

Chapter 2

A Framework for Contextual Information Fusion to Detect Cyber-Attacks

Ahmed AlErroud and George Karabatis

Abstract The focus of this research is a novel contextual approach that will be used in detecting zero-day cyber-attacks, generating possible zero-day attack signatures, and automatically measuring their risk on specific software components. In general, zero-day attacks exploit a software vulnerability that has not been discovered, and it is called zero-day vulnerability. This work proposes an approach to identify both zero-day attacks (in real time) and also zero-day vulnerabilities by examining known software vulnerabilities.

The proposed work is an innovative approach, which automatically and efficiently extracts, processes, and takes advantage of contextual information to identify zero-day attacks and vulnerabilities. Contextual information (time, location, etc.) identifies the context that can be used to infer relations between entities, such as cyber-attacks. These relations are called contextual relations. We propose methods to generate zero-day attack signatures using graph-based contextual relations between (1) known attacks and (2) vulnerable software components. These are certainly hard problems to solve, and we doubt that incremental improvements in IDSs will result in a significant solution that drastically improves their effectiveness. Consequently, we propose a substantially different and novel approach: contextual relations, if used intelligently, can reduce the search space in IDSs so that zero-day attacks can be identified in realistic and practical amount of time. There are several reasons that led us to investigate the use of contextual relations to detect zero-day attacks. First, the traditional data mining and pattern recognition techniques lack the desirable effectiveness since they focus on analyzing the data without the use of context. To better identify suspicious activities, direct and indirect contextual paths need to be identified among these activities. These are usually identified manually by domain experts (e.g., identifying relations between cyber-attacks). However, it is quite daunting and challenging to identify all possible

A. AlErroud (✉)

Department of Computer Information Systems, Yarmouk University, Irbid 21163, Jordan
e-mail: Ahmed.aleroud@yu.edu.jo

G. Karabatis

Department of Information Systems, University of Maryland, Baltimore County (UMBC),
1000 Hilltop Circle, Baltimore, MD 21250, USA
e-mail: georgek@umbc.edu

relations via manual investigation. Second, there are several contextual relations that need to be identified among vulnerabilities to predict which ones can lead to zero-day attacks and the software modules they are located, thus, empowering us to generate possible signatures for these attacks.

2.1 Limitations of the Current Research

The current intrusion detection techniques, which utilize some context aspects to detect known and zero-day attacks, have significant limitations. These limitations fall in three major categories as follows:

1. *Lack of semantic relations when context is applied to detect attacks*

The intrusion detection approaches, which have been discussed in Chap. 2, consider some contextual information in their operations. However, such approaches struggle when it comes to apply *semantic relations* with context. Utilizing semantic relations in a specific context is quite significant for predicting cyber-attacks. The attack graph approaches utilized by Noel et al. [1–4] consider the relationships between machines with vulnerabilities, where an attack graph of machines which have vulnerabilities is constructed before the events occur. An alert is raised at runtime if its corresponding events are mapped to adjacent vulnerabilities in the attack graph. While the attack graphs incorporate some contextual information such as the relationships between the vulnerabilities of machines, their major limitations are threefold.

First, an attack graph requires to be traversed at runtime in order to create possible attack scenarios. However, an attack graph contains in general too many paths. Traversing all paths to discover possible attack scenarios makes the detection process not efficient at runtime.

Second, the attack graph is only aware of known vulnerabilities; thus, it cannot be used to detect unknown or zero-day attacks.

Third, updating such graphs, with new knowledge about the domain (e.g., information about machine vulnerabilities), in a real-time manner is not feasible. As new vulnerabilities are added, the paths, in these graphs between different vulnerabilities, need to be updated in order to add new attack scenarios. Due to these limitations, attack graph approaches might result in high false alarm rate, specifically if machines are recently patched against the exploits in attack graph. Attack scenarios need to be precomputed and well defined by incorporating semantic relations between contextually related attack scenarios. We believe that it is more feasible to pre-calculate the contextual semantic relationships between possible attacks by (1) finding the similarity between these attack features, (2) pre-calculating the possible paths that connect these attacks, and (3) automatically augmenting such relations using domain expertise.

Few studies have investigated the issue of incorporating semantic information in the process of detecting cyber-attacks. Mathew et al. [5] address the third limitation of attack graphs. They present a strategy for a real-time situation

awareness of multistage cyber-attacks. They adopt a practical correlation approach that considers high-level multistage attack semantics of heterogeneous sensor events without delving into vulnerability level details. Attack models called guidance templates are used to guide this correlation approach and produce dynamic attack hypotheses that are relevant to an evolving situation. However, there are two limitations in this approach. *First*, it does not maintain any statistical models to discover relationships between attacks, such as the similarity between attack features. While defining guidance template is definitely useful to capture relations between attacks, identifying such relations in a manual manner is not effective to capture all possible relations between attacks. *Second*, this approach completely ignores the information about the target environment, such as information about host vulnerabilities. Thus, this approach would result in a higher false alarm rate, if the hosts were patched against these vulnerabilities, but the IDS was not aware about it.

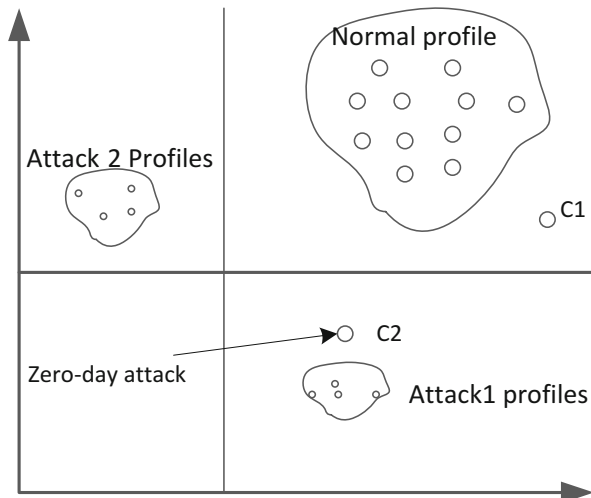
2. *Overlooking the similarity with known attack contextual profiles to discover potential zero-day attacks*

A zero-day attack is an attack, which utilizes the previously unknown vulnerability in a computer application. A zero-day attack occurs on day zero of awareness of the vulnerability that causes it. The developers have no knowledge about zero day to address and patch such vulnerability. Since approaches that utilize vulnerability databases fail to detect zero-day attacks, there have been few studies addressing zero-day attack detection problem using machine learning techniques such as unsupervised anomaly detection techniques [6, 7], support vector machines (SVM) [8], and clustering approaches [7, 9].

The main assumptions in these approaches are twofold: first, to assume any instance (e.g., connection) *that does not conform to a well-defined notion of normal activity as a zero-day attack* and, second, to assume any instance *that does not match the known attack signatures as a normal activity*. However, there are two main shortcomings in these assumptions; for the first assumption, it is not always possible to generate well-defined normal profiles. For the second assumption, it ignores the fact that in general, the zero-day attacks have some degree of similarity with one or more known attack signatures. The previous approaches, however, look at the problem as semi-supervised anomaly detection one, focusing only on discovering zero-day attacks as anomalies that deviate from normal profiles. Since it is not easy to define normal profiles because there are too many of normal patterns, utilizing only semi-supervised anomaly detection techniques to detect zero-day attacks might lead to a high false positive rate. As reported by Leung [6], considering semi-supervised anomaly detection, such as one-class SVM (using the inner product technique), leads to more than 48 % of false positives (normal activities predicted as zero-day attacks).

Figure 2.1 gives an example of the problem that existing approaches have in detecting zero-day attacks. The figure shows several normal profiles as one cluster. These profiles can be created using machine learning techniques, and they describe normal signatures; thus, it can be used to match incoming connections at runtime. The figure also shows another two clusters of known attack

Fig. 2.1 The limitation of detecting zero-day attack using anomaly detection



profiles. For simplicity, we assume that a specific IDS has knowledge about these two attacks only. Each attack cluster consists of several profiles for the same attack.

Let us also assume that there are two network connections C_1 and C_2 that target a system which utilizes a semi-supervised anomaly-based ID technique to detect attacks. While the network connection C_1 is far from both attack 1 and attack 2 profiles, it may be predicted, by the semi-supervised anomaly techniques, as a zero-day attack because it is relatively far from normal profiles. However, since C_1 is not similar to any attack profiles, it is very possible that it is a new pattern of normal activities. Thus, the typical prediction, by the anomaly detection technique, would be most likely a false positive. On the other hand, if C_1 similarity with attack 1 and attack 2 is considered, it may be deemed as a normal activity because it is not similar to any attack profile.

Additionally if the similarity between connection C_2 features and both attack profiles is considered, C_2 may be declared as zero-day attack, since (1) it is far normal profiles, and thus it can be declared as an anomaly, and (2) it is also similar to attack 1 profiles. Thus, for C_2 connection we have two evidences that it could be a zero-day attack. Considering similarity with profiles of known attacks can be significant to avoid a large amount of false positives. Thus, we aim to combine both semi-supervised anomaly detection techniques and similarity with known attack profiles in order to detect zero-day attacks. We want to examine if such a combination would result in improving the zero-day attack detection rate, minimizing the false positive rate, and improving the computational efficiency in the detection process.

3. Existing attack detection techniques are not computationally feasible

The network data, which has to be analyzed by IDS, contains too many numerical and categorical attributes. Thus, the dimensionality of this data is relatively high. When such data is analyzed at system runtime to detect possible intrusions, the computational time of the detection process is definitely high. In anomaly detection approaches, this problem is more noticeable. These techniques need to compare the incoming connections with too many normal profiles in order to discover zero-day intrusions, and thus these techniques are not computationally efficient.

There are several approaches that have been applied to reduce the dimensionality of ID data, such as principal component analysis (PCA) [10–12], singular value decomposition (SVD) [10], local linear embedding (LLE) [13], and linear discriminant analysis (LDA) [14]. However, the existing approaches apply the dimensionality reduction process on the entire distribution of the dataset. Regardless of how effective these approaches are, the dimensionality reduction techniques have not been utilized to discover the local *context* where a particular attack occurs, that is, the dimensionality reduction techniques have not been performed, by considering only the instances of that attack in the ID data. The local context, such as the *activities* which identify the context in which a particular attack happens, is very important to investigate for zero-day attacks. As mentioned earlier, such attacks have a similarity with known attack profiles. Thus, the ability to accurately identify the local contexts of attacks (i.e., their context profiles) has two benefits in the reduced dimensional space. *First*, it minimizes the computational time of the detection process as the incoming connections need to be compared with few attack profiles described in the reduced dimensional space. *Second*, it can be used to detect zero-day attacks that occur in a similar context of particular known attack.

As a conclusion of our discussion, there is a need for an approach that provides the following functionalities to effectively detect known and zero-day attacks:

- To apply semantic relations with context in the process of intrusion detection
- To consider the characteristics of an environment (e.g., the patches on hosts) in detecting known and zero-day attacks
- To consider both anomaly detection and similarity with known attack context profiles in detecting zero-day attacks
- To apply dimensionality reduction and other transformation techniques, within a particular context, not on the entire distribution of data
- To consider other contextual aspects in data such as location and time in detecting known and zero-day attacks

In order to address the limitations discussed above, we propose a framework, which will be utilized to extract, model, and use contextual information in detecting both known and unknown attacks.

To address the first limitation, the proposed framework applies semantic networks of attacks and several types of context profile, which describe specific environment, to detect known attacks.

To address the second limitation, the proposed framework considers semi-supervised anomaly detection and attack profile similarity to detect zero-day attacks.

To address the third limitation, the proposed framework utilizes data transformations through linear discriminant analysis, which can efficiently handle numerical features in a reduced dimensional space.

In this chapter, we describe our proposed contextual framework to detect known and zero-day attacks. The proposed framework will be validated through a series of experiments, conducted on an intrusion detection dataset.

The chapter is organized as follows: In Sect. 2.1, we discuss the limitations of the existing research approaches, which utilize context in their operation and have been utilized to detect known and zero-day attacks. Next, in Sect. 2.2, we describe the components of the proposed contextual infusion framework, which will be used to detect known and zero-day attacks. The proposed framework aims to address the limitations of the existing approaches. The next section describes the components of the proposed framework in details.

2.2 A Framework for Contextual Information Fusion to Detect Known and Zero-Day Attacks

In this section, we present our framework, which consists of several contextual models that utilize different contextual information categories for the detection of known and zero-day attacks. Figure 2.2 shows a high-level architecture of our framework.

The process of creating, using, and evaluating this framework consists of three phases:

- I. *Static phase*: during which ID data are being used to generate contextual models ahead of time that will be used to detect known and zero-day attack at runtime phase. The static phase consists of *data preprocessing*, *contextual information extraction*, and the *contextual information modeling* subphases as outlined below:
 - *Data preprocessing phase*: during this phase, several data preprocessing techniques are applied on ID data. Data preprocessing is needed to facilitate extracting contextual information, which will be used in creating contextual models (e.g., semantic networks) that have the capability to detect attacks.
 - *Contextual information extraction*: during this phase, several categories of contextual information (e.g., the similarity between attacks to identify

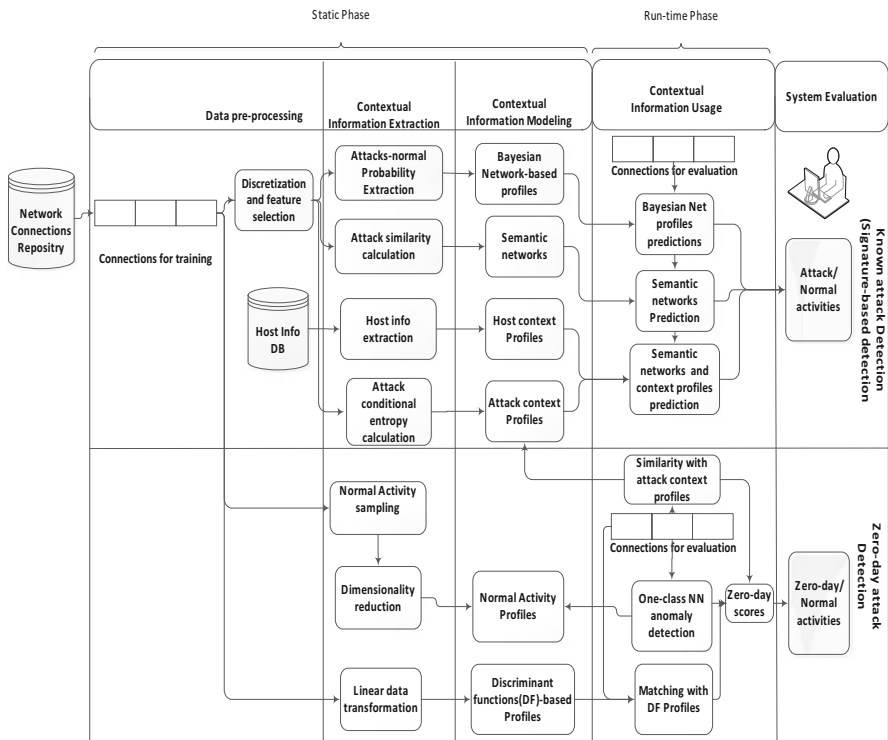


Fig. 2.2 The proposed framework for contextual information infusion to detect known and zero-day attacks

relation context) are extracted and identified. The extracted contextual information is then used to create contextual models as profiles.

- *Contextual information modeling*: during this phase, the contextual information extracted in the previous phase is modeled using several context modeling techniques. The contextual models are utilized at runtime to detect known and zero-day attack.

II. *Detection (runtime) phase*: during this phase, contextual model implementations are utilized to analyze an incoming connection and determine whether it is a known, a zero-day attack, or a normal activity. The runtime phase contains a contextual information usage subphase, which utilizes different types of profiles to discover attacks.

III. *System evaluation phase*: during this phase, a series of experiments have been conducted to validate the effectiveness and efficiency of the contextual models we utilized to detect known and zero-day attack. We use several measures during the evaluation phase, such as precision, recall, F-measure, true positive (TP) and false positive (FP) rates, detection accuracy (AC), and the ROC curves. The evaluation process is discussed in the experiment chapter.

It should be emphasized that the phases above have been applied to create contextual models to detect either known or zero-day attacks. The phases of creating these models are therefore customized based on the detection mode, which can be either a *known attack* (signature-based) detection mode or *zero-day attack* detection mode.

We next provide a high-level description of the ID data used as input, the detection modes, and phases utilized in creating the contextual models for each mode:

- (a) *The network connection repository*: The system takes its input data from this repository, which serves as a database that stores data about connections that target computer networks. This data is collected by network sensors (e.g., IDSs), and it can be represented as features, which describe individual characteristics of the connection, such as connection *protocol*, connection *duration*, *services requested*, packet flow statistics, etc. The data is presented in a high-level format called *connection records*. Each connection consists of 41 features (38 numerical and 3 categorical). The connections are labeled as normal or attacks, where the attacks are categorized in one of the following four categories: user to root (U2R), remote to local (R2L), denial of service (DoS), and probe. The connections are divided into two parts:

1. *The training part*: these connections have been utilized during the static phase to create the contextual models we utilize to detect known and zero-day attacks.
2. *The evaluation part*: these are treated as incoming connections, which are used during runtime phase, to evaluate the rate of attack detection and also to measure how efficient the contextual models are.

- (b) *System detection modes*

The framework is designed to detect attacks in two modes: the *known attack (signature-based) detection mode*, during which the known attacks are identified, and the *zero-day attack detection mode*, during which zero-day or unknown attacks are identified. We now provide a high-level description of both known and zero-day attack detection modes and the phases utilized in creating and using the contextual models for both modes:

1. *The known attack (signature-based) detection mode*: The contextual models in this mode are used to detect known attacks, whose signatures can be detected within a specific context. The models in this mode are created through the following phases:
 - I. *Data preprocessing*: During this phase, discretization is used to convert the numerical features into bins. Feature selection has been also utilized to identify features that best describe attacks and normal contexts hidden in training connections. The outcomes of feature selection and discretization processes are used to extract contextual information.
 - II. *Contextual information extraction*: During this phase, contextual information is identified and extracted after data preprocessing is carried out

on training connections. The extracted information is used later to create contextual models that are eventually utilized during runtime to detect known attacks. During this phase, the following extraction procedures are carried out:

- *Attack and normal probability extraction*: The conditional and joint probabilities of attacks and normal activities are extracted, based on the occurrence of particular features (activities) in the training connections. The probability information extracted during this phase is used to model the *activity context*, which describes the conditions of occurrence of attacks and normal activities as Bayesian network-based context profiles.
- *Attack similarity calculation*: Several similarity coefficients are utilized to calculate the pairwise similarity between different types of attacks extracted from training connections. The similarity between attacks is used to model the *relation context* information as *semantic networks*.
- *Attack conditional entropy calculation*: The training connection records are used to calculate the conditional entropy of attacks given particular features. The features, which result in low conditional entropy values, are used to model the *activity context* as *attack context profiles*. These profiles are used to describe the preconditions of attack occurrence. Such profiles are used at runtime to predict the occurrence of different types of attacks.
- *Host information extraction*: Information about the environment of hosts, such as the recent patches, operating systems, applications, application versions, etc., is extracted from the host info DB. Automatic vulnerability scanners might be aware of collecting information about the computing environment. The information about hosts and their environment is used to model the *individuality context* information about hosts as *host profiles*.

III. *Contextual information modeling*: During this phase, the contextual information extracted during contextual information extraction phase is modeled using the following types of profiles:

- *Bayesian network-based profiles*: The extracted attacks and normal probability information are used to create these types of profiles. These profiles consist of different attack types, normal activities, connection feature values, and the probability of specific attack/normal activity given these feature values.
- *Semantic networks*: The attack similarity values calculated earlier are used to create semantic networks, which are graphs with nodes representing attacks and edges representing relationships between them. Domain knowledge about attacks is also added to these networks to further improve the quality of semantic relations between

attacks. The semantic relationships among attacks are used to expand predictions at runtime.

- *Attack context profiles*: The conditional entropy of attacks given feature values is used to create attack context profiles. These profiles consist of several known attack signatures. Each profile consists of an attack name and the connection features which minimize the uncertainty about the occurrence of that attack (i.e., the features which minimize conditional entropy of that attack). These profiles are used at runtime to detect attacks when the signatures described in their context profiles are triggered.
- *Host context profiles*: The information extracted about hosts is used to create host context profiles. These profiles incorporate contextual information about hosts and their environment and are used to filter out potential attacks that are out of host's current context.

IV. *Contextual information usage*: During this phase, the contextual models created during the previous phase are utilized to analyze runtime connection and examine if they are possible known attacks or normal activities. The types of predictions, which are carried out by contextual models at runtime, are:

- *Bayesian network profile predictions*: This type of prediction is performed as follows: the runtime connection features are analyzed. Then, the probabilities of each attack and the normal activity are calculated. The prediction with the maximum probability is considered the correct prediction. This prediction can also be passed to semantic networks, which will be used to expand the prediction made by Bayesian network profiles.
- *Semantic network predictions*: The initial prediction made by Bayesian network is selected as a starting node to search in semantic network for additional relevant attacks. Based on the strengths of that node's relation with other nodes in the semantic network, some of these nodes are added to the initial prediction. This expansion process helps in predicting these attacks ahead of time, especially if the initial attack is a step in a multistage attack.
- *Semantic network and context profile predictions*: Expanding the initial predictions may be helpful to include some semantically related attacks to the initial one; nevertheless, this expansion would result in adding some attacks which are not relevant to the current context. Thus, context profiles are used to filter out (discard) some semantic network predictions, which are not relevant to the current context. The process of discarding the nonrelevant semantic network predictions is done based on the *activity* context, that is, the flow of activities (feature values) in the corresponding incoming connection, on which the predictions are made. The feature values of these connections might not match the context profiles of attacks predicted

by semantic networks. Therefore, these predictions need to be filtered out using attack context profiles. Host context profiles are used to filter out the predictions, which are not relevant to the current status of host targeted by the connection. Each host is supposed to have a specific context. For instance, the host could be patched against specific vulnerability if such vulnerability is related to a particular attack that has been predicted by a semantic network; this attack needs to be discarded since it is not relevant to that host's *individuality* context.

2. *The zero-day attack detection mode*: In this mode, several contextual models are used to detect zero-day attacks given an incoming connection, which does not conform to a well-defined notion of normal activity or a known attack context profile. In this detection mode of our system, the connections used at runtime are not passed to Bayesian networks or to semantic networks because these models are specific to detect known attacks. The only part we reuse, from known attack detection models to discover zero-day attacks, is attack context profiles. The contextual models in the zero-day attack detection mode are created and used through the following phases:

I. *Contextual information extraction*: During this phase, contextual information about normal and attack connections is identified and used to create specific contextual models that can be used to discover zero-day attacks. The following extraction procedures are carried out during this phase:

- *Normal activity sampling*: This process is carried out to select representative normal training connections, which are used as part of an anomaly-based zero-day attack detection technique. These connections are represented as normal activity profiles, and they are used to examine runtime connection records for possible zero-day attacks.
- *Linear data transformation*: The linear data transformation process utilizes linear discriminant analysis (LDA) technique on the training connections to create several discriminant functions that linearly separate normal activities from attack activities. The extracted linear discriminant functions are used to create several profiles, which are used at runtime, to estimate the probability of zero-day attack given the incoming connection record features as inputs.

II. *Contextual information modeling*: During this phase, the contextual information extracted during contextual information extraction phase is modeled using the following types of profiles:

- *Normal activity profiles*: The representative normal connections are stored as normal activity profiles. These profiles are used at runtime to detect zero-day attacks using a semi-supervised anomaly detection technique. A dimensionality reduction process has been carried out on

these profiles to improve the efficiency of the detection process. These normal profiles are then stored in a reduced dimensional space, by utilizing only the most significant eigenvectors to describe their characteristics.

- *Discriminant function-based profiles*: Each of these profiles consists of discriminant functions, which linearly describe attacks and normal activity patterns. The purpose of generating discriminant function-based profiles is to use them at runtime to estimate the probability of zero-day attack, given an incoming connection that does not match known attack context profiles.

III. *Contextual information usage*: During this phase, the profiles which are created during contextual information modeling phase are utilized at runtime to detect possible zero-day attacks. The suspicious incoming connections, which have no matching known attack context profiles, are processed using one or more of the techniques outlined below:

- *Similarity with known attack context profiles*: During which, the incoming connection features are matched with several known attack context profile features to calculate the similarity between connection record and these known attack profiles. The output of such matching is the maximum similarity value. The higher this value is, the greater the possibilities that the incoming connection record could be a zero-day attack, while we do not claim that all attacks detected using similarity with attack profiles are zero-day attacks as some of them might be known attacks. However, we focus mainly on the success rate of our approach in detecting zero-day attacks which are unknown to our framework components as we show in our experiments.
- *One-class NN anomaly detection using normal activity profiles*: The deviation of incoming connection features from normal activity profiles is calculated. An anomaly score is assigned to each connection, where the high anomaly score indicates that the incoming connection could be a zero-day attack.
- *Estimated probability calculation discriminant function (DF)-based profiles*: The connection records are passed to several linear discriminant functions to estimate the probability of attack labels given the features of connections. Once the estimated probabilities are calculated using the corresponding discriminant functions, the highest estimated probability value is selected and used as a zero-day score. This score is used to declare a possible zero-day attack.

We next describe in detail the process of creating contextual models and use their implementation to detect attacks in known and zero-day attack detection modes.

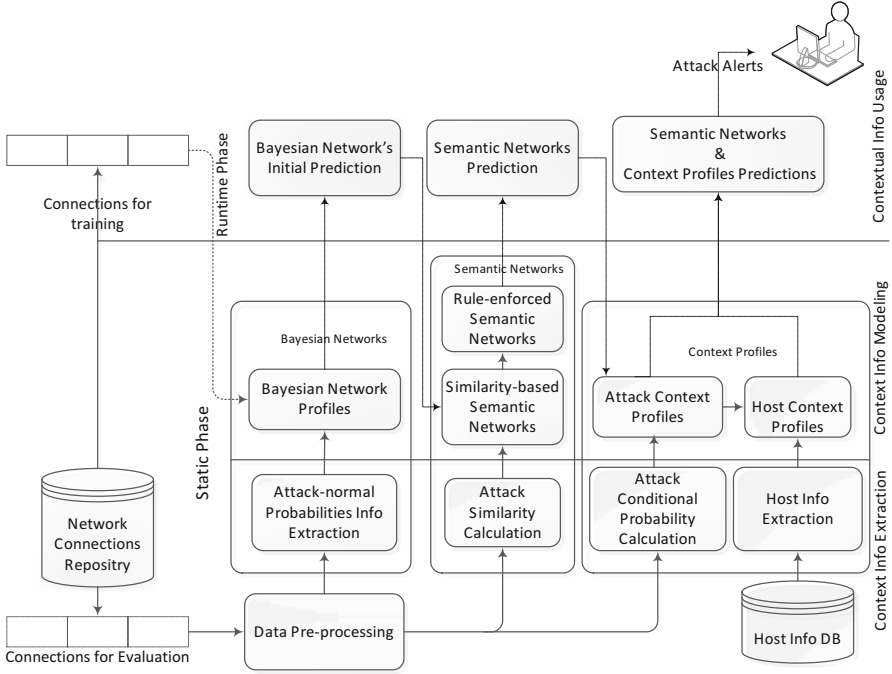


Fig. 2.3 The misuse detection components

2.2.1 The Known Attack (Signature-Based) Detection Mode

In this section, we describe the details of the contextual model we create and use during this mode.

Figure 2.3 illustrates the major components and processes, which are created and used during this mode. The *static phase* consists of several pre-calculated contextual models that are being used at runtime. Let us describe in detail the input data utilized and the contextual model which are created.

2.2.1.1 Network Connection Repository

This is the standard NSL-KDD database [15, 16] which is used to create and evaluate the effectiveness of contextual model implementations in terms of known attack detection rate in this mode. The database consists of connection records each of which represents a time window of two seconds of interaction between specific source and target hosts. Each record in the dataset is labeled with the name of an actual attack or “normal” if there is no attack.

Table 2.1 shows a sample of the connections with some connection features. For instance, the *duration* describes the duration of the connection; *src_bytes* and

Table 2.1 A sample of connection records with selected features

Connection_id	Duration	Protocol	Service	Flag	src_bytes	dst_bytes	Count	Label
82755	0	TCP	IRC	REJ	0	0	477	Satan
82970	0	TCP	bgp	REJ	0	0	324	Satan
101057	0	TCP	Courier	SH	0	0	1	Nmap
220670	0	TCP	Domain	SH	0	0	1	Nmap
9	0	UDP	domain_u	SF	29	0	2	Normal
76	0	UDP	domain_u	SF	44	115	1	Normal
13632	5	ICMP	eco_i	SF	18	0	1	Ipsweep
13638	0	ICMP	eco_i	SF	18	0	1	Ipsweep

dst_bytes are bytes exchanged by source and destination and vice versa. *Flag* describes the status flag of the connection. *Protocol* is the connection protocol (e.g., TCP, UDP, etc.). *Service* is the destination service (e.g., telnet, Ftp, etc.). *Count* describes the number of connections to the same host as the current connection in the past 2 s. The last column, *label*, represents the actual attack, if any, this connection led to. Some connections are normal activities. We use this attribute to evaluate the correctness of attacks predicted by our system. The data in this repository is divided into two parts, the connections, which have been used to create the contextual models to detect known attacks (or the training connections), and the connections which have been used to evaluate these contextual models in detecting these attacks (the evaluation connections). Some preprocessing steps are required to create the contextual models. Next, we describe these data preprocessing steps.

2.2.1.2 Data Preprocessing

The data preprocessing in this mode, as shown in Fig. 2.4, consists of two processes, *discretization* and *feature selection*. Most of the features in the dataset are numerical. Continuity in these feature values makes it more challenging to apply data mining concepts to the dataset. We discretized the continuous features in the training connections. Bins are automatically created based on the label values present in the dataset using a supervised binning approach.

The dataset has 41 columns (including the label). This means that each label is described by 40 features. In order to minimize the computational complexity, we carried out feature selection using the correlation-based feature selection method proposed by Hall et al. [17] on this dataset to yield the highly ranked features shown in Table 2.2. Previous work done on this dataset utilizes most of these features in their experiments. The preprocessed data is used to create several contextual models to detect zero-day attacks. Let us start with one of these models, which are Bayesian network-based profiles.

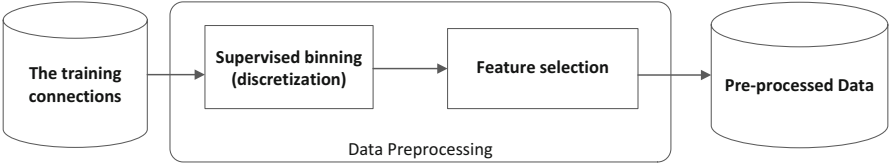


Fig. 2.4 The data preprocessing steps

Table 2.2 Correlation-based feature selection method on NSL-KDD dataset

Feature	Description
Duration	Duration of the connection
Land	1 if connection if from/to the same host/port, 0 otherwise
Protocol	Connection protocol (e.g., TCP, UDP)
same_srv_rate	% of the connections to the same service
Flag	Status flag of the service
dst_bytes	Bytes sent from destination to the source
Count	Number of connections to the same host as the current connection in past two seconds
diff_srv_rate	% of connections to different services
dst_host_diff_srv_rate	% of different services on the current host
Hot	Number of “hot” indicators
dst_host_same_src_port_rate	% of connections to the current host having the same src port
wrong_fragment	Number of wrong fragments
src_bytes	Bytes sent from source to destination
Service	Destination service (e.g., telnet, Ftp)

2.2.1.3 Using Bayesian Networks (BN) for Known Attack Detection

Bayesian networks are directed acyclic graphs that have been used to infer causal relationships based on conditional probability of a node given its parent. We use a simplified version of Bayesian network called simple Bayesian network, which consists of one parent node and many children leaf nodes. The parent node represents the label to be predicted, which can be a specific attack or a normal activity; the child nodes represent some connection features which are the observations that will be used to calculate attack probability. The processes of creating and using Bayesian networks to detect attacks in known attack detection mode consist of the following phases:

Contextual information extraction (attack and normal probabilities info extraction): Information about probabilities of attacks and normal activities, given specific observation in the dataset, are calculated. In BN, each prediction can be an attack or normal activity, and it represents an unobserved parent node (i.e., the label) to be predicted. Initially, the training connections are used to calculate the probability of each feature value given a specific attack or a normal activity. Once the conditional probabilities of features given specific attacks or normal patterns are generated, they can be stored in Bayesian network-based profiles.

Table 2.3 The structure of Bayesian network-based context profiles

Attack name	Feature name	Lower bound	Upper bound	Categorical feature value	Probability
Rootkit	src_bytes	2347.51	5678.765		0.01639
PortswEEP	src_bytes	2347.51	5678.765		0.00149
Neptune	src_bytes	2347.51	5678.765		0.00756
Normal	src_bytes	2347.51	5678.765		0.00500
Smurf	src_bytes	2347.51	5678.765		0.00136
Smurf	Protocol			TCP	0.96825
Loadmodule	Protocol			TCP	0.90476
Perl	Protocol			TCP	0.77777
Guess_Password	Protocol			TCP	0.98165
...

Contextual information modeling (Bayesian network-based profiles): The probabilities extracted in the previous phase are modeled as Bayesian network-based profiles. Table 2.3 shows the structure of Bayesian network-based profiles. Each profile represents the *activity context* of a specific attack. We also created a separate profile for normal activities. The attack name represents the name of the attack for which the profile is created. The feature name represents the name of the feature used in creating Bayesian network-based profiles. In the case of numerical values, lower bound represents the smallest value of the bin for numerical features that have been discretized. The upper bound represents the highest value of the bin for the numerical features which have been discretized. The categorical feature value represents the value of features which are nominal. It should be noted that the Bayesian network-based attack context profiles consist of all selected feature values, along with the probability of success of an attack over the given feature value; thus, Table 2.3 shows only a few features in each profile.

Contextual information usage (Bayesian network profile prediction at runtime): The BN-based profiles are used to examine runtime connections to find the probabilities of different attacks/normal activities. The combined probability of any label a_i , which can be either an attack or normal activity, is calculated as follows:

$$P(a_i|F) = (P(f_1|a_i) \cdot P(f_2|a_i) \dots, P(f_n|a_i) \cdot P(a_i)) / P(F) \quad (2.1)$$

where $P(a_i|F)$ is the probability of label a_i given the connection feature vector F , which consists of connection features $\{f_1, \dots, f_n\}$, under the assumption that the features are conditionally independent. The probability $P(f_i|a_i)$ reveals how often each feature value f_i occurs with a_i in the dataset; this value is retrieved from the corresponding Bayesian network-based profile. The basis for retrieval is the feature values possessed by this connection record. For numerical value attributes, the corresponding interval is looked upon in the Bayesian network-based profiles, and the probability associated with this feature interval is recorded. For categorical features, it is matched with the corresponding value in BN-based profiles, and the

associated conditional probability is recorded. $P(a_i)$ is the prior probability of a_i occurrence, whose value is extracted from training connections. Finally $P(F)$ is the probability of occurrence of the feature vector F . The system calculates the probability of each label (attack, normal activity), given the connection record features, to identify the label with maximum probability, which is passed as an initial prediction to the semantic network.

2.2.1.4 Using Semantic Networks (SN) for Detection of Known Attacks

A semantic network or net is a graphic notation for representing knowledge in patterns of interconnected nodes and arcs. Computer implementations of semantic networks were first developed for artificial intelligence and machine translation, but earlier versions have long been used in philosophy, psychology, and linguistics. Sowa gives a descriptive outline on the types and use of semantic networks in different disciplines [18, 19]. We used semantic networks to model the *relation* aspect of context. We created semantic networks as graphs where nodes represent attacks and edges model semantic relationships between such attacks. We used semantic networks to infer relevant attacks that are semantically related to the initial prediction produced by Bayesian networks, utilizing relationships which cannot be captured by simple classification techniques. A semantic network is created at the static phase, and each one of its nodes represents one of the 22 attacks in the dataset. Normal activity is also treated as a node in the semantic network. To avoid the intricate task of manually constructing and maintaining the semantic network, we adopt an automatic approach which consists of two phases. In the first phase, we created a similarity-based semantic network (i.e., the first mode network). In the second phase, we created a rule-enforced semantic network (i.e., the second mode network). The similarity-based or first mode network is a graph which identifies the degree of relevancy among attacks based on similarity values to connect such attacks using edges. The second mode network modifies the first mode and adjusts it by adding domain expertise; let us explain first how similarities between attack features are utilized to extract the contextual relationship among them to construct the first mode semantic networks.

Contextual information extraction (attack similarity calculation): Let $A = \{a_1, \dots, a_n\}$ be the set of network attacks, where each $a_p \in A$ is associated with a set of characterizing features forming an *attack feature vector* $V_{a_p} = \langle f_{a_{p_1}} \dots f_{a_{p_m}} \rangle$ where $f_{a_{p_j}}$ is the normalized frequency of feature j occurring with attack a_p . The feature values may have numerical and/or categorical values; hence, prior to creating an attack feature vector, we apply binning to convert the numerical features into bins. This is necessary since some similarity measures utilized in creating semantic networks require categorical values to calculate similarity among attacks. A sample of attack feature vectors is shown in Table 2.4. We determine the similarity between attacks as follows: first, we generate a single *universal feature vector* V (the first column in Table 2.4), which is the union of all

Table 2.4 Universal and attack feature vectors

Universal feature vector	Attack feature vector		...
	Guess_Password	Imap	
Dst_host_srv_serro_1	0.924528302	0.83333333	
Dst_host_srv_serro_2	0.037735849	0.043333333	
Dst_host_srv_serro_3	0.38867925	0.5	
Dst_host_srv_serro_4	0	0.166666667	
Dst_host_srv_serro_5	0.18867925	0.166666667	
Duration_1	1	1	
Duration_2	0	0	
Duration_3	0	0	
Duration_4	0	0	
REJ	0	0	
Rsto	0.849056604	0	
Rstos0	0	0	
...			

features extracted from attack feature vectors. The universal feature vector V is used to calculate the similarity between attack feature vectors V_{a_1}, \dots, V_{a_n} . We utilize several *similarity coefficients* to calculate the similarity among attacks. These coefficients serve as an effective tool for measuring the similarity among vectors. Lewis and Janeja [20] surveyed 35 different coefficients that can be used to find the similarity between vectors V_{ap} and V_{aq} . The inputs to similarity coefficient measures utilized in this work are binary vectors of zeros and ones; thus, we convert each attack feature vector V_{a_i} into a binary vector of zeros and ones by applying the absolute cutoff data transformation technique [21] using 0.5 as a cutoff point. We found experimentally that smaller cutoff points result in high similarity between attacks that belong to dissimilar categories.

To avoid the intricate task of manually constructing and maintaining the SLNs, we adopt an automatic approach which consists of two phases similarly to the process followed in Karabatis et al. [22]. In the first phase, we create similarity-based SLNs (first mode networks). In the second phase, we create rule-enforced SLNs (second mode networks). The similarity-based or the first mode SLN is a graph which identifies the degree of relevancy between nodes based on direct or indirect similarity relationship among them. The second mode SLN modifies the degree of relevancy in the first mode by applying domain knowledge in creating the relationships between nodes.

Domain-specific example: Let us consider an initial SLN (see Fig. 2.5) to explain how to generate relevance scores among its nodes. The nodes in the SLN represent some attacks in the DARPA dataset [23] which contains suspicious and benign connections. Suppose that the objective is to calculate the relevance score $rs'_{(Warezcclient \rightarrow Ftp\ write)}$ between the nodes representing the Warezcclient and Ftp_Write attacks.

Fig. 2.5 An initial SLN example for some attacks in DARPA dataset

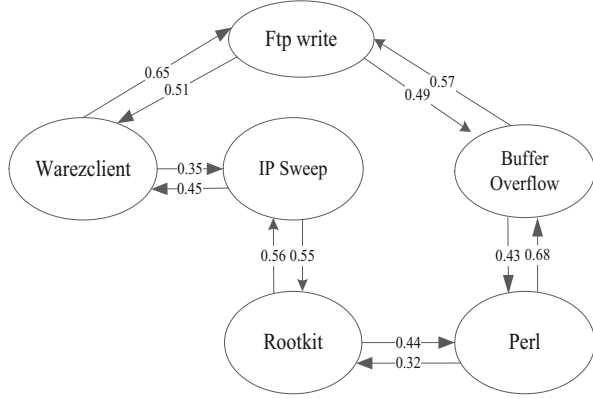


Table 2.5 DARPA dataset attack taxonomy

Attack category	Attack name
Denial of service (DoS)	Smurf, Neptune, Back, Teardrop, Pod, Land
Remote to local (R2L)	Warezcclient, Guess_Password, Warezmater, Imap, Ftp_Write, Multihop, Phf, Spy
User to root (U2R)	Buffer_Overflow, Rootkit, Loadmodule, Perl
Probe	Satan, Ipsweep, Portsweep, Nmap

Figure 2.5 shows the six attacks in the SLN and the transition probabilities between them. The relevance score $rs_{(Warezcclient \rightarrow Ftp\ write)}$ should be the maximum relevance score over all possible paths connecting Ftp_Write to Warezcclient. After two reasoning steps, the resulting $rs = 0.5$, however, the direct path $t_1(Warezcclient \rightarrow Ftp\ write)$ with length = 1 results in $rs = 0.65$; thus, the final relevance score $rs_{(warezcclient \rightarrow Ftp\ write)}$ is the maximum of these two rss which is 0.65.

Now let us adjust the rs from Warezcclient to Ftp_Write to generate a second mode SLN. We will use the DARPA dataset attack taxonomy to perform this adjustment. This taxonomy classifies some cyber-attacks in four categories as shown in Table 2.5. It has been created by security specialists who state most of the attacks which belong to the same category have stronger semantic relationships between them, compared to those that belong to different categories.

Since Ftp_Write and Warezcclient attacks belong to the same category (i.e., $\beta_{(n_i, n_j)} - rs_{(n_i, n_j)} > 0$), the new adjusted relevance score $rs'_{(warezcclient \rightarrow Ftp\ write)}$ is $0.65 + (|1 - 0.65| \times 0.66) \approx 0.88$. It is noteworthy to mention that domain knowledge stored in taxonomies is an important aspect in adjusting the values of rs . However, creating SLNs using taxonomies only is not sufficient to discover all relationships between nodes. Therefore, the *SBSLNs* are still needed, in particular to discover relationships that cannot be revealed using taxonomies (e.g., nodes in different categories). SLNs receive as input (starting node) the most probable node

that corresponds to a specific connection as predicted by the BN. This “initial predicted node” serves as the starting node in searching for other semantically related nodes based on a user-defined threshold (tr' threshold).

2.2.1.5 Attack Profiles (APs) to Discard Inaccurate Predictions

The SLNs include benign activity nodes that might be included in the predictions made to a specific connection based on the value of rs' threshold. Therefore, we may have a scenario where the set of predictions made to a specific connection includes both attacks and benign activities. It is then necessary to apply context-based filtering to avoid such scenario and filter out nonrelevant predictions. Therefore, we created attack profiles as a second prediction model; they consist of preconditions represented as features to trigger the occurrence of specific attacks, and they work as filters on the predictions of SLNs to discard a predicted attack n_i when the features of an incoming connection do not match the profile of attack n_i . When the connection is a benign activity, *APs* are supposed to remove all attack predictions made to such a connection. When the connection is an attack, the *APs* must remove all nodes that correspond to benign activities from the prediction list. Another purpose of *APs* is to identify attacks that are relevant to a particular context and to filter out some predictions made by SLNs that are “out of context,” thus reducing the amount of inaccurate predictions.

This work proposes a novel technique in which we utilize conditional entropy to create *APs*. The technique assumes that there could be several attacks that co-occur in similar contexts, and therefore it is fairly significant to predict them together. In addition, the proposed technique preserves the contextual relationship produced by SLNs. When *APs* are applied to the predictions produced by SLNs, they remove only the nonrelevant attacks. Lastly, the technique we proposed to create *APs* does not only help in filtering nonrelevant predictions but also identifying unknown attacks as we shall elaborate shortly. *APs* are initially created by measuring the conditional entropy of attacks based on the occurrence of each feature observed in a labeled dataset that consists of TCP connections. The *APs* denoted by AP_{n_i} for each attack n_i are created using the features that decrease the conditional entropy, thus decreasing the uncertainty about the occurrence of n_i . Conditional entropy in this context is the amount of information needed to infer the degree of uncertainty about n_i based on the occurrence of feature f . The conditional entropy for each single attack n_i on the condition of occurrence of f is

$$H(n_i|f) = - \sum_{j=1}^v P(n_i, f_j) \log_2 P(n_i|f_j) \quad (2.2)$$

where $P(n_i, f_j)$ is the joint probability of n_i and f_j (one possible value of f) and $P(n_i|f_j)$ is the conditional probability of n_i given f_j . In the context of attack

prediction from TCP connections, we utilize the features of labeled connections to find the conditional entropy for each attack n_i on the condition of occurrence of different values of feature f .

The sum of conditional entropy for n_i conditioned on all values of f is a measure of the local conditional entropy. Local conditional entropy measures the contribution of a particular feature to the occurrence of a particular attack. The lower the value of local conditional entropy of attack n_i given f , the better the feature f can predict the occurrence of attack n_i . Since SLNs predict several related attacks, our objective is to preserve the relation aspect of context, namely, the majority of the predictions produced by SLNs. We create *APs* by considering the fact that there could be several attacks co-occurring in similar contexts (e.g., both attacks have a goal to get root access). To be aware of such relationships between attacks, features that predict related attacks are considered in creating *APs*. If one has a partial knowledge about the target domain (e.g., the group of attacks that might co-occur together from a taxonomy of attacks or a domain expert knowledge), the conditional entropy for each group of related attacks (given such taxonomy of attacks) when conditioned on f can be calculated. This value indicates the global entropy which measures the importance of a feature for predicting the occurrence of each group of related attacks. The lower the value of global conditional entropy, the better the discrimination between different contexts. Let G_{n_i} denote the set of features which gave the lowest global conditional entropy values with the set $N' = \{n_1, \dots, n_k\}$ of related attacks (e.g., attacks that belong to similar category). Let L_{n_i} denotes the set of features which gave the lowest local conditional entropy values when conditioned on a specific attack $n_i | n_i \in N'$. The features in G_{n_i} and L_{n_i} convey better quality information; consequently, they are used in creating *AP*. The resulting *APs* are supposed to identify the combination of features that lead to one or more related attacks. Therefore some *APs* can be similar so that the corresponding attacks can be triggered together. The set of attacks which are semantically relevant is not supposed to be discarded by attack profiles. Accordingly, *APs* for semantically related attacks need to have similar content, i.e., an overlap between *APs* of related attacks is expected.

Domain-specific example: In this example, we demonstrate the process of creating attack profiles for the attacks in the DARPA intrusion detection dataset. Based on the domain expert categorization of the attacks shown in Table 2.5, the features in the set G_{n_i} with low global entropy values are formulated for each attack category as shown in Table 2.6. The number of features for each attack category is chosen according to a domain expert analysis conducted by Gupta et al. [24] who selected the most important features for each category based on attack semantics. Similarly, we select a new set of features L_{n_i} for each attack n_i . Table 2.7 shows some related attacks in the R2L attack category and the set of features ($G_{n_i} \cap L_{n_i}$) for each attack (e.g., *service, number of shells, number of files accessed, flags, is hot login, logged in* for Warezclient attack).

The value column ([V]) lists some values of these features. The occurrence of each feature with the corresponding attack is expressed as a conditional probability

Table 2.6 Important features in Gn_i for attacks in DARPA data/per category

Feature name	Attack category			
	Gn_{Probe}	Gn_{DoS}	Gn_{R2L}	Gn_{U2R}
Duration	✓	✓	✓	
protocol_type	✓	✓		
Service	✓	✓	✓	✓
Flag	✓	✓	✓	
src_bytes	✓	✓	✓	
wrong_fragment			✓	
Hot	✓			
num_failed_logins			✓	✓
logged_in			✓	
num_compromised			✓	
root_shell				✓
num_root			✓	✓
num_file_creations			✓	✓
num_shells			✓	✓
num_access_files			✓	✓
is_hot_login			✓	✓
is_guest_login			✓	
Count				
dst_host_same_srv_rate		✓		
dst_host_serror_rate		✓		
dst_host_srv_serror_rate		✓		
dst_host_rerror_rate		✓		
Number of features	6	9	14	8

Table 2.7 Sample attacks in R2L category and the corresponding features ($G_{n_i} \cap L_{n_i}$)

Feature attack	Service		Number of shells		Number of files accessed		Flag		Is hot login		Logged in	
	[V]	[P]	[V]	[P]	[V]	[P]	[V]	[P]	[V]	[P]	[V]	[P]
Warezcilent	Ftp_data	0.91	0	1	0	1	SF	1	0	0.96	1	0.98
	Ftp	0.09	1	0	1	0	–	0	1	0.04	0	0.02
Ftp_Write			0	1			SF	1	0	0.95	1	0.90
			1	0			–	0	1	0.05	0	0.10
Imap	Imap4	1	0	1	0	1			0	0.98	0	0.92
	–	0	1	0	1	0			1	0.02	1	0.08

in the column [P]. As shown in the table, the shaded feature values have been selected to create the attack profile entries for the corresponding attack. For instance, the final *AP* for Warezcilent and Ftp_Write attacks is expressed as a set of feature value pairs:

$$\begin{aligned}
& [Service_{is(ftp_data)}, No_of_Shell(0), No_of_files_accessed(0), flag('SF'), is_hot_login(0) \\
& \quad logged_in(1) \rightarrow warezclient] \\
& [No_of_Shell(0), flag('SF'), is_hot_login(0), logged_in(1) \rightarrow ftp_write]
\end{aligned}$$

Based on the similarity of their profiles, Warezcclient and Ftp_Write are contextually related. By looking at their profiles, the set of features which are selected to create AP_{ftp_write} is a subset of features used to create $AP_{warezcclient}$; consequently, there is a high probability that these two attacks are initiated under similar circumstances. APs are used to discard some predictions made by SLNs. The features of a connection c (for which a set of predictions N' are produced by SLNs) are matched with attack profiles. When the features of c do not match the profile $AP_{n_i} | n_i \in N'$, the attack n_i is discarded from N' . As such, the final objective is to keep attacks that are relevant and contextually related. For instance, if the set of final predictions contains *Warezcclient* attack as a prediction, there is a high probability that the same set contains *Ftp_Write* as a prediction as well. Incoming connections passing through SLNs but do not match any APs are deemed as benign activities.

2.2.1.6 Host Profiles (HPs) to Discard Inaccurate Predictions

The attack profiles do not convey any contextual features about the targeted hosts such as the patches applied against some attacks; thus, the domain experts must make assumptions about the missing information (e.g., the patches applied to the targeted) which may lead to incorrect predictions, namely, false positives. This situation needs to be addressed by gathering contextual features about the network hosts to filter out potential attacks that are not relevant, based on the current state on the target host. The information gathered represents the individuality features about the target hosts. This information can be collected using a patch management system or host monitoring tools. In our framework such information is represented as host profiles.

A host profile (*HP*) for a particular host h_d consists of logic facts about that host; each fact belongs to a specific type t . Example fact types are patches applied in response to the discovered vulnerabilities, applications on such host, its operating system, services running, etc. *HPs* are utilized in a process called individuality-based filtering through which these profiles are applied to filter out nonrelevant predictions produced by SLNs and APs .

As part of host-based filtering, our framework is aware of the existing facts about known attacks. An attack n_i is said to be out of context of host h_d , when all facts about such an attack that have a particular type t contradict with all host facts of the same type t . In other words, there is at least one requirement of the attack that is not satisfied on the targeted host. Once a context mismatch (described in Chap. 3) is detected, the attack is considered as nonrelevant and it is filtered out. Context mismatch is detected as follows: first, all facts $f_1(n_i) \dots f_m(n_i)$ of type t about a

specific predicted attack n_i are retrieved. The context mismatch between the facts about a specific attack n_i and those of host h_d occurs when for all facts of type t about the attack n_i , there is no relevant fact of the same type on the host h_d (e.g., the target host is patched against all known vulnerabilities that cause the attack). If the attack n_i is out of context of the host h_d , it needs to be discarded from the predictions. It is noteworthy to mention that the filtering extent of *HPs* depends mainly on the amount of information collected about the targeted hosts; therefore, there is no assumption about the completeness of such profiles. Individuality-based information is considered complete if all details about hosts are profiled in real-time manner; however, there are several reasons that lead to incomplete *HPs*. First, not all information is documented about each host, since this might lead to more overhead during the detection process. Second, even a small organization with a single server can expect to spend time reviewing a handful of critical patches per month. Therefore, if information about particular host is not available (e.g., patches against a specific attack n_i), our framework considers that h_d is vulnerable to the attack n_i , and therefore a context match is triggered if such an attack is in the prediction list. We create a synthetic individuality-based setting to generate *HPs* and use them to discard nonrelevant attacks. The details are discussed in the experiment chapter. The following attack scenario illustrates the applicability of *HPs* on top of SLNs and *AP*.

Attack Scenario 3.2: The Perl attack is a user to root (U2R) type of attack where attackers set the user ID to root in a Perl script and create a root shell. Usually the intention of an attacker is to perform other types of L2R such as DoS and buffer overflow attacks, specifically, on systems with the Apache server installed. One form of this attack allows local attackers to launch a symlink using an e-command to overwrite arbitrary files on the targeted hosts that have *Red Hat Linux* operating system. This attack is initiated when the attacker exploits the -e option vulnerability with reference *CVE-1999-1386*. If the same system has an Apache server, the “sioux” vulnerability (*CVE-1999-1199*) allows remote attackers to establish a denial of service back attack using GET requests containing a bulky number of “\” characters. Suppose that the target host h_d has *Red Hat Linux version 5.0 operating system* and an *Apache server version 1.3.1* installed on it. Suppose that there is no information available about patches against Perl on the target host. Let connection $c \subseteq h_s \times h_d$ be established between an attacker host h_s and a target host h_d at a particular time interval. The following facts are derived about the target host h_d at the time of connection: $hasOS(redhat\ linux5.0) \wedge hasApplication(Apache\ server\ 1.3.1) \wedge Patched(CVE-1999-1199) \wedge Patched(CVE-1999-0513)$. Additionally, the following facts are known about the Perl attack ($existOS(Redhat\ linux5.0)$ and $CausedByVulnerability(CVE-1999-1386)$), the DoS Back attack ($exist\ Application\ (Apache\ server\ 1.3.1) \wedge CausedByVulnerability(CVE-1999-1199)$), and the DoS Smurf attack ($CausedByVulnerability\ (CVE-1999-0513)$). Smurf attack is a method by which an attacker can send a moderate amount of traffic and cause a virtual explosion of traffic at the target host. Since we do not have any facts on the

Table 2.8 SLNs and context profile prediction for connection on host h_d

Initial prediction	Predictions of SLNs	$rs' \text{ Rootkit} \rightarrow n_j$	AP_{n_i} match	HP_{h_d} match	Possible attack?
Rootkit	Buffer_Overflow	0.80	Y	Y	Y
	Back	0.56	Y	N	N
	Perl	0.74	Y	Y	Y
	Smurf	0.45	N	N	N

victim host h_d about the patches applied against vulnerability *CVE-1999-1386*, h_d is considered vulnerable to Perl attack.

Table 2.8 shows the predictions that correspond to the connection c after the initial prediction is made (column 1); other relevant nodes are retrieved by SLN (column 2), and after attack/host profiles are applied to filter out the nonrelevant predictions (columns 4 and 5). The SLNs predicted several relevant attacks based on 0.4 rs' threshold. Such attacks are deemed relevant to “Rootkit” (i.e., the starting node in SLN) which is a U2R attack.

Out of these, only two attacks (Buffer_Overflow and Perl) are kept since their corresponding APs match the features of connection c . Back attack is filtered out since h_d is patched against this attack. Smurf attack is filtered out since its AP does not match the features of connection c . Buffer_Overflow attack is kept in the final prediction list since there are no sufficient evidences based on information in the profile of h_d that patches have been applied against this attack. The next chapter will discuss the creation of our prediction models using time and location features in a network environment with network flows.

2.3 Experiments and Analysis

In this set of experiments, we present the results of the experiments that have been conducted to test the effectiveness of our framework.

2.3.1 Experiments on Discovering Known Attacks in TCP Connections

In this section we describe the results of several experiments that measure the effectiveness of the prediction models when applied to identify known attacks from TCP connections. SLNs and context profiles are applied in a layered manner to identify semantically related attacks that occur in similar contexts. We first provide some preliminary details about the implementation of our prediction models, the setting under which the experiments are conducted, the datasets utilized for evaluation, the evaluation metrics used, and the types of experiments performed.

Implementation, dataset, and experiment settings: We developed a prototype system which includes the implementation of our prediction models using an Oracle database. Several data mining tools are used to perform the preprocessing steps in our approaches. In particular, we used Weka [25] and KNIME [26] data mining tools to perform preprocessing tasks such as feature selection along with extracting the probabilities of attacks and benign activities to create the BN-based classifier. We utilized PL/SQL to implement several other preprocessing steps such as the creation of feature vectors and similarity coefficients. For this set of experiments, we used the DARPA intrusion detection dataset [27], which is considered as a benchmark on which different methodologies can be evaluated. The DARPA dataset was created based on a simulation of a military network consisting of three “target” machines running various operating systems and services. Another three machines were then used to spoof different IP addresses to generate network traffic. Finally, there was a sniffer that recorded all network traffic using the TCP dump format. The dataset includes a wide variety of intrusions simulated in a military network environment. It also contains normal traffic. The TCP dump format of this dataset is summarized into connection sessions. Specifically, a connection is a sequence of TCP packets starting and ending at some well-defined times. A review that has been published in 2009 by Chih et al. [28] shows that the majority of intrusion detection approaches utilized this dataset as a benchmark to validate their methods. Due to privacy reasons, the same review demonstrates that very few research approaches use nonpublic datasets to validate their intrusion detection methods.

In this set of experiments, we utilize a pre-identified subset of 976,067 connections from this data, out of which 544,000 connections are used during the training phase to create our prediction models and 432,067 connections are used during the evaluation (runtime) phase. The training data contains 22 types of attacks, and each connection in the training data is labeled as either an attack or a benign activity. Connections that contain attacks belong to one of the four attack categories: DoS, R2L, U2R, or probing. Denial of service (DoS) attacks occur when the attacker tries to prevent legitimate users from using a service. Remote to local (R2L) attacks occur when the attacker does not have an account on the victim machine and tries to gain access. User to root (U2R) attacks occur when the attacker has local access to the victim machine and tries to gain a super user privileges. Probe attacks occur when the attacker tries to gain information about the target host. The connections in the evaluation part are also labeled as benign activities or attacks. The distribution of connections utilized during the training and evaluation phases is shown in Table 2.9.

The data preprocessing steps consist of two processes, discretization (binning) and feature selection. Most of the features in this dataset are numerical. Continuity in these feature values makes it harder to apply some data mining techniques. Therefore, the continuous features are automatically discretized using equal width binning approach [25]. The dataset has 41 features (including the label). This means that each connection is described by 40 features. In order to decrease the computations during evaluation time, we carried out feature selection using the

Table 2.9 Connections used in experiments from DARPA intrusion detection dataset

Category	Connections to create the prediction models	Connections to evaluate the prediction models
Benign	147,250	110,620
Probe	4,112	4,171
DoS	391,458	300,853
R2L	1130	16,354
U2R	50	69
<i>Total</i>	<i>544,000</i>	<i>432,067</i>

correlation-based feature selection method proposed by Hall [17] to yield the highly ranked features. The preprocessed data is used to create the prediction models described earlier. A BN classifier is created by extracting information to generate the probability of each attack and benign activity given the features extracted from the dataset. In BN, each prediction can be an attack or a benign activity, and it represents an unobserved parent node (i.e., the label) to be predicted at runtime. Once the conditional probabilities of nodes based on the occurrence of specific features are generated, they are stored as profiles and used to predict attacks/benign activities based on matching features of incoming connections.

Similarity-based SLNs (*SBSLNs*) are created based on the similarity information extracted from the dataset. Each node in the *SBSLN* represents one of the 22 attacks found in the dataset. Benign activity is also treated as a node in the SLNs. Our experiments were performed on a server with Intel Pentium D Dual Core 3.4 GHz CPU with 8 GB RAM running 64-bit Windows. Several types of experiments are conducted as part of this set of our experiments.

Precision, recall, and F-measure are used as evaluation metrics, as defined below:

$$P = \frac{TP}{TP + FP} \quad (2.3)$$

$$R = \frac{TP}{TP + FN} \quad (2.4)$$

$$F = \frac{(1 + \beta^2) \times PR \times DR}{\beta^2 \times (PR + DR)} \beta^2 = 1 \quad (2.5)$$

TP , FP , and FN represent the true positives, false positives, and false negatives, respectively. A TP occurs when a specific incoming connection is correctly recognized by the prediction model as an attack. TP s for a connection labeled as an attack are expected to be the actual attack (the label) and the attacks that are contextually related to it (e.g., both attacks targeted Ftp application). The latter are identified based on many real-world attack scenarios described in the common vulnerability exposure (CVE) database [29]. A FP occurs when a specific connection under evaluation is incorrectly recognized as an attack. A FN occurs when a specific incoming connection is incorrectly recognized by the system as a benign activity, but in reality it is an actual attack.

2.3.2 Applying SLNs and Context Profiles (CPs)

In this set of experiments, we measure the effectiveness of context profiles (both attack profiles (*APs*) and host profiles (*HPs*)) when they are applied as filters to remove potential incorrect predictions produced by SLNs. Some nodes retrieved by SLNs can be false positives, that is, the connection itself is a benign activity but predicted as an attack. This scenario occurs when the set of predictions that correspond to a specific connection contains both attacks and benign predictions. In addition, a prediction is incorrect when the set of predictions generated by SLNs contains an attack that is not relevant based on the profiles of the targeted hosts. One main issue in this experiment is the creation of *HPs*. Different hosts have different sets of patches, application types, operating systems, etc. Host vulnerabilities and the required patches can be discovered using vulnerability management tools. Additionally, various tools can be used to create organizational maps. However, this task is beyond the scope of this work. Instead, we created a testbed environment that resembles a realistic organization based on information extracted from the connections in the dataset we experimented on. There are some metadata features in each connection that indicate facts or evidences that we utilized to create host profiles. We utilize the connection features (service, flags, and protocol type) to create host profiles (*HPs*) given the constraint that no knowledge is available about the type (label) of such a connection. *First*, we randomly assign each target host h_d an identifier which is its IP address. *Second*, we extract the metadata features from connections destined for host h_d , and we identify all possible attacks that may contain these features. The possible attacks are selected from each attack category to make sure that the host profile that corresponds to h_d contains facts about attacks in all categories. *Third*, we utilize the common vulnerability exposure (CVE) database to query about all possible configurations that pertain to these attacks. The collected data is used to generate attack facts and host facts. The host facts include patches against some attacks in each attack category. Given that our framework has no knowledge about the type of an incoming connection, the *HPs* might discard any prediction generated by SLNs, including some relevant predictions. Note that the majority of the remaining predictions, after *APs* are applied, are expected to be relevant based on context. Additionally, few predictions are expected to be irrelevant but have not been discarded by *APs*. Since the probability of removing a relevant or a nonrelevant prediction by *HPs* is the same, and there are more relevant predictions at this point, there is a higher probability not to discard them by *HPs* (given that such profiles contain partial information about the targeted hosts). The advantage of using this method is to create hypothetical yet realistic host contexts that can be used to filter out some predictions based on what is known about the targeted hosts. For this reason we used *HPs* as a complementary layer to validate our detection approach. We created 100 different *HPs* based on the data collected using the protocol, service, and flag features. Given an incoming connection that targets h_d , *HPs* discard attack predictions made by SLNs if the required patches have already been applied to that host. The attack predictions about which

Fig. 2.6 Average precision for 2nd mode SLNs and context profiles

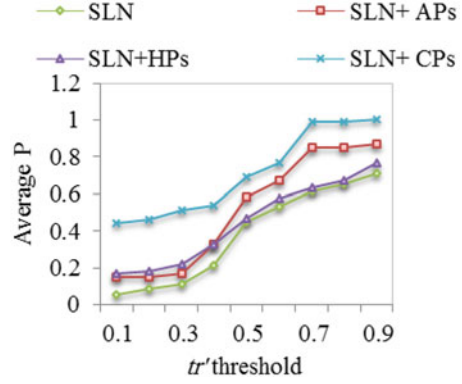
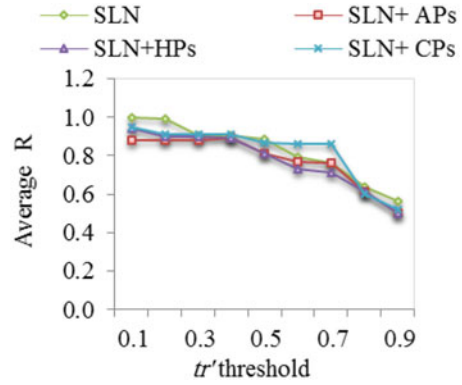


Fig. 2.7 Average recall for 2nd mode SLNs and context profiles



the host profile has no knowledge are kept in the final prediction list, and they are considered true positives. The best performing 2nd mode SLNs created using Anderberg (AD) similarity coefficient are used in this experiment with one or both types of context profiles. In particular, we evaluate (1) SLNs without any context profiles, (2) *APs* when applied as filters on the predictions of SLNs (SLN+*APs*), (3) *HPs* when applied to the predictions of SLNs (SLN+*HPs*), and (4) *APs* and *HPs* when applied in a compound fashion to the predictions of SLNs (SLNs+*CPs*). We added the following constraint when *HPs* are applied on top of SLNs and *APs*: if the incoming connection does not match any *AP*, the connection is deemed as a benign activity and we skip the *HPs*. Finally, the predictions of SLNs, which still remain after *APs* and *HPs* are applied, are the ones that characterize the incoming connection; as a result the target host h_d is deemed vulnerable to them.

Figures 2.6 and 2.7 show the average precision and recall results in this set of experiments. Figure 2.6 illustrates that utilizing *APs* on top of SLNs improves precision. *APs* are quite efficient in differentiating between the context of suspicious and benign activities. The improvement achieved when *APs* are applied to the predictions made by SLNs is explained by the reduction achieved in false positive rate. As noticed from the results shown in Fig. 2.6, the *HPs* without *APs* are not very

effective when applied to the predictions made by SLNs, resulting in an average precision of 0.63. This is interpreted by predicting some benign connections as attacks (i.e., false positives). The reasons for this result are twofold: (1) these connections were incorrectly predicted as attacks by SLNs, and (2) the context of the targeted hosts also matches some of these predictions. In the end, these incorrect predictions made by SLNs are not discarded by *HPs*. This situation explains the benefits of utilizing *AP* profiles to recognize false positives before *HPs* are applied. When all layers are used together, the SLNs and both context profiles (shown as SLNs+ CPs line in figures), the precision is almost 0.98 at the 0.7 tr' threshold compared to 0.62 when SLNs are used without any context profiles and 0.63 when only *HPs* are applied on top of SLNs.

Figure 2.7 shows the average recall in this experiment. SLNs without any profiles have a slightly higher recall value at small tr' threshold; however, the difference is not very high compared to using SLNs with context profiles. The SLNs have an average recall of 1 at 0.1 and 0.2 thresholds. As observed, the recall of SLNs and other context profiles starts to decline after 0.7 level of tr' threshold. This decline indicates that the SLNs start missing some relevant predictions beyond the 0.7 threshold (they become too selective). The recall for SLNs is 0.76 at 0.7 threshold compared to 0.71 when SLNs are used with *HPs* and 0.86 when both context profiles are utilized on top of SLNs. The relatively higher recall of CPs at higher threshold values (0.5–0.8) can be interpreted by the layered filtering mode (applying first *APs*, followed by *HPs*), making them more effective in handling connections that represent benign activities versus SLNs without any context profiles. Overall, the layered filtering mode achieves good precision and recall values. Figure 2.8 shows the results of F-measure values, when context profiles are used on top of SLNs. The best F-measure value when *APs* are applied on top of *SLNs* is 0.8, and it is achieved at 0.7 tr' threshold. By contrast, this value is approximately 0.68 when SLNs are used without any context profiles. Overall, when both context profiles are applied in a layered manner on top of SLN, the *F*-measure value is approximately 0.92, and it is obtained at the 0.7 tr' threshold. As observed in

Fig. 2.8 Average F-measure for 2nd mode SLNs and context profiles

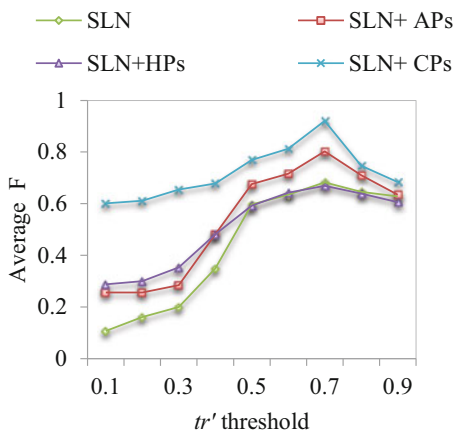


Fig. 2.8, the F-value obtained by applying the *HPs* on the results of *SLNs* is about 0.66 at 0.7 threshold which is lower than when *APs* is applied. *In summary, the HPs show some limitations in handling benign connections when no APs are used. When both context profiles are used together, a higher precision is achieved resulting in higher F-measure values.*

2.3.3 Factors Affecting the Effectiveness of *APs*

The effectiveness of *APs* which are utilized to filter out nonrelevant predictions might be affected by several factors. *First*, the effectiveness of *APs* might be affected by the distribution of attacks and benign activities in the training data used to create them. In particular, the proportion of attacks to benign activities in the training data used to create *APs* might affect the capability of the resulting profiles to discriminate between the context of cyber-attacks and benign activities at runtime. *Second*, the effectiveness of *APs* might be affected by the number of features utilized in creating them. This section demonstrates the results of two experiments that we conducted to compare the effectiveness of *APs* when changing these two factors. In the first experiment, two training subsets with different distribution of attacks to benign activity are utilized to create *APs*. The resulting *APs* from each subset are then applied to the predictions of *SLNs*, and their effectiveness is compared. In the second experiment, we focus on varying the number of features to create *APs*, and we then measure the effect of such changes on the effectiveness of these profiles. Let us give more details about the creation of *APs* before demonstrating the settings of both experiments. In general, our objective during the creation of *APs* is to decrease the overlap between profiles that correspond to attacks which occur in different contexts. An observation by Gupta et al. [24] on the same dataset reveals that attacks which belong to different categories occur under different preconditions. Thus, they presumably must have different attack profiles. As a second observation, we notice some attacks belonging to the same category but having different features. Thus, although some attacks belong to the same category, they do not necessarily occur under identical circumstances.

The methodology we utilize to create *APs* using the DARPA intrusion detection dataset takes these two observations into consideration. Besides the features that have high weights (low global entropy) in each attack category, we focus on features that characterize the context in which a specific attack occurs (i.e., features that have low entropy with each attack).

The first observation has been partially considered by Gupta et al. [24] where the authors utilize a local (per attack category) feature selection method which iteratively constructs feature conjunctions to increase the conditional log-likelihood when added to a conditional random field attack prediction model. The direction in Gupta et al. work is to select a specific set of features for each attack category to create an attack prediction model that differentiates between different contexts. Similarly, for the creation of *APs*, we select the features from each attack category

Table 2.10 A subset S_2 of connections selected from DARPA dataset

Attack category	Attack type	Number of connections
Remote to local (R2L)	dict, dict_simple, Ftp_Write, guest, Multihop, Phf, Spy, warez, Warezclient, Warezmaster	2723
Denial of service (DoS)	Land, syslog, Teardrop	1124
User to root (U2R)	Eject, eject-fail, ffb, ffb_clear, format_fail, format_clear, format, imap, load_clear, load_clear, Loadmodule, perl_clear, perlmagic, Rootkit	81
Benign	–	174,873

so that the resulting APs of attacks in different categories are dissimilar. Prior to running the first experiment, we utilize a new subset S_2 of connections from the DARPA intrusion detection data in addition to the main subset S_1 utilized earlier in our previous experiments. The majority of connections in S_2 are benign activities and a few attack connections that belong to the three categories shown in Table 2.10. This subset has been extracted from the TCP dump format of DARPA dataset by Perona et al. [30], and it consists of attacks in the categories DoS , $R2L$, and $U2R$. There are no flooding (probe) attacks in this subset. The connections in S_2 are divided into training and evaluation parts. The training part is used to create AD -based $SLNs$ and APs . To create APs , the connections which contain attacks are divided into disjoint categories ($c_1 = Probe$, $c_2 = DoS$, $c_3 = R2L$, $c_4 = U2R$ for the connections in the first subset S_1) and ($c_1 = DoS$, $c_2 = R2L$, $c_3 = U2R$ for the second subset S_2). For each category c_i , we created the set $Gc_i = \{f_1, \dots, f_m\}$ which contains the features that give the lowest global conditional entropy given the attacks in that category. For each category, we selected about the same number of features to those selected by Gupta et al. [24] (see Sect. 4.5). The features which contribute to the occurrence of a specific attack n_i are also considered in the creation of attack profiles. Therefore, we selected a new set of features $Ln_i = \{f_1, \dots, f_n\}$ which have the lowest local conditional entropy for a specific attack n_i . We then start creating AP_{n_i} for each attack using the features in $Gc_i \cap Ln_i$. For the first experiment, the incoming connections selected from S_1 and S_2 are initially processed by a BN classifier, and then they are passed to $SLNs$ to retrieve additional relevant predictions and APs to filter out the nonrelevant ones.

The values of precision, recall, and F-measures are reported in Figs. 2.9, 2.10, and 2.11. The figures clearly illustrate better effectiveness of APs when more attack connections are used to create the profiles ($SLNs+APs_{S_1}$). The average precision, recall, and F -values are better when the subset S_1 is used in this experiment. The reason we observe relatively lower precision and recall results with S_2 is related to the strength of relationships between benign and attack nodes in the $SLNs$ created using fewer attack connections (using the subset S_2). In these networks, attacks and benign activity nodes are not well separated. While the 2nd mode $SLNs$ are expected to lower the probability of this problem by adjusting relationships between nodes, some false positives are still expected resulting in lower precision values.

Fig. 2.9 Average precision for AD-based 2nd mode SLNs and APs using S1 and S2

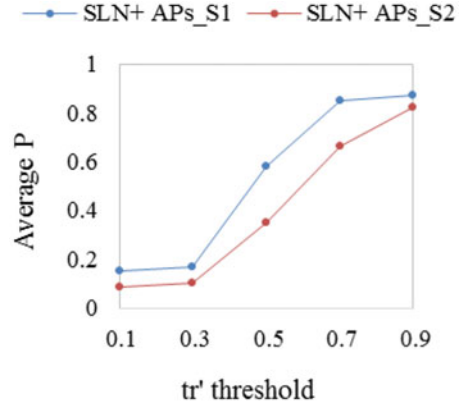
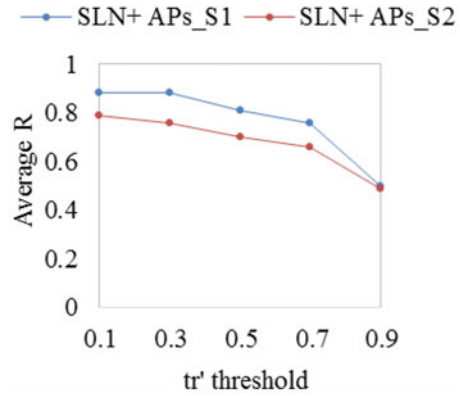


Fig. 2.10 Average recall for AD-based 2nd mode SLNs and APs using S1 and S2



The primary reason is related, however, to the quality of features utilized in creating *APs*. When fewer connections from each attack category are utilized in creating *APs*, the boundaries between contexts become less. With this limitation, the selected features become less discriminatory across different contexts, thus, affecting the overall detection rate. The second experiment measures the effect of the number of features selected for each attack (the number of features in the set L_{n_i}) to create its profile on F-values. We experimented with a range of 4–14 features. The experiment is conducted on both subsets of connections S_1 and S_2 . Figure 2.12 shows the effectiveness of SLNs and *APs* in terms of F-values when changing the number of features to create *APs*. It can be observed that utilizing six to eight features from the set L_{n_i} produces relatively better results. For computational efficiency, we only utilize the six highly ranked features from L_{n_i} to create *APs* for attacks in S_1 and seven features to create *APs* for attacks in S_2 . Several profiles are found similar when the connections in the subsets S_1 and S_2 are used to create *APs*. In the end, each attack is identified using one profile.

Fig. 2.11 Average F-measure for AD-based 2nd mode SLNs and APs using S1 and S2

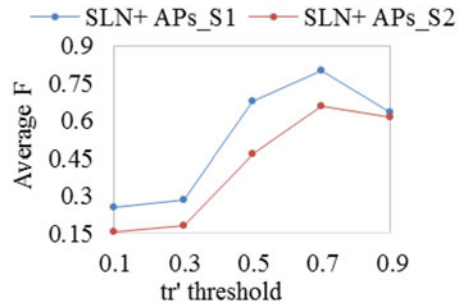
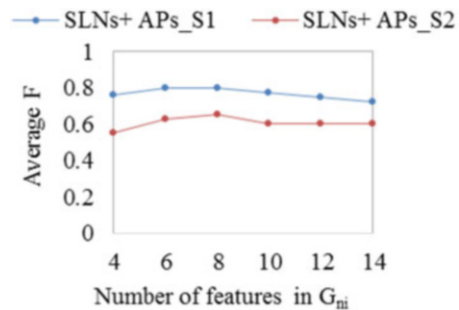


Fig. 2.12 The effects of number of features used in creating APs on F-measure values



References

1. Noel, S., Jajodia, S.: Understanding complex network attack graphs through clustered adjacency matrices. In 21st Annual Computer Security Applications Conference AZ, USA, 5–9 December 2005, pp. 10 pp.–169. doi:[10.1109/CSAC.2005.58](https://doi.org/10.1109/CSAC.2005.58) (2005)
2. Noel, S., Robertson, E., Jajodia, S.: Correlating intrusion events and building attack scenarios through attack graph distances. In: 20th Annual Computer Security Applications Conference (CSAC'04), Tucson, AZ, USA, pp. 350–359. doi:[10.1109/CSAC.2004.11](https://doi.org/10.1109/CSAC.2004.11) (2004)
3. Noel, S., Sushil, J., O'Berry, B., Jacobs, M.: Efficient minimum-cost network hardening via exploit dependency graphs. In: Proceedings. 19th Annual Computer Security Applications Conference, Orlando, FL USA, 8–12 December 2003, pp. 86–95. doi:[10.1109/CSAC.2003.1254313](https://doi.org/10.1109/CSAC.2003.1254313) (2003)
4. Ritchey, R., O'Berry, B., Noel, S.: Representing TCP/IP connectivity for topological analysis of network security. In: Proceedings. 18th Annual Computer Security Applications Conference, Las Vegas, Nevada, pp. 25–31. doi:[10.1109/CSAC.2002.1176275](https://doi.org/10.1109/CSAC.2002.1176275) (2002)
5. Mathew, S., Upadhyaya, S., Sudit, M., Stotz, A.: Situation awareness of multistage cyber attacks by semantic event fusion. In: Military Communications Conference, 2010—milcom 2010, San Jose, CA, 31 October 2010–3 November 2010, pp. 1286–1291. doi:[10.1109/MILCOM.2010.5680121](https://doi.org/10.1109/MILCOM.2010.5680121) (2010)
6. Leung, K., Leckie, C.: Unsupervised anomaly detection in network intrusion detection using clusters. In: Proceedings of the Twenty-eighth Australasian conference on Computer Science, Newcastle, NSW, Australia. Australian Computer Society, Inc., pp. 333–342 (2005)
7. Portnoy, L.: Intrusion detection with unlabeled data using clustering. Data Mining Lab, Department of Computer Science, Columbia University (2001)
8. Song, J., Takakura, H., Kwon, Y.: A generalized feature extraction scheme to detect 0-day attacks via IDS alerts. In: Proceedings of the 2008 International Symposium on Applications

- and the Internet, Turku, Finland, pp. 55–61. 1442004: IEEE Computer Society. doi:[10.1109/saint.2008.85](https://doi.org/10.1109/saint.2008.85) (2008)
9. Hendry, G.R., Yang, S.J.: Intrusion signature creation via clustering anomalies. In: Proc. of SPIE Bellingham, WA, vol. 6973, pp. 69730C–69731 (2008)
 10. Kuchimanchi, G.K., Phoha, V.V., Balagani, K.S., Gaddam, S.R. Dimension reduction using feature extraction methods for Real-time misuse detection systems. In: Proceedings of the Fifth Annual IEEE SMC Information Assurance Workshop, West Point, New York. IEEE, pp. 195–202 (2004)
 11. Liu, G., Yi, Z., Yang, S.: A hierarchical intrusion detection model based on the PCA neural networks. *Neurocomputing* **70**(7–9), 1561–1568 (2007). doi:[10.1016/j.neucom.2006.10.146](https://doi.org/10.1016/j.neucom.2006.10.146)
 12. Siraj, M.M., Maarof, M.A., Hashim, S.Z.M.: Intelligent clustering with PCA and unsupervised learning algorithm in intrusion alert correlation. In: Fifth International Conference on Information Assurance and Security (IAS '09). Xi'an, China, 18–20 August 2009, vol. 1, pp. 679–682. doi:[10.1109/IAS.2009.261](https://doi.org/10.1109/IAS.2009.261) (2009)
 13. Zheng, K., Qian, X., Wang, P.: Dimension reduction in intrusion detection using manifold learning. In: International Conference on Computational Intelligence and Security (CIS'09). Beijing, China, vol. 2, pp. 464–468. IEEE (2009)
 14. Li, X.-B.: A scalable decision tree system and its application in pattern recognition and intrusion detection. *Decis. Support Syst.* **41**(1), 112–130 (2005). doi:[10.1016/j.dss.2004.06.016](https://doi.org/10.1016/j.dss.2004.06.016)
 15. Tavallaee, M., Bagheri, E., Wei, L., Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In: IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA'09), Ottawa, ON, 8–10 July 2009, pp. 1–6. doi:[10.1109/CISDA.2009.5356528](https://doi.org/10.1109/CISDA.2009.5356528) (2009)
 16. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.: NSL-KDD Dataset. <http://iscx.ca/NSL-KDD/> (2009)
 17. Hall, M.A.: Correlation-Based Feature Selection for Machine Learning. The University of Waikato (1999)
 18. Sowa, J.F.: Principles of Semantic Networks. Morgan Kaufmann Pub., San Mateo, CA (1991)
 19. Sowa, J.F.: Semantic networks. *Encyclopedia of Cognitive Science* (2006)
 20. Lewis, D.M., Janeja, V.P.: An empirical evaluation of similarity coefficients for binary valued data. *Int. J. Data Warehous. Min.* **7**(2), 44–66 (2011)
 21. Pensa, R.G., Leschi, C., Besson, J., Boulicaut, J.F.: Assessment of discretization techniques for relevant pattern discovery from gene expression data. In: Proceedings ACM BIOKDD, vol. 4, pp. 24–30 (2004)
 22. Karabatis, G., Chen, Z., Janeja, V.P., Lobo, T., Advani, M., Lindvall, M., et al.: Using semantic networks and context in search for relevant software engineering artifacts. *J. Data Semant.* **5880**(1), 74–104 (2009). doi:[10.1007/978-3-642-10562-3_3](https://doi.org/10.1007/978-3-642-10562-3_3)
 23. Lippmann, R. MIT Lincoln Laboratory KDD Attack Taxonomy. <http://www.ll.mit.edu/misison/communications/cyber/CSTcorporation/ideval/docs/>. Accessed 05 June 2014 (2014)
 24. Gupta, K.K., Nath, B., Kotagiri, R.: Layered approach using conditional random fields for intrusion detection. *IEEE Trans. Dependable Secure Comput.* **7**(1), 35–49 (2010)
 25. Frank, E., Smith, T., Witten, I.: Weka A machine learning software. Machine Learning Group at the University of Waikato. <http://www.cs.waikato.ac.nz/ml/weka/> (2014)
 26. Wiswedel, B., Ohl, P., Gabriel, T.: KNIME: Kontaz Information Miner. <http://www.knime.org/> (2014)
 27. Granchelli, D.: DARPA intrusion detection datasets. Massachusetts Institute of Technology (MIT): MIT Lincoln Laboratory (1999)
 28. Tsai, C.F., Hsu, Y.F., Lin, C.Y., Lin, W.Y.: Intrusion detection by machine learning: a review. *Expert Systems with Applications* **36**(10), 11994–12000 (2009)
 29. Lawler, S., Meunier, P. Common vulnerabilities and exposures. <http://cve.mitre.org/>. Accessed 07 October 2014 (2012)
 30. Perona, I., Gurrutxaga, I., Arbelaitz, O., Martín, J.I., Muguerza, J., Pérez, J.M.: Service-independent payload analysis to improve intrusion detection in network traffic. In: Proceedings of the 7th Australasian Data Mining Conference, SA, Australia, pp. 171–178. Australian Computer Society, Inc. (2008)

Information Fusion for Cyber-Security Analytics

Alsmadi, I.; Karabatis, G.; Aleroud, A. (Eds.)

2017, X, 379 p. 85 illus., 61 illus. in color., Hardcover

ISBN: 978-3-319-44256-3