.1 Packet mining attributes

The packet-mining algorithm concentrates mainly on the following packet attributes:

- Packet Type (ARP, ICMP, UDP, TCP, etc)
- Source and Destination Ports
- Packet Size
- TCP Flags

Both packet sizes and port numbers can have a large number of values that are not practical to be implemented as separate entities. Instead, categories are introduced to allow a range of values to be characterised by a single description. For example, packet sizes are currently split into five categories, ranging from zero to the

14

maximum size attainable on an Ethernet network. Category boundaries can be set by domain experts, or by analysing the network stream over time for suitable values. For port numbers, some often seen values are implicitly recorded while others are put into two additional categories, one for reserved port numbers and another for the rest.

4.2 Mining Full TCP Connections

The full TCP connection mining process is a variation of the packet mining technique based on association mining as described above. Instead of analysing raw network packets, rules are extracted to describe TCP connections. The difference between the two processes is that full connection mining requires an intermediate step, namely the generation of summary data about each connection.

4.2.1 Connection Mining Attributes

The connection mining routine will relinquish some of the attributes required for packet mining and add new ones pertinent to TCP connections:

- Packet Type is always TCP (no attributes needed)
- Source and Destination Ports
- Connection Duration
- Amount of Data Transmitted each way
- Ratio of Traffic (Incoming/Outgoing)
- Average Payload Length (not including headers)
- Unusual TCP Flags Used

Again, some of the attributes are best categorised as too many individual values exist to be representable for the mining algorithm.

If some of the content, (typically, header information for specific protocols), is also available, additional attributes can be added after investigating the text data contained therein. A possible set of additional attributes may include for these example protocols:

1) HTTP
   a) User Agent (Mozilla or Other)
   b) Operating System (Windows, Linux, Solaris and so on)
2) SMTP
   a) Agent
   b) Source/relay addresses

4.2.2 Technique Employed

Rather than operating on a raw network dump file, as is the case in our packet mining, the full connection mining agent uses summary information to perform its analysis. An algorithm is used to generate summary lines about each full TCP connection found in a dump file. Modified Crashcart components are used to perform this intermediate step. The resulting summary file is then parsed and processed in a similar fashion to the network packet mining tool.

4.3 Agent Operation

The agents are implemented as a set of three programs, two of which perform data mining on their input while the third creates a summary connection information file. Output is written to standard output in each case. The syntax of these programs is as follows:

<div align="center">

pcap2logic [OPTIONS] <input-file>

connectinfo <network data file>

connect2logic [OPTIONS] <input-file>

</div>

Connectinfo has the simplest syntax of the three as it performs a straightforward summarising operation. It parses a network data file, identifies full connections and generates summary information on each. The other two programs share their options list and take two different kinds of input files. For **pcap2logic**, this is either a network data file similar to the one accepted by **connectinfo**, or an itemset file produced by an earlier run of the program. For **connect2logic**, the two alternatives are a connection summary file output by **connectinfo** or an itemset file. The options available for both programs include:

❑ Parameters to control the association mining process, such as the rule confidence and support, and the maximum number of records to be read from the input file.
❑ Parameters to determine the format of the output which may vary from traditional association rules to Prolog or CLIPS logic representation.
❑ Two parameters to force an intermediate operation of the programs. Since association mining is a two-step process, consisting of itemset creation and rule generation, it is sometimes useful to be able to stop this process after the first step. This is often the case when several different types of final rule output are required. As itemset generation is by far the costlier of the two steps, it makes sense to be able to store the intermediate itemset results and only repeat the simpler rule generation several times to obtain various outputs.

Figure 3 shows some of the output produced by the agents. This particular example was produced by association mining some 20,000 network connections containing a fair number of attacks. They included host and port scans, where connections were attempted to a large number of different addresses and port numbers in the hope of finding available servers. This is primarily responsible for rules containing descriptions such as Zero-size (no data has been transmitted) or Zero-Dur (the connection attempt was not responded to). The numbers in brackets behind each rule express confidence and support percentages for each rule.

...
Tcp-Reset Src-Other -> Tcp-NoFin (0.9675, 0.2811)
Tcp-Reset -> Tcp-NoFin Dest-Well-Known (0.8620, 0.2515)
Tcp-Reset Tcp-NoFin -> Dest-Well-Known (0.8924, 0.2515)
Tcp-Reset Dest-Well-Known -> Tcp-NoFin (1.0000, 0.2515)
Tcp-NoFin Dest-Well-Known -> Tcp-Reset (0.5253, 0.2515)
Tcp-Reset -> Src-Other Dest-Well-Known (0.8618, 0.2514)
Tcp-Reset Src-Other -> Dest-Well-Known (0.8653, 0.2514)
Tcp-Reset Dest-Well-Known -> Src-Other (0.9998, 0.2514)
Dest-Ftp -> Tcp-NoFin Src-Other (0.7156, 0.1982)
Tcp-NoFin Dest-Ftp -> Src-Other (1.0000, 0.1982)
Src-Other Dest-Ftp -> Tcp-NoFin (0.7156, 0.1982)
Tcp-NoFin -> Src-Other Dest-Well-Known (0.6897, 0.4782)
Src-Other -> Tcp-NoFin Dest-Well-Known (0.5472, 0.4782)
Dest-Well-Known -> Tcp-NoFin Src-Other (0.8678, 0.4782)
Tcp-NoFin Src-Other -> Dest-Well-Known (0.6916, 0.4782)
Tcp-NoFin Dest-Well-Known -> Src-Other (0.9988, 0.4782)
Src-Other Dest-Well-Known -> Tcp-NoFin (0.8688, 0.4782)
Zero-Dur -> Zero-Size Tcp-NoAck Tcp-NoFin (0.9961, 0.1871)
Zero-Dur Zero-Size -> Tcp-NoAck Tcp-NoFin (1.0000, 0.1871)
Zero-Dur Tcp-NoAck -> Zero-Size Tcp-NoFin (0.9961, 0.1871)
Zero-Dur Tcp-NoFin -> Zero-Size Tcp-NoAck (0.9961, 0.1871)
Zero-Dur Zero-Size Tcp-NoAck -> Tcp-NoFin (1.0000, 0.1871)
Zero-Dur Zero-Size Tcp-NoFin -> Tcp-NoAck (1.0000, 0.1871)
Zero-Dur Tcp-NoAck Tcp-NoFin -> Zero-Size (0.9961, 0.1871)
Zero-Dur -> Zero-Size Tcp-NoAck Src-Other (0.9941, 0.1867)
...

Figure 3: Network connection rules

5. Meta-Mining Observations

This section presents the results of some of the IDDM experiments performed using the association and meta-mining paradigms. First, there is short review of meta-mining concepts from an earlier paper [10] and a description of the expected behaviour of the test data. This is followed by detailing each test in its own sub-section with comments regarding the outcome.

5.1 Meta-mining process overview

The basic meta-mining principle requires two existing rulesets to be separated into four new rule categories holding old, new, changed and unchanged rules from rule set one to rule set two. This is done in the following way: Some elements from the structure of a rule are appointed to be 'self-defining', whilst others may change without destroying the essence of a rule. For association rules, the support and confidence percentages may be appointed as non-defining. That is, an association rule is said to be unchanged from one rule set to another if it retains its structure A $\Rightarrow$ B defined in Section 2.3.1, where both A and B contain the same set of attributes in both rule sets. Change happens when a non-defining element of a rule exhibits deviation larger than some pre-defined threshold. Rules become old when they no longer appear in the second rule set with the same structure. New rules can appear in the second rule set when they have structure that is not found in the first. The four

categories holding these old, new, changed and unchanged rules form the meta-ruleset base. The categories can be further mined individually as a data source for higher level meta-rules describing regularities in each different category. The resulting rules inherently express differences in separately compiled rule sets and therefore are indicators of trends.
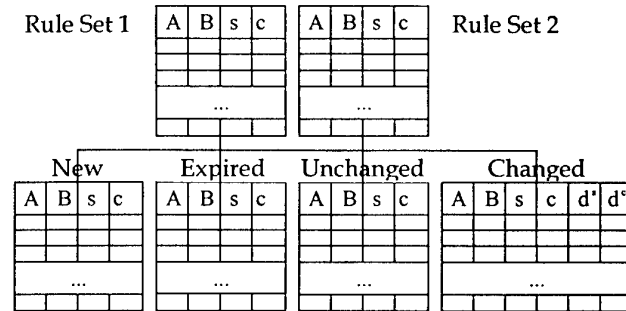

Figure 4: Example association rule sets and meta-ruleset base

Some invariant observations that can be made from Figure 4 are:

- The total number of rules T in a meta-ruleset base M is $T = O + N + U + C$, with the letters O, N, U and C denoting the number of elements in the old, new, unchanged and changed categories.

- Suppose a meta-ruleset base M_i is constructed from rulesets R_i and R_{i+1}. Then, the number of elements S_i and S_{i+1} in these rulesets can be expressed as totals of the elements of some of the meta-ruleset base categories, $S_i = O_i + U_i + C_I$ and $S_{i+1} = N_i + U_i + C_i$.

- Since S_{i+1} can also be expressed as $S_{i+1} = O_{i+1} + U_{i+1} + C_{i+1}$ from T_{i+1}, the total number of rules in consequent meta-ruleset bases can be expressed $T_{i+1} = T_i + N_{i+1} - O_i$. This holds assuming the same starting rule set R_{i+1} is used with some new R_{i+2} to form M_{i+1}.

5.2 Meta-ruleset base expectations

In a stable network stream, when mining two snapshots of the stream, we should observe little change in the rules generated. That is, there should be a

- small number of rules in the old category,
- small number of rules in the new category,
- potentially very large number of rules in the unchanged category, and
- small to medium number of rules in the changed category

of the meta-ruleset base. This feature set can be artificially achieved by setting the mining parameters, such as confidence and support percentage levels, until the above holds. In general, any difference in the above expectations should be immediately investigated. In the course of normal operations, only new and changed rules should require attention.

However, a network stream may not be stable, that is, it may exhibit fluctuations between snapshots. If large numbers of rules disappear or re-appear between snapshots, too much effort would be required to investigate these changes. The solution to this problem would be the finetuning of mining parameters to achieve a stable stream. Unfortunately, this may introduce yet another problem where too many of the more interesting rules may disappear from the snapshot rule sets.

The meta-mining tests performed in the IDDM project were conducted on association rules, with three adjustable thresholds. The first two, confidence and support percentages, relate to the initial mining of snapshot rule sets. The third, deviation percentage, is used to determine whether a rule changes from one rule set to the next. That is, if deviation is observed in either the support or confidence of a rule that is greater than the specified threshold, then the rule is judged to have changed. Otherwise, it is classified into the unchanged meta-rule set base category. The base association rule sets were generated by converting the characteristics of network packets into bit patterns, with each packet potentially setting any of approximately 50 bits.

An early example of meta-ruleset base category rule counts, generated using four sets (night, morning, day and evening) of data extracted from a single day's traffic, with low support, confidence and deviation thresholds is presented in Table 1.

Table 1: Meta-test of four day segments

Test 1	Old	New	Unchanged	Changed
Night & Morning	751	1017	200	98
Morning & Day	333	668	721	261
Day & Evening	1336	613	104	210

This example shows meta-ruleset base category rule counts exhibiting too much fluctuation between rule sets which makes it unusable for meta-mining purposes.

5.3 Non-TCP record test

Test results for the data set with all TCP packets removed yield a more stable picture. The corresponding rule counts using support 0.01, confidence 0.3 and three different deviation values 0.10/0.20/0.30 are shown in Table 2.

Table 2: Meta-test of four non-TCP day segments

Test 2	Old	New	Unchanged	Changed
Night & Morning	0/0/0	0/0/0	83/107/107	24/24/0
Morning & Day	0/0/0	0/0/0	50/50/88	57/57/19
Day & Evening	0/0/0	0/0/0	50/69/107	57/38/0

The results from this test indicate that even with low (support, confidence) percentage pairs it is possible to achieve desirable results if

❑ the deviation is set appropriately, and
❑ the rules are created for a specialised data set.

19

That is, tests including TCP packets may require a much higher granularity than initially used. This means increasing the length of the bit vectors by including more details from the headers for each packet.

5.4 TCP test with 8 watches used

The following tests were performed to determine changes in rulesets over a period of one day, but with higher resolution than in Test 1. Both the support values used to generate the initial itemsets and the confidence values for subsequent rule sets are varied extensively in order to produce results resembling the desired stable looking data stream. This affects the number of total rules produced for each comparison (the higher the support and confidence the fewer rules are produced in each set). Table 3 shows rule counts for a test performed with eight watches. Typically, the final support values used were different not only between rule sets used for different meta-ruleset base generations but also between rule set pairs used to produce a single meta-ruleset base.

In the layout of Table 3, the left-most column includes two 3 hour periods on which meta-mining was performed, with the decimal value displaying the support used to generate the initial itemsets. For example, the expression 0002.01 refers to an association rule generation period from midnight to just before 3am with support 0.01. The right-most pair of numbers indicates the confidence and deviation used for meta-mining. Note that 0608.05 to 0911.10 produced a large number of old rules as the best solution, therefore this period should be investigated further (although some of the old rules seem to have reappeared in the next slot).

Table 3: Meta-test for 8 day segments

Test 3[1]	Old	New	Unchanged	Changed	Confidence-deviation
0002.01 - 0305.01	54	0	526	138	(0.45, 0.25)
0305.01 - 0608.05	46	48	475	143	(0.45, 0.15)
0608.05 - 0911.10	261	83	193	80	(0.45, 0.25)
0911.05 - 1214.05	88	266	658	238	(0.35, 0.20)
1214.20 - 1517.20	60	0	262	31	(0.40, 0.15)
1517.20 - 1820.20	4	7	228	121	(0.40, 0.20)
1820.10 - 2123.10	69	47	551	143	(0.40, 0.30)

5.5 Meta-mining observations

Although the above tables already serve as indication, they and further tests conducted confirmed a large divergence in rule numbers for

❑ low support values, and
❑ low confidence values as expected.

---

[1] Notice that the invariant rules outlined in the beginning of this section do not hold for the meta-ruleset base category totals. This is because different mining parameters were used to generate the meta-rulesets and thus the initial number of rules generated from a given dataset could vary for each test.

There also seems to be a varying 'cut-off' value in deviation that controls the number of changed rules. That is, increasing deviation by a small step of 0.01 from a particular value may cause changed rules to disappear from the meta-ruleset base, even though a higher 0.1 increase may not have had much effect earlier for the same two input rule sets.

It is probably not a good idea comparing rule sets produced with different support and confidence values from the meta-mining point of view as it dramatically affects the number of old and new rules. This follows directly from a characteristic of association mining, namely the lower the confidence and/or support, usually the greater the number of rules in the rule set.

An algorithm to automatically find the best parameters for two given itemsets has been implemented, operating on the observation that *deviation* controls the number of changed rules versus unchanged rules, and *confidence* and *support* control the number of old and new rules in the meta-ruleset base construction. Although confidence and support affect the number of changed and unchanged rules via affecting the total number of rules generated from itemsets, these two parameters are used to control only the old and new rules. The order of preference to achieve optimal threshold values hence is to

❑ increase deviation to control changed and unchanged rule count to fall below pre-defined percentages of the total rule count,

❑ increase confidence by a small amount up to a pre-specified maximum to control new and old rule counts to fall below pre-defined percentages of the total rule count, and

❑ if the above fails, increase support by a small amount up to a given maximum to control new and old rule counts to fall below pre-defined percentages of the total rule count.

It is not guaranteed that this process terminates with a result that is within the desired guidelines. It is also not guaranteed that a solution found will be the optimal solution, and not a local minimum. However, exhaustive search is not used in the program to avoid extensive running times as local minimums should already provide a good indication of the differences between rule sets.

The above program, although initially designed for testing purposes, could also be used for

❑ measuring the stability of a data stream by outputting support, confidence and deviation triplets satisfying our pre-defined perception of a stable data stream for each snapshot, and

❑ as an alert mechanism to be triggered when the above triplets exhibit a large deviation from usual values.

5.6 Stability test on 8 watches a day

The aforementioned parameter-finding program has been used to attain optimal support, confidence and deviation triplets when comparing watches intra- and inter-days using filtered net data obtained over the course of several days. As some periods during each day may be relatively inactive, rather than sticking to strict

calendar day limits, each data set to be partitioned may have been collected over several days. This ensures that enough data is present in a given watch. This also means, that the day-by-day comparison is rather a short period-by-short period comparison which we believe still retains the original intention of comparing data by time-of-day.

A number of observations have been made on the streams collected this way, some of which have been discussed earlier. A default support, confidence, deviation triplet of (0.01, 0.15, 0.1) has been used by parameter-finding algorithm with 0.05 increments for each element:

1. For watches within (our definition of) a 'day', the triplets generally exhibited large variations, indicating a shift in the use of services/traffic flow as the day progresses. This is consistent with our results presented in Table 1 and to some degree, Table 3.
2. When comparing the same watches on separate days, the results are mixed. In some cases, very little difference is observed, in others, large change between rule sets is obvious. In many occasions this also depends on the time of day, that is, day watches may fluctuate more than evening ones.
3. Out of curiosity, tests were also performed for different watches on separate days, for example comparing the period at midnight on one day to a period during the day on another. The expectation was to find large differences in the rules, which was largely supported. However, in some cases, virtually no difference was observed.

The anomalies observed against expectations in points 1 and 2 above have been found to attribute to stream content. Some of the data sets were found to be largely homogenous (eg. containing mainly FTP packets), which resulted in the unusual results indicating no rule changes within a day or across separate watches on different days. When the data contained diverse information - in this case, UDP, ICMP and TCP data (mainly ports 21 FTP, 53 DOMAIN and 119 NNTP) -, the rule sets were changing as expected during the day (eg. FTP did not usually occur outside working hours). Rules in the same timeslots across days also exhibited at least a modest level of similarity. Typically, the initial triplet of (0.01, 0.15, 0.1) changed to values around (0.06 – 0.16, 0.25 – 0.40, 0.10 – 0.45) when satisfying our stability requirement. Higher support and confidence values result in a significant loss of rules in both sets and indicate that they largely differ.

5.7 Stability test on services

Tests were also performed for TCP data associated with a particular service, in this case FTP. The results were encouraging, with very little difference observed both across days and within days. There were, however, some exceptions, which could be explained by the lack of data volume in certain time periods.

5.8 Concerns and general comments

Despite promising initial results with meta-mining, some problems need to be addressed before consideration can be given to the further development of this particular part of the project:

❑ Dependencies within the attributes being mined. Some packet attributes are clearly dependent on others (for example, TCP attributes cannot be set if the IP packet does not carry a TCP packet as payload).

❑ Difficulty in tailoring attributes for mining special data, for example services. Currently, the bits representing attributes and their manipulation are hard-coded and the program needs to be re-compiled for every service. There is also some difficulty in achieving an optimal set of attributes for mining. It would be useful to present attributes as a plug-in module to the main mining algorithms.

❑ Data diversity/volume. During testing, there were sometimes very small sets of data available. Rather than splitting into watches by time, watches by volume could be used. This requires the modification of the data transformation program.

❑ Other than associations, characterisation may be used to describe the general contents of a data stream. A simple algorithm (attribute-oriented induction) that uses domain knowledge is available for this purpose and could be tested to produce a more compact and possibly better description of a net data slice, as illustrated in Section 2.5.

❑ Classification remains the best candidate to capture 'unusual' behaviour in the data, eg. existing attacks. However, changes in classification models over time are probably not as interesting as changes in net data characteristics for finding unknown attacks.

6. Conclusions

Whilst there have been successful applications of data mining techniques in the field of intrusion detection, it is clear that there are further possibilities for the use of the technology. This paper explored some of these additional opportunities, especially focusing on recognising trends in the inherently temporal network data flow. We have presented an overview of techniques that are good candidates for this purpose, and described an architecture facilitating trend analysis requirements.

Initial experiments using components of the architecture indicate that real-time operation is difficult to achieve. Results are produced only after sufficient amount of data is collected and analysed, which can be more appropriately described as near real-time operation. This time lag increases the likelihood of new types of short duration attacks being missed while in progress if they terminate before the next rule update is performed. The chance of this occurring may be reduced by performing updates more frequently on smaller input data sets. However, this may adversely affect rule set stability, by increasing the number of alarms generated due to changes in rule sets over time.

References

[1] Stolfo, S., Prodromidis, A., Tselepsis, S., Lee, W., Fan, W. & Chan, P. (1997) "JAM: Java Agents for Meta-Learning over Distributed Databases", *Proceedings of the Third International Conference Knowledge Discovery and Data Mining*.

[2] Lee, W., Stolfo, S. & Mok, K. (1999) "Mining in a Data-flow Environment: Experience in Network Intrusion Detection", *Proceedings of the Fifth ACM SIGKDD*

International Conference on Knowledge Discovery and Data Mining (KDD 99), San Diego, USA.

[3] Intrusion Detection System homepage URI: http://www.cs.columbia.edu/ids/.

[4] Helmer, G. G., Wong, J. S. K., Honavar, V. & Miller, L. (1999) "Intelligent Agents for Intrusion Detection", *Proceedings of the IEEE Information Technology Conference*, Syracuse, USA.

[5] *Crashcart Documentation*, DSTO, Australia.

[6] Anderson, M., Engelhardt, D., Fiddyment, C., Marriott, D., North, C. & Rajaratnam, D. (2000) "Shapes Vector: An Overview", *DSTO General Document DSTO-GD-0205*, DSTO, Australia.

[7] Schapire, R. E. (1999) "A Brief Introduction to Boosting", *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, Vol 2:1401-1406.

[8] Hofmeyr, S. A., Forrest, S. & Somayaji, A. (1998) "Intrusion Detection using Sequences of System Calls", *Journal of Computer Security*, 6:151-180.

[9] Agrawal, R., Imielinski, T. & Swami, A. (1993) "Mining Association Rules between Sets of Items in Large Databases", *Proceedings of the 1993 International Conference on Management of Data (SIGMOD 93)*, Washington, USA.

[10] Abraham, T. & Roddick, J. F. (1999) "Incremental Meta-Mining from Large Temporal Data Sets", *Advances in Database Technologies, Proceedings of the First International Workshop on Data Warehousing and Data Mining, DWDM'98*, Lecture Notes in Computer Science, Berlin. Springer-Verlag. 1552:41-54.

[11] Han, J., Cai, Y. & Cercone, N. (1992) "Knowledge Discovery in Databases: An Attribute-Oriented Approach", *Proceedings of the 18th International Conference on Very Large Data Bases*, Vancouver, Canada.

[12] Han, J. and Fu, Y. (1994) "Dynamic Generation and Refinement of Concept Hierarchies for Knowledge Discovery in Databases", *Proceedings of AAAI'94 Workshop on Knowledge Discovery in Databases (KDD'94)*, Seattle, USA, pp. 157-168.

[13] Portnoy, L. (2000) "Intrusion Detection with Unlabeled Data using Clustering", *Undergraduate Thesis Paper*, Department of Computer Science, Columbia University, New York, USA.

[14] *KDD Cup 1999 Data Set*, used for The Third International Knowledge Discovery and Data Mining Tools Competition, one of several possible homepage URIs: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

Appendix A: IDDM Library Description

This Appendix lists the algorithms that were created as part of a data mining library to be used for the Intrusion Detection with Data Mining project.

A.1. Library Structure

The data mining library currently consists of at least two separate components:

- Support routines
- Data mining routines
    - Association and meta-mining routines
    - Input/output operations for mining results
    - Future mining techniques

A.2. Support Library

The support library contains generic functions that are used by the data mining algorithms and are not directly available as part of the standard C libraries. They include:

- Bit-wise operators to set/clear/negate a particular bit in memory.
- Print routines to output a portion of memory to stderr as a bitvector/bitmatrix.
- An operator to copy a bitvector into the appropriate row of a bitmatrix.

A.3. Data Mining Library

The data mining library consists of algorithms that perform rule extraction as well as the necessary input/output operations for reading and converting data and presenting results. Figure 5 illustrates the set of routines and files involved in extracting association and meta-association rules for the IDDM project.

Association mining is performed in two steps, itemset creation and rule generation. Separate routines exist for each of these two steps which allows execution to stop midway. The major advantage of this implementation is that by supplying different rule generation algorithms several different output formats may be generated from the itemsets created in the first step. The actual itemset generator is implemented as a binary association miner that does most of its work in memory and requires only a single pass over the data. Therefore it is well suited to extract rules from a data-flow environment where actual data may not be saved to file but transformed directly into input for the mining algorithm.
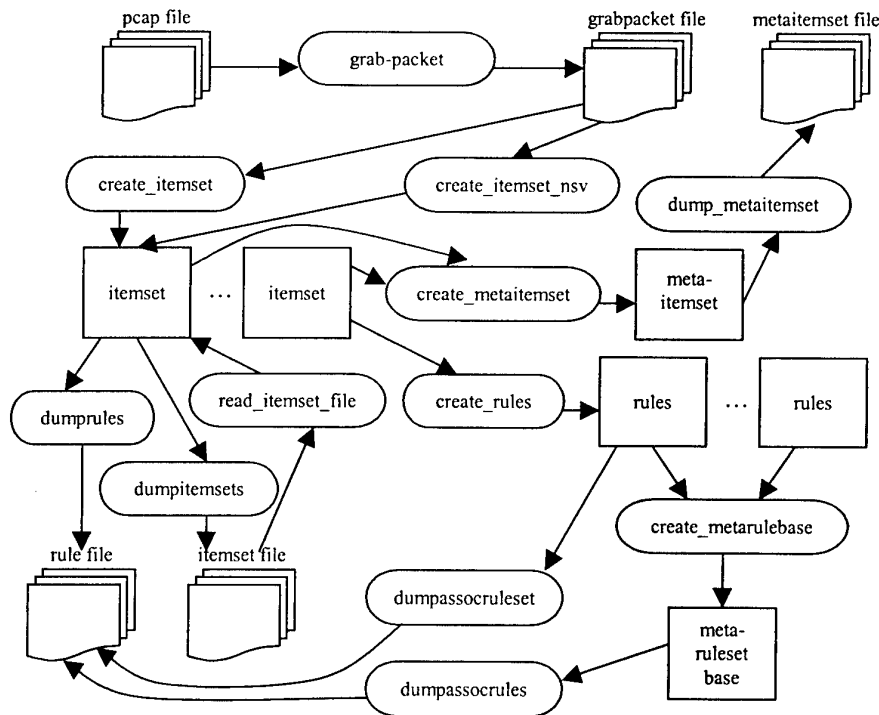
Figure 5: Association mining of net data

A.4. Itemset generation

The itemset generation algorithm runs on a bitmatrix where one dimension is the number of bits representing data elements (attributes) and the other the number of records. The length of the first dimension is rounded up to nearest integer divisible by eight for memory management purposes. This means that some bits at the end of each record may not be used. The bitmatrix used as input is generated from raw network or connection data by separate conversion algorithms.

The output from the algorithm is an itemset chain where each element references a group of items (represented by bits) that occur together with sufficient support. This support value, expressed as a percentage of all records, is supplied as a parameter to the algorithm and is adjustable by the user. An invariant of the itemset chain representation, which is exploited in the implementation of the algorithm, is that smaller sized itemsets are found earlier in the itemset chain, and for each k-length itemset, the indexes of the bits making up the itemset are ordered by magnitude both within a single itemset and also within all k-length itemsets. This form of pre-sorting greatly reduces execution time. The construction of itemsets also relies heavily on the observation that if an itemset is large, that is, has enough support, then all of the smaller itemsets that can be constructed from it must also be large. This explains the following logic employed by the algorithm:

Step 1: Loop through the data and find all 1-large itemsets
Step 2: For every pair of existing k-large itemsets
        Check if the first k-1 elements are the same

If so,
    Construct a $k+1$ itemset comprising of the common $k-1$ elements and the k-th element from each of the pair
    Check if the itemset has enough support
    If so,
        add to the end of the itemset chain[2]
Step 3: See if any itemsets have been created in Step 2.
    If so,
        increase k
        go to Step 2

Because of processing taking place in memory, the algorithm may have high memory requirements. This is contrasted against the fact that only single access to each data record is required, when they are transformed into a binary representation. The binary bitmatrix is then scanned once to produce 1-large itemsets as described in Step 1. After this step, this potentially considerably sized bitmatrix could be discarded as the support of potential larger itemsets can be established from the conjunction of support vectors stored for each 1-itemset. This approach ensures that subsequent itemsets are constructed faster, as only a single AND operation is needed as opposed to k for a k-large itemset. It has, however, the drawback of potentially using even more memory. In the worst case scenario, when every element is a large itemset, for example, when every bit of the bitmatrix is set to 1, the memory requirement increases exponentially. This can amount to a huge additional memory requirement and is probably not a plausible solution in the above case. A better approach would be to not store the support vector for each itemset, but compute it again when necessary. This operation would be far more costly than a simple bitwise AND of two existing vectors, as for a potential k-itemset this means k AND operations. In addition, these AND operations would be performed for each bit, as support vectors are vertical (columns of) in the bitmatrix, while support vectors stored in itemsets are horizontal and thus the AND operation is available for bytes rather than bits. The extra cost of additional AND operations can be relieved somewhat with the use of intermediate support vectors in Step 2, though.

If, however, the number of 1-itemsets is only a small percentage of the number of elements (which heuristically is often the case), the total memory needed may not be much larger, or even smaller than the original size of the bitmatrix. In this case, discarding the original bitmatrix after Step 1 and using support vectors instead may be justifiable.

---

[2] Note that this logic ensures the appropriate ordering of itemsets/elements within itemsets mentioned earlier.

DISTRIBUTION LIST

IDDM: Intrusion Detection using Data Mining Techniques

T. Abraham

AUSTRALIA

DEFENCE ORGANISATION

Task Sponsor
    DISG – Desk Officer MAJ Brenda Sharp   1

S&T Program
    Chief Defence Scientist
    FAS Science Policy
    AS Science Corporate Management   Shared Copy
    Director General Science Policy Development
    Counsellor Defence Science, London   (Doc Data Sheet)
    Counsellor Defence Science, Washington   (Doc Data Sheet)
    Scientific Adviser to MRDC Thailand   (Doc Data Sheet)
    Scientific Adviser Policy and Command   1
    Navy Scientific Adviser   (Doc Data Sheet and distribution list)
    Scientific Adviser - Army   (Doc Data Sheet and distribution list)
    Air Force Scientific Adviser   1
    Director Trials   1

Electronics and Surveillance Research Laboratory
    Director   (Doc Data Sheet and distribution list)
    Chief of Information Technology Division   1
    Research Leader Advanced Computer Capabilities   1
    Michael Pope   1
    Tamas Abraham   1

DSTO Library and Archives
    Library Fishermans Bend   (Doc Data Sheet)
    Library Maribyrnong   (Doc Data Sheet)
    Library Salisbury   1
    Australian Archives   1
    Library, MOD, Pyrmont   (Doc Data Sheet)
    US Defense Technical Information Center   2
    UK Defence Research Information Centre   2
    Canada Defence Scientific Information Service   1
    NZ Defence Information Centre   1
    National Library of Australia   1

Capability Systems Staff
    Director General Maritime Development   (Doc Data Sheet)
    Director General Aerospace Development   (Doc Data Sheet)

Knowledge Staff
    DGC4   (Doc Data Sheet)
    DGISREW R1-3-A142 CDRE Russ Crane, Canberra ACT 2600   1
    DGDKNIT R1-5-A165, Canberra ACT 2600   (Doc Data Sheet)

Army
    ASNSO ABCA, Puckapunyal   4
    SO (Science), DJFHQ(L), MILPO Enoggera, Queensland 4051 (Doc Data Sheet)

Air Force
    IO SQN WGCDR Alex Gibbs   1

Intelligence Program
    DGSTA Defence Intelligence Organisation   1
    Manager, Information Centre, Defence Intelligence Organisation   1

Other Defence Departments
    DSD   1

Corporate Support Program
    Library Manager, DLS-Canberra   1

UNIVERSITIES AND COLLEGES
    Australian Defence Force Academy
        Library   1
        Head of Aerospace and Mechanical Engineering   1
    Hargrave Library, Monash University   (Doc Data Sheet)
    Librarian, Flinders University   1

OUTSIDE AUSTRALIA

ABSTRACTING AND INFORMATION ORGANISATIONS
    Library, Chemical Abstracts Reference Service   1
    Engineering Societies Library, US   1
    Materials Information, Cambridge Scientific Abstracts, US   1
    Documents Librarian, The Center for Research Libraries, US   1

INFORMATION EXCHANGE AGREEMENT PARTNERS
    Acquisitions Unit, Science Reference and Information Service, UK   1
    Library - Exchange Desk, NIST, US   1

SPARES   5

Total number of copies: **42**

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA		1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)

2. TITLE IDDM: Intrusion Detection using Data Mining Techniques	3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)
	Document (U) Title (U) Abstract (U)

4. AUTHOR(S) Tamas Abraham	5. CORPORATE AUTHOR Electronics and Surveillance Research Laboratory PO Box 1500 Salisbury SA 5108 Australia

6a. DSTO NUMBER DSTO-GD-0286	6b. AR NUMBER AR-011-868	6c. TYPE OF REPORT General Document	7. DOCUMENT DATE May 2001

8. FILE NUMBER N9505-21-38	9. TASK NUMBER JNT 98/152	10. TASK SPONSOR DISG	11. NO. OF PAGES 34	12. NO. OF REFERENCES 14

13. URL on the World Wide Web http://www.dsto.defence.gov.au/corporate/reports/DSTO-GD-0286.pdf	14. RELEASE AUTHORITY Chief, Information Technology Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT
Approved for public release
OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, SALISBURY, SA 5108

16. DELIBERATE ANNOUNCEMENT
No Limitations

17. CASUAL ANNOUNCEMENT	Yes

18. DEFTEST DESCRIPTORS
Intrusion Detection Computer Security Intelligent Agents Knowledge Discovery

19. ABSTRACT

The IDDM project aims to determine the feasibility and effectiveness of data mining techniques in real-time intrusion detection and produce solutions for this purpose.

Traditionally, data mining is designed to operate on large off-line data sets. Previous attempts to apply the discipline in real-time environments met with varying success. In this paper, we overview earlier attempts to employ data mining principles in intrusion detection and present a possible system architecture for this purpose. As a consequence, we show that by combining data mining algorithms with agent technologies, near real-time operation may be attained.