Creation and Deployment of Data Mining-Based Intrusion Detection Systems in Oracle Database 10g

Marcos M. Campos
Oracle Data Mining Technologies
marcos.m.campos@oracle.com

Boriana L. Milenova
Oracle Data Mining Technologies
boriana.milenova@oracle.com

Abstract

Network security technology has become crucial in protecting government and industry computing infrastructure. Modern intrusion detection applications face complex requirements – they need to be reliable, extensible, easy to manage, and have low maintenance cost. In recent years, data mining-based intrusion detection systems (IDSs) have demonstrated high accuracy, good generalization to novel types of intrusion, and robust behavior in a changing environment. Still, significant challenges exist in the design and implementation of production quality IDSs. Instrumenting components such as data transformations, model deployment, and cooperative distributed detection remain a labor intensive and complex engineering endeavor. This paper describes DAID, a database-centric architecture that leverages data mining within the Oracle RDBMS to address these challenges. DAID also offers numerous advantages in terms of scheduling capabilities, alert infrastructure, data analysis tools, security, scalability, and reliability. DAID is illustrated with an Intrusion Detection Center application prototype that leverages existing functionality in Oracle Database 10g.

1. Introduction

Intrusion detection is an area growing in relevance as more and more sensitive data are stored and processed in networked systems. An intrusion detection system (IDS) monitors networked devices and looks for anomalous or malicious behavior in the patterns of activity in the audit stream. A comprehensive IDS requires a significant amount of human expertise and time for development. Data mining-based IDSs require less expert knowledge yet provide good performance [19, 1, 15, 6]. These systems are also capable of generalizing to new and unknown attacks. Data mining-based intrusion detection systems can be classified according to their detection strategy. There are two main strategies [19]: misuse detection and anomaly detection. Misuse detection attempts to match observed activity to known intrusion patterns. This is typically a classification problem. Anomaly detection attempts to identify behavior that does not conform to normal behavior. This approach has a better chance of detecting novel attacks. IDSs can also be distinguished on the basis of the audit data source (e.g., network-based, host-based). Successful detection of different types of attacks typically requires a variety of audit data sources [13].

Building an IDS is a complex task of knowledge engineering that requires an elaborate infrastructure. An effective contemporary production-quality IDS needs an array of diverse components and features, including:

- Centralized view of the data
- Data transformation capabilities
- Analytic and data mining methods
- Flexible detector deployment, including scheduling that enables periodic model creation and distribution
- Real-time detection and alert infrastructure
- Reporting capabilities
- Distributed processing
- High system availability
- Scalability with system load

Recent proposals [8, 7] have highlighted the need for an architecture and framework specification for IDSs. While these proposals provide a good foundation, they are somewhat general in nature and focus either on a methodology specification that alleviates the knowledge engineering effort or on component interaction specification (e.g., XML metadata). The details of the infrastructure required to support these feature-rich complex frameworks are not provided, instead such details are proposed to be handled by the system engineers responsible for the implementation.

This paper demonstrates that the Oracle Database, with its capabilities for supporting mission critical applications, distributed processing, and integration of analytics, can be an appropriate platform for an IDS implementation. Given the data-centric nature of the intrusion detection process [13], leveraging existing RDBMS infrastructure can be both efficient and effective.

The current paper presents DAID (Database-centric Architecture for Intrusion Detection) for the Oracle Database. This RDBMS-centric framework can be used to build, manage, deploy, score, and analyze data mining-based intrusion detection models. The described approach is adopted in the Intrusion Detection Center (IDC) prototype – an application implemented using the capabilities of Oracle Database 10g Release 2.

The paper is organized as follows. Section 2 outlines the proposed architecture. The individual components are described and illustrated with references to the IDC prototype and functionality available in the Oracle Database. Section 3 presents the conclusions and directions for future work.

2. A database-centric architecture

DAID (Figure 1) shares many aspects of the AMG (Adaptive Model Generation) architecture [8]. As in AMG, a database component plays a key role in the architecture. Unlike AMG, where the database is only a centralized data repository, in DAID, all major operations take place in the database itself. DAID also explicitly addresses data transformations, an essential component in analytics. DAID has the following major components:

- Sensors
- Extraction, transformation and load (ETL)
- Centralized data warehousing
- Automated model generation
- Automated model distribution
- Real-time and offline detection
- Report and analysis
- Automated alerts

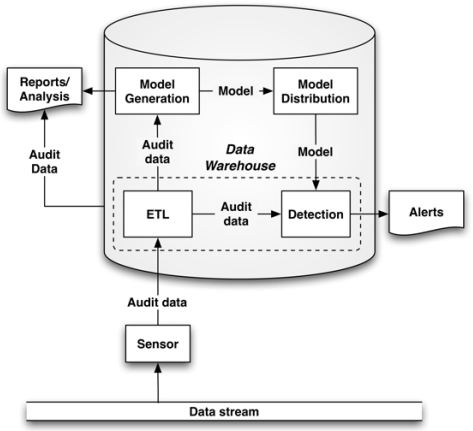

Figure 1. Database-centric architecture for intrusion detection

The activity in a computer network is monitored by an array of sensors producing a stream of audit data. The audit data are processed and loaded in a centralized data repository (ETL). The stored data are used for model generation. The model generation data mining methods are integrated in the database infrastructure – no data movement is required. The generated intrusion detection models can undergo scheduled distribution and deployment across different database instances. These models monitor the incoming audit data. The database issues alerts when suspicious activity is detected. The models and the stored audit data can be also further investigated using database reporting and analysis tools.

The key aspect to the described data flow is that processing is entirely contained within the database. With the exception of the sensor array, all other components can be found in modern RDBMSs. Among the major benefits of using such an integrated approach are improved security, speed, data management and access, and ease of implementation. The following sections discuss and exemplify the functionality and usage of the individual components.

2.1. Sensors

A sensor is a system that collects audit information. Many types of audit streams can be used for detecting intrusions – examples include network traffic data, system logs on individual hosts, and system calls made by processes. Network sensors typically filter and reassemble TCP/IP packets in order to extract high-level connection features (e.g., duration of connection, service, number of bytes transferred). A number of utilities exist to assist the user in this data extraction process [20, 18]. Host sensors monitor system logs, CPU and memory usage on a machine. In a distributed architecture, an emphasis is placed on creating lightweight sensors since they are the only components that must run on the system that is being protected. DAID also favors a lightweight sensor approach since all computation intensive tasks (e.g., feature extraction, model generation, detection) take place in the Oracle RDBMS. In the IDC prototype, we simulate a network environment by streaming previously collected network activity data. This dataset was originally created by DARPA and later used in the KDD'99 Cup

[12]. The same dataset has already been successfully used for demonstrating the capabilities of another intrusion detection framework [2].

2.2. ETL

Typically, sensor audit streams require further pre-processing and feature extraction before the data can be successfully used for data mining model generation. For example, temporal statistical features of connections and sessions have been found informative [13]. Other more elaborate approaches (e.g., conceptual clustering) have also shown promise [11].

In the RDBMS context, SQL and user-defined functions offer a high degree of flexibility and efficiency when extracting key pieces of information from the audit stream. Useful SQL capabilities include math, aggregate, and analytic functions. For example, windowing functions can be used to compute aggregates over time intervals or number of rows. The following query shows a windowing analytic function that computes the number of http connections to a given host during the last 5 seconds:

```
SELECT count(*) OVER
     (ORDER BY time_stamp
      RANGE INTERVAL '5'
      SECOND PRECEDING) as http_cnt
FROM  connection_data
WHERE dest_host = 'myhost'
AND   service = 'http';
```

The KDD'99 network activity data used in our study had already been suitably pre-processed. Therefore the current version of the IDC prototype does not include feature extraction or other raw data transformations at the processing ETL stage.

2.3. Data warehouse

Using the Oracle database as a centralized data repository offers significant flexibility in terms of data manipulation. Inputs from different sources can be combined through joins. Without replicating data, database views or materialized views can capture different slices of the data (e.g., data over a given time interval, data for a specific host). Such views can be used directly for model generation and data analysis. The Oracle database has the additional benefits of data security, high availability and load support, and fast response time [14, 4].

2.4. Model generation

A number of data mining techniques have been found useful in the context of misuse and anomaly detection. Among the most popular techniques are association rules, clustering, support vector machines (SVM), and decision trees [1, 15, 17, 10]. The 10g version of the Oracle database offers robust and effective implementations of data mining techniques that are fully integrated with core database functionality. The incorporation of data mining eliminates the necessity of data export outside the database thus enhancing data security. The model representation is native to the database and no special treatment is required to ensure interoperability.

In order to programmatically operationalize the model generation process, data mining capabilities can be accessed via APIs (e.g., JDM standard Java API [9], PL/SQL data mining API [5]). Specialized GUIs as entry points can be also easily developed, building upon the available API infrastructure (e.g., Oracle Data Miner). Such GUI tools enable ad hoc data exploration and initial model investigation.

The IDC prototype leverages the `dbms_data_mining` PL/SQL package to instrument the model build functionality. We train linear SVM [25] anomaly and misuse detection models. SVM models have been shown to perform very well in intrusion detection tasks [10, 17]. The intrusion detection dataset includes examples of normal behavior and four high-level groups of attacks - probing, denial of service (dos), unauthorized access to local superuser/root (u2r), and unauthorized access from a remote machine (r2l). These four groups summarize 22 subclasses of attacks. The test dataset includes 37 subclasses of attacks under the same four generic categories. We use the test data to simulate the performance of the IDC prototype when operating in detection mode.

For the misuse detection problem, the SVM model was used to classify the network activity as normal or as belonging to one of the four types of attack. The misuse classification results are summarized in Table 1. The overall accuracy of the system was 92.1%. Applying the cost matrix from the KDD'99 competition, the model had a misclassification cost of 0.248. These results are competitive with KDD'99 published results [21]. The poorer performance on the two rare classes (u2r and especially r2l) is a common problem for this dataset and is attributed to differences in data distribution between training and test data.

For the anomaly detection problem, a one-class SVM model [22] was used to identify the network activity as normal or anomalous. Since anomaly detection does not rely on instances of previous attacks, the one-class model was built on the subset of normal cases in the DARPA dataset. On the test dataset, the model had excellent discrimination with an ROC area of 0.989. Sliding the probability decision

Table 1. Confusion Matrix on DARPA Intrusion Detection Dataset (KDD'99 Competition)

actual \pred	normal	probe	dos	u2r	r2l
normal	59332	1048	45	57	111
probe	602	3251	212	62	39
dos	7393	88	222288	75	9
u2r	178	1	8	33	8
r2l	14683	41	7	31	1427

threshold allows trading-off the rate of true positives and false alarms – for example, in this model a true positive rate of 96% corresponded to a false alarm rate of 5%.

Oracle's implementation of SVM is highly scalable [16]. Figure 2 depicts the build scalability of a linear SVM misuse model with increasing number of records. The datasets of smaller size represent random samples of the original intrusion detection data. Tests were run on a machine with the following hardware and software specifications: single 3GHz i86 processor, 2GB RAM memory, and Red Hat enterprise Linux OS 3.0.

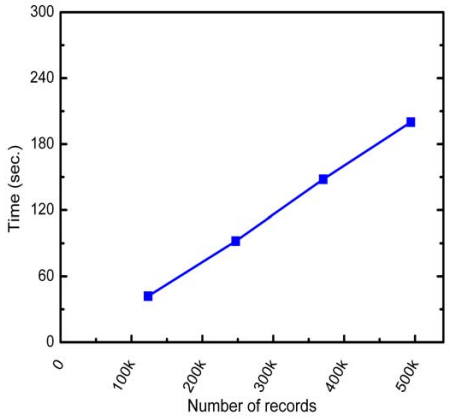

Figure 2. SVM build scalability

The fast training times allow for frequent model rebuilds. In the IDC prototype, we schedule periodic model updates as new data is accumulated. A model rebuild is also triggered when the performance accuracy falls below a predefined level.

2.5. Model distribution

In DAID, model distribution is greatly simplified since models are not only stored in the database but are also executed in it as well. Models are periodically updated by scheduling automatic builds. The newly generated models are then automatically deployed to multiple database instances.

Noel et al. [19] point out a recent trend towards distributed intrusion detection systems where a single IDS monitors a number of network nodes and the monitored information is transferred to a centralized site. Examples of distributed architectures can be found in [26, 24, 23, 3]. IDSs implemented using a framework based on the Oracle RDBMS can transparently leverage Oracle's grid computing infrastructure – Real Application Clusters (RAC). Grids make possible pooling of available servers, storage, and networks into a flexible on-demand computing resource capable of achieving scalability and high availability [14, 4]. RAC allows a single Oracle database to be accessed by concurrent database instances running across a group of independent servers (nodes). The RAC nodes share a single view of the distributed cache memory for the entire database.

An IDS built on top of RAC can successfully leverage server load balancing (distribution of workload across nodes) and client load balancing (distribution of new connections among nodes). Transparent application and connection failover mechanisms are also available, thus ensuring uninterruptible system uptime. Figure 3 illustrates the architecture of an IDS running on an Oracle RAC system.

Audit data are collected from Local Area Networks (LANs). The audit data stream is monitored in one of the available database instances running on the Oracle RAC nodes. In addition to localized monitoring, the shared storage system allows pooling together of information from different sources and enables detection of system-wide attack patterns.

A grid-enabled RDBMS-based IDS needs to be seamlessly integrated with a scheduling infrastructure. Such an infrastructure enables scheduling, management, and monitoring of model build and

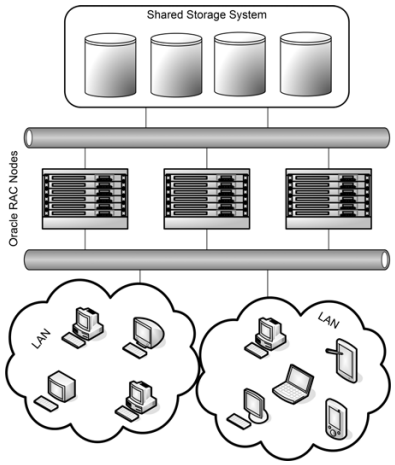

Figure 3. Oracle grid computing

deployment jobs. Although the IDC prototype has not yet been deployed on a RAC system, we use Oracle Scheduler – a scheduling system that meets the above requirements – for management of the model build and deployment jobs.

2.6. Detection

Intrusion detection can be performed either real-time or offline. An IDS typically handles large volumes of streaming audit data. Real-time detection and alarm generation capabilities are critical for the instrumentation of an effective system. ADAM [1] and MAIDS [2] are examples of data mining IDSs addressing issues in real-time detection.

In the context of DAID, an effective real-time detection mechanism can be implemented by leveraging the parallelism and scalability of the Oracle database. This removes the need for a system developer to design and implement such infrastructure. Figure 1 shows detection as a separate entity. However, inside the database, detection can be tightly integrated, through SQL, within the ETL process itself (indicated by the dashed box in Figure 1).

The following example demonstrates how this integration was achieved in the IDC prototype. We make use of Oracle's 10g Release 2 `PREDICTION` SQL operator. The audit data are classified as attack or not by the misuse detection SVM model. The model scoring is part of a database `INSERT` statement:

```
INSERT INTO attack_predictions
      (id, prediction)
VALUES(10001,
            PREDICTION(
              misuse_model USING
             'tcp' AS protocol_type,
             'ftp' AS service,
             ...
             'SF' AS flag,
              27 AS duration));
```

In addition to real-time detection, it is useful to perform offline scoring of stored audit data. This provides an assessment of model performance, characterizes the type and volume of malicious activity, and assists in the discovery of unusual patterns. Having detection cast as an SQL operator allows powerful database features to be leveraged. In our prototype, we create a functional index on the probability of a case being an attack. The following SQL code snippet shows the index creation statement:

```
CREATE INDEX attack_prob_idx
ON audit_data
   (PREDICTION_PROBABILITY(
          anomaly_model,
          0
          USING *));
```

We use the anomaly detection SVM model. The 0 argument indicates that the probability of an anomaly/attack will be returned. Alternatively, a value of 1 would produce the probability of a connection being normal. The `*` symbol maps the `audit_data` table columns to the list of predictors used during the model build. The functional index optimizes query performance on the audit data table when filtering or sorting on anomaly/attack probability is desired. The following query, which returns all cases in `audit_data` with probability greater than 0.5 of being an attack, will have better performance if the `attack_prob_idx` index is used:

```
SELECT *
FROM audit_data
WHERE PREDICTION_PROBABILITY(
            anomaly_model,
            0
            USING *) > 0.5;
```

Processing high volumes of streaming audit data requires a system capable of scoring large datasets in real time. Figure 4 illustrates the `PREDICTION` operator's scalability. The scalability results were generated using a linear SVM misuse model built on 500,000 connection records. The same hardware was used as in the build timing tests.

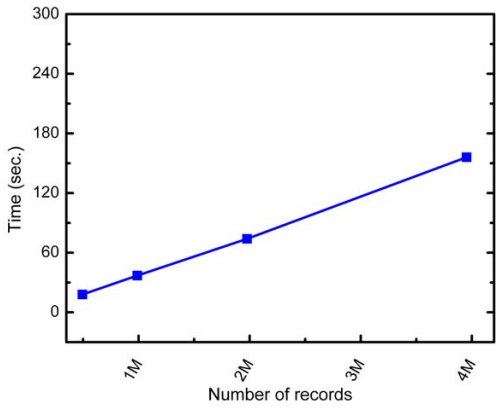

Figure 4. Prediction scalability

The SQL `PREDICTION` operators also allow for the combination of multiple data mining models that are scored either serially or in parallel, thus enabling hierarchical and cooperative detection approaches. Models can be built on different types of audit data or different timeframes, can have different scope (localized vs. global detectors), and can use different data mining algorithms. The combination of multiple models and techniques has been found to be a key requirement for a successful data mining-based IDS [1, 26]. The next example shows a hypothetical use case where two models perform parallel cooperative

detection. The query returns all cases where either `model1` or `model2` indicate an attack with probability higher than 0.4:

```
SELECT *
FROM audit_data
WHERE PREDICTION_PROBABILITY(
            model1,
           'attack'
            USING *) > 0.4
OR PREDICTION_PROBABILITY(
          model2,
         'attack'
          USING *) > 0.4;
```

In the current IDC prototype, the misuse and anomaly detection models are scored independently. We plan on extending the IDC implementation with sequential (or "pipelined") cooperative detection, where the results of one model influence the predictions of another model. In this case, when the `anomaly_model` classifies a case as an attack with probability greater than 0.5, the `misuse_model` will attempt to identify the type of attack:

```
SELECT id, PREDICTION(
                misuse_model USING *)
FROM audit_data
WHERE PREDICTION_PROBABILITY(
            anomaly_model,
            0 USING *) > 0.5;
```

2.7. Alerts

Upon detection of malicious or anomalous activity, an IDS needs to generate an alarm, notify interested parties, and possibly initiate a response. Such a requirement can be easily met by employing existing Oracle database infrastructure. Database triggers are powerful SQL mechanisms that initiate a predefined action when a specific condition is met. The following statement shows the trigger definition used in the IDC prototype where an attack results in posting a message in a queue for handling attack notifications:

```
CREATE TRIGGER alert_trg
   BEFORE INSERT ON
      attack_predictions
   FOR EACH ROW
WHEN (new.prediction <> 'normal')
BEGIN
DBMS_AQ.ENQUEUE(
    'attack_notify_queue',
     ...);
END;
```

The IDC prototype uses Oracle's Publish-Subscribe messaging infrastructure (`DBMS_AQ` PL/SQL package). We chose Publish-Subscribe as the messaging approach since it handles well asynchronous communications in distributed systems that operate in a loosely-coupled and autonomous fashion and which require operational immunity from network failures. The detectors act as publishers who send alerts without explicitly specifying recipients. The subscribers (users/applications) receive only messages that they have registered an interest in. The decoupling of senders and receivers is achieved via a queuing mechanism. Each queue represents a subject or a channel. Active publication of information to end-

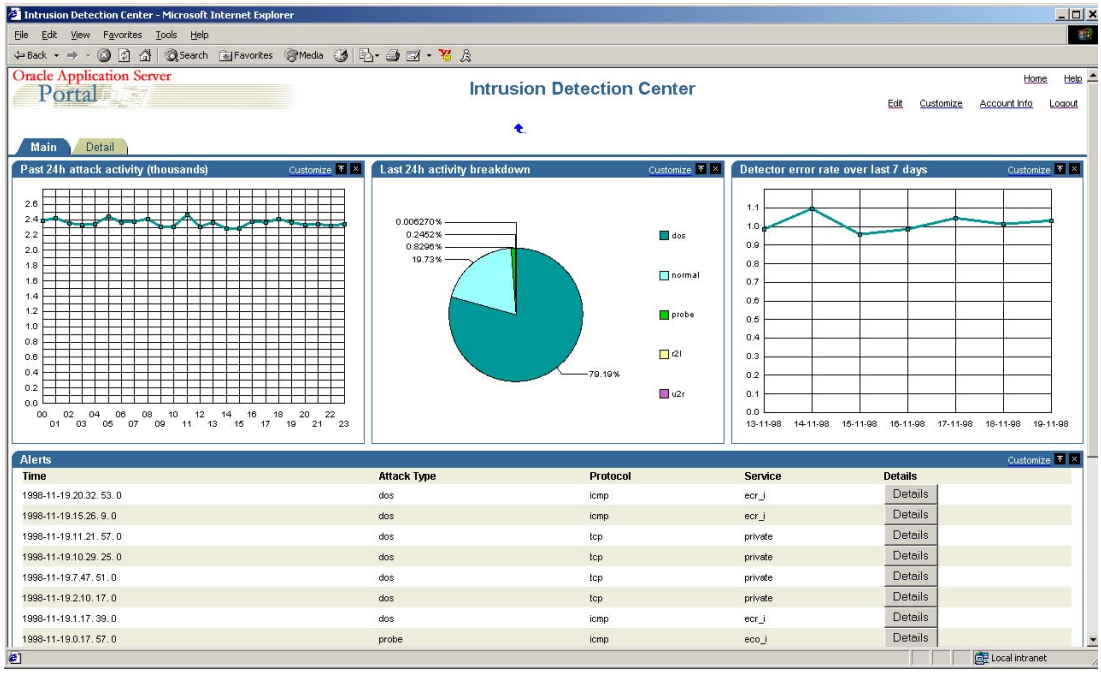

Figure 5. IDC dashboard

users in an event-driven manner complements the more traditional pull-oriented approaches to accessing information that are also available in Oracle. IDS alerts can be also delivered via a diverse range of channels (e.g., e-mails, cell phone messages).

In the IDC prototype, an Oracle Application Server Portal-based application – the Intrusion Detection Center dashboard – monitors the state of the network and displays relevant information. One of the tasks of the IDC dashboard is to monitor the alert queue and display alert notifications. Figure 5 shows a screen shot of the IDC dashboard. The alert notification information includes the type of attack (based on the misused model prediction) and some of the important connection details.

2.8. Reports and analysis

Using the database as the platform for IDS implementation facilitates the generation of data analysis results and reports. Collected audit data, detection predictions, as well as model contents, can be inspected either directly using queries or via higher level reporting and visualization tools (e.g., Discoverer, Oracle Reports). This allows circumvention of a lengthy application development process and provides a standardized and easily customized report generation and delivery mechanism.

The IDC dashboard leverages the tools available in Oracle Portal to instrument a network activity reporting and analysis mechanism. On the main page (Figure 5), users can monitor:

- number of intrusion attempts during the past 24 hours (top left panel)
- breakdown of the network activity into normal and specific types of attack (top center panel)
- detector error rate over the last 7 days (top right panel)
- log of recent alerts (bottom panel)

On the details page (Figure 6), users can review historic data. When a date is selected in the top left panel, the graphs are updated with the corresponding network activity information. The bottom left panel displays the breakdown of network activity into normal and specific types of attack for the selected date. The middle panels show the distribution of connection activity over different types of protocol, service, etc. Clicking on any of the bars produces a breakdown of the network activity (normal and types of attack) for this particular value. For example, in Figure 6 the top right panel shows the breakdown of activity for the tcp protocol for the selected date.

3. Conclusions

Database-centric IDSs offer many advantages over alternative systems. These include tight integration of individual components, security, scalability, and high availability. Current trends in RDBMSs are moving

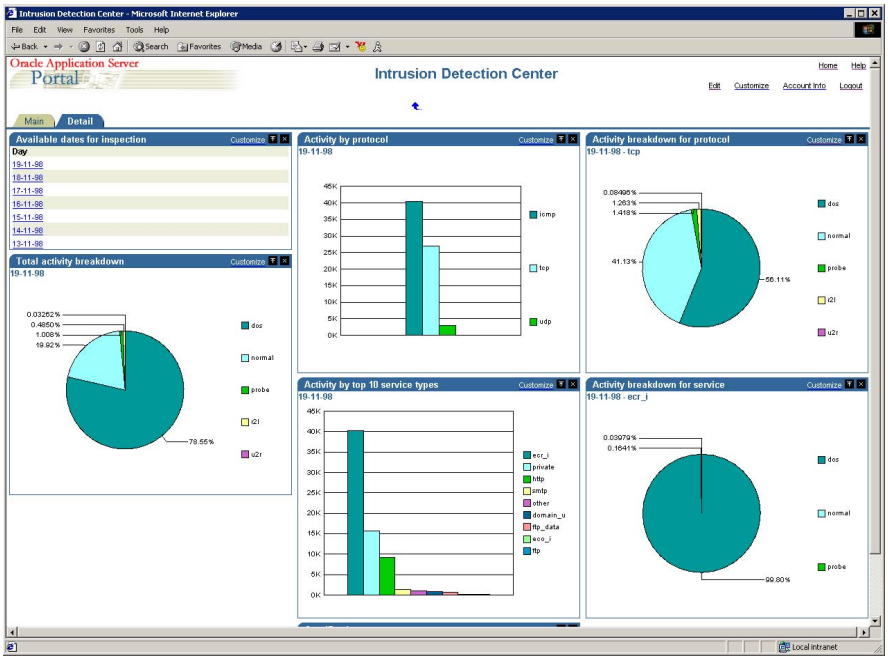

Figure 6. IDC detail statistics

towards providing all key components for delivering comprehensive state-of-the-art IDSs. As illustrated above, Oracle Database 10g Release 2 already incorporates these key functionality elements. By leveraging the existing technology stack, a full-fledged IDS can be developed in a reasonably short time-frame and at low development cost. The similarities between real-time intrusion detection and real-time fraud detection (e.g., credit-card fraud, e-commerce fraud) suggest that DAID could be also applicable to these types of applications. This will be investigated in a future work. We also plan to extend the IDC prototype to take advantage of RAC and sensor data pre-processing at the ETL stage.

4. References

[1] Barbarà, D., Couto, J., Jajodia, S., Popyack, L., and Wu, N., ADAM: A Testbed for Exploring the Use of Data Mining in Intrusion Detection, *ACM SIGMOD Record*, 30(4), 2001, pp. 15-24.

[2] Cai, Y. D., Clutter, D., Pape, G., Han, J., Welge, M., and Auvil, L., MAIDS: Mining Alarming Incidents from Data Streams, In *Proc. 2004 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'04)*, ACM Press, New York, NY, 2004, pp. 919-920.

[3] Chatzigiannakis, V., Androulidakis, G., and Maglaris, B., A Distributed Intrusion Detection Prototype Using Security Agents, *Workshop of the HP OpenView University Association*, 2004.

[4] DataDirect Technologies, Using Oracle Real Application Clusters (RAC), http://www.datadirect.com/techzone/odbc/docs/odbc_oracle_rac.pdf, 2004.

[5] DBMS_DATA_MINING PL/SQL package, Oracle10g PL/SQL packages and types reference, Ch. 23, Oracle Corporation, 2003.

[6] Eskin, E., Arnold, A., Prerau, M., Portnoy, L., and Stolfo, S. J., A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data, In D. Barbarà and S. Jajodia (eds.), *Applications of Data Mining in Computer Security*, Kluwer Academic Publishers, Boston, MA, 2002, pp. 78-99.

[7] Eskin, E., Miller, M., Zhong, Z.-D., Yi, G., Lee, W.-A., and Stolfo, S. J., Adaptive Model Generation for Intrusion Detection, In *Proc. ACMCCS Workshop on Intrusion Detection and Prevention*, 2000.

[8] Honig, A., Howard, A., Eskin, E., and Stolfo, S. J., Adaptive Model Generation, an Architecture for the Deployment of Data Mining-Based Intrusion Detection Systems, In D. Barbarà and S. Jajodia (eds.), *Applications of Data Mining in Computer Security*, Kluwer Academic Publishers, Boston, MA, 2002, pp. 154-191.

[9] Hornick, M. and JSR-73 Expert Group, JavaTM Specification Request 73: JavaTM Data Mining (JDM), http://jcp.org/en/jsr/detail?id=73, 2004.

[10] Hu, W., Liao, Y., and Vemuri, V. R., Robust Support Vector Machines for Anomaly Detection in Computer Security, In *Proc. International Conference on Machine Learning and Applications*, 2003, pp. 168-174.

[11] Julisch, K. and Dacier, M., Mining Intrusion Detection Alarms for Actionable Knowledge, In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM Press, New York, NY, 2002, pp. 366-375.

[12] KDD'99 Competition Dataset, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999.

[13] Lee, W. and Stolfo, S. J., A Framework for Constructing Features and Models for Intrusion Detection Systems, *ACM Transactions on Information and System Security*, 3(4), 2000, pp. 227-261.

[14] Lowery, J. C., Scaling-out with Oracle Grid Computing on Dell Hardware, http://downloadwest.oracle.com/owsf_2003/40379_Lowery.pdf, 2003.

[15] Markou, M. and Singh, S., Novelty Detection: A review, Part 1: Statistical Approaches, *Signal Processing*, 8(12), 2003, pp. 2481-2497.

[16] Milenova, B. L., Yarmus, J., and Campos, M. M., SVM in Oracle Database 10g: Removing the Barriers to Widespread Adoption of Support Vector Machines, *Proc. 31st Int. Conf. Very Large Data Bases*, ACM Press, 2005, pp. 1152-1163.

[17] Mukkamala, S., Janoski, G., and Sung, A., Intrusion Detection Using Neural Networks and Support Vector Machines, In *Proc. of IEEE International Joint Conference on Neural Networks*, IEEE Press, 2002, pp. 1702-1707.

[18] Network Flight Recorder Inc., http://www.nfr.com, 1997.

[19] Noel, S., Wijesekera, D., and Youman, C., Modern Intrusion Detection, Data Mining, and Degrees of Attack Guilt, In D. Barbarà and S. Jajodia (eds.), *Applications of Data Mining in Computer Aecurity*, Kluwer Academic Publishers, Boston, MA, 2002, pp. 2-25.

[20] Paxton, V. BRO: A System for Detecting Network Intruders in Real-Time, *Computer Networks*, 31, 1999, pp. 2435 – 2463.

[21] Results of the KDD'99 Classifier Learning, http://www-cse.ucsd.edu/users/elkan/clresults.html, 1999.

[22] Scholkopf, B., Platt, J. C., Shawe-Taylor, J., and Smola, A. J., Estimating the Support of a High-Dimensional distribution, *Neural Computation*, 13(7), 2001, pp. 1443–1471.

[23] Spafford, E. H. and Zamboni, D., Intrusion Detection Using Autonomous Agents, *Computer Networks*, 34(4), 2000, pp. 547-570.

[24] Staniford-Chen, S., Crawford, R., Dilger, M., Frank, J., Hoagland, J., Levitt, K., and Zerkle, D., GrIDS: A Graph-Based Intrusion Detection System for Large Networks, *ACM Transactions on Information and System Security*, 4(4), 2001, pp. 407-452.

[25] Vapnik, V. N., *The Nature of Statistical Learning Theory*, Springer Verlag, Berlin, 1995.

[26] Zhang, Y., Lee, W., and Huang, Y.-A., Intrusion Detection Techniques for Mobile Wireless Networks, *Wireless Networks*, 9(5), 2003, pp. 545-556.