Implications of IDS Classification on Attack Detection

Magnus Almgren Emilie Lundin Erland Jonsson
Department of Computer Engineering, Chalmers University of Technology
SE-412 96 Göteborg, Sweden
{almgren,emilie,erland.jonsson}@ce.chalmers.se

Abstract

Accurate taxonomies are critical for the advancement of research fields. Taxonomies for intrusion detection are not fully agreed upon, and further lack convincing motivation of their categories. Most classifications do not even answer the question of which detection paradigm is suitable for detecting a specific attack. We have surveyed and summarized previously made taxonomies for intrusion detection. Focusing on categories relevant for detection methods, we have redefined two commonly used attributes found in many surveys: the *type of reference model* used and the *creator of the reference model*. Using our framework, the range of previously used terms can easily be explained and our continuous scale reflects the progress made in the field. We have studied the usefulness of these two concepts by two empirical evaluations. Firstly, we used the taxonomy to create a survey of existing research IDSs, with a successful result, i.e. the IDSs are well scattered in the defined space. Secondly, we evaluated the effectiveness of three classified detection methods for detecting specific attacks through a case study. Our preliminary findings indicate that the attribute reflecting the *type of reference model* is well motivated, while the attribute reflecting the *creator of the reference model* does not seem to have as much support, and should possibly be replaced.

Keywords: intrusion detection, taxonomy, classification, detection methods

1 Introduction

It is a well-known fact that the research in a field greatly benefits from a good taxonomy and hence a good classification. A shared vocabulary enables efficient communication and a shared classification may direct future research into open areas, i.e. holes, in the classification. In some cases, a classification not only helps cluster the field but may also well reflect some intrinsic value, which in turn leads to a refinement of the models in the field.

There has been several defined taxonomies, classifications and subsequent surveys for intrusion detection. The goals of these efforts have also been quite diverse; some only try to survey the field and find it easier with labels on the systems, while others try to use the taxonomies for a deeper understanding or to guide future research efforts. Despite these previous efforts, intrusion detection still lacks a widely applicable and accepted taxonomy. This may in part be because of it being a young research field, part of it being fast-paced and maybe part of it owing to its inherent complexity.

In this paper, we survey several taxonomies for attacks and intrusion detection systems (IDSs). In our view, many of the existing taxonomies have used different terms for the same or similar concepts. We highlight two of the concepts relevant for detection methods and present a discussion of how previously used terms fit into our framework. We conclude the first part of the paper with a classification of well-known research IDS prototypes according to the consolidated framework.

In the second part of the paper, we further investigate the two chosen parameters that existing taxonomies have used. We find that these categories have not been sufficiently motivated, and even if they well divide the area of intrusion detection (as shown in the first part) a taxonomy should be applicable beyond a simple division of the field. For that reason, we perform a case study with two attacks and three detection methods to evaluate the detection capabilities of each method in relation to specific attacks.

The organization of this paper is as follows. In Section 2 we survey several existing classifications relevant for intrusion detection. In Section 3 we summarize some of the terms used in the surveys, and then consolidate them according to their common underlying concept. The section is concluded with a classification of well-known IDSs according to two orthogonal concepts. In Section 4 we see if the concepts related to detection methods have any relevance when it comes to finding out the suitability of a detection method for detecting particular attacks. We present two attacks and three detection methods for a case study. After an informal evaluation, we conclude the paper in Section 5.

2 Classifications Relevant for Intrusion Detection

In this section, we present some of the previous taxonomies and classification efforts relevant for intrusion detection. This list is meant to summarize previous efforts, and should not be seen as an exhaustive enumeration of all previously made classifications. We have divided the classifications based on their main area of focus, but a few efforts span several areas and are thus reiterated in several sections below.

2.1 Intrusion Detection System Classification

Debar et al. [1] probably did the first taxonomic survey of intrusion detection system research. The main categories used in their taxonomy of IDSs are *detection method*, *behavior on detection*, *audit source location*, and *usage frequency*. These categories are further divided into two possible sub-categories. For example, the detection method is further refined as either behavior-based or knowledge-based detection.

Table 1: Axelsson's detection principles

Anomaly	Self-learning	Non time series
		Time series
	Programmed	Descriptive statistics
		Default deny
Signature	Programmed	State modeling
		Expert-system
		String-matching
		Simple rule-based
Signature inspired	Self-learning	Automatic feature selection

Axelsson [2] also provides a survey and taxonomy of intrusion detection systems. His approach is more fine-grained than the one by Debar et al. The taxonomy treats detection principles and system characteristics separately.

The detection principles are divided into *anomaly*, *signature*, and *signature-inspired*, with the sub-categories *self-learning* and *programmed*. Additional layers of sub-categories are used, depending on the specific class. Table 1 shows an overview of the detection principles. The system categories used in the taxonomy are *time of detection*, *granularity of data processing*, *source of audit data*, *response to detected intrusions*, *locus of data processing*, *locus of data collection*, *security*, and *degree of interoperability*.

The contribution of the two IDS surveys mentioned above is mainly to give a systematic overview of IDS research. In addition, Axelsson uses his survey to point out current trends in IDS research.

A European Union funded project, entitled the MAFTIA project [3], attempts to form a detailed taxonomy over IDSs, that is then used for determining their detection scope and how different IDSs can be used to complement each other. The taxonomy aims at describing the capabilities of an IDS with respect to the analysis of activities (attacks and related events) and it includes a large number of attributes for describing both the detector (analysis mechanism) and the sensors (information sources) of the IDS. It should be possible to match the IDS description to any activity description and determine whether the IDS generates an alarm for that activity. Unfortunately, there are no detailed examples of IDS descriptions in this report. We are going to focus on the detection method in this paper and for that reason we limit our discussion to the detection component categories and attributes found in the MAFTIA project. Alessandri et al. [3] start by separating the attributes into *generic characteristics*, *data preprocessing*, and *instance analysis*. Instance analysis, which determines what kind of detection the IDS does, is then further divided into such sub-categories as *single-instance analysis* and *cross-instance analysis*. Subsequent division leads to the *analysis-level* category, which can take the possible values of *basic analysis*, *logic verification*, and *semantic verification*. They also consider the attribute *analysis technique*, which are divided into *timing analysis*, *information item analysis*, *data category analysis*, *sequence analysis*, and *statistical analysis*. The attributes for these categories are, e.g., string matching, advanced string matching, regular expression, and size verification.

Another classification is proposed by Halme and Bauer [4], where they use the classes of anomaly detection and misuse detection to describe IDSs.

Ko et al. [5] have not done a formal taxonomy, but divide IDSs into systems using *anomaly detection*, *misuse detection*, and *specification-based monitoring*.

There are two more significant surveys of the IDS field [6, 7], but both these efforts' main focus is to describe the existing IDSs to provide information for someone who wants to compare and deploy an IDS and not to create a taxonomy with a classification. Many of the characteristics they study are about usability and management of the IDSs and no attempt has been made to say anything about the detection coverage. For that reason, the surveys of Jackson [6] and Kvarnström [7] are not discussed in further detail.

2.2 Attack Classification

Going beyond the IDSs to the attacks that are the reason for their existence, we have several additional classifications. Kumar [8] includes a classification scheme for *intrusion signatures* with hierarchical classes, based on the complexity of matching the attack (signature). He divides the signature types into four categories: *existence patterns*, *sequence patterns*, *RE patterns*,[1] and *other patterns*.

The MAFTIA project [3], presents a fine-grained activity taxonomy to describe an attack in such detail that it can be matched against the detection capabilities of an IDS. For example, there are attributes for the affected object, such as storage device, memory, operating system core, file system object, or process. There are also attributes for the method invocation type, e.g. object creation, object execution etc.

Lindqvist [9] extends the classification scheme for *intrusion techniques* by Neumann and Parker [10]. The main categories in Lindqvist's classification are *bypassing intended controls*, *active misuse of resources*, and *passive misuse of resources*. Each main category has a number of sub-categories and the paper also includes a taxonomy of intrusion results.

Mell [11] tries to capture statistics of attacks based on special characteristics, such as *script goal*, *target type*, *attacker platform*, and *transmission method*. He also mentions that the database of classified attacks can be used for two different purposes: forensics, when creating a list of possible attacks that may have compromised a penetrated system, and as a search tool for specific attack scripts.

There are also a series of classifications focused on the intended result of the attacks. For example, the DARPA 1999 intrusion detection evaluation experiment used five main types of attacks: *denial of service*, *remote to user*, *user to superuser*, *surveillance/probing*, and *data attack*. More information is found in [12].

Finally, worth mentioning is also the seminal paper by Denning [13] with a list of attacks. It is not a classification, but a description of different types of attacks that is then used when designing the intrusion detection system described in the paper.

[1] RE stands for Regular Expressions, equivalent to regular grammars defi ning regular languages in the Chomsky hierarchy.

2.3 Other Classifications

There are also classifications concerning the *attackers* and the underlying *vulnerabilities* (or flaws) that are being exploited. For example, Landwehr et al. [14] and Krsul [15] have done vulnerability taxonomies and classifications. Their objective is to document flaws in the hopes that knowledge of how systems fail helps build systems that can better resist failure. Howard [16] has classified the attackers of computer systems and their objectives.

The attributes found in these efforts could be relevant for intrusion detection, but it is not within the scope of this paper.

2.4 Relevance of Detection Method Classification

Few of the taxonomies discuss in any detail what each defined detection method implies for the ability to detect different types of attacks. However, brief discussions can be found in some papers. For example, Ko [5] says that anomaly detection provides a method to detect penetrations without specific knowledge about the operating system or its security flaws and that it is the only viable technique to detect masqueraders. It may, however, be difficult to establish normal behavior and to set thresholds for what is considered anomalous. Misuse detection, on the other hand, is good at detecting known attacks but not at detecting novel attacks.

Axelsson [2] talks about the high-level categories of *well-known intrusions*, *generalizable intrusions*, and *unknown intrusions*. Well-known intrusions are easily detected by signature-based systems, and the signature-inspired systems are more sophisticated in that they consider both normal and attack behavior. More advanced pattern matching systems may detect generalizable intrusions. Finally, the anomaly detection systems are the only ones that may have a chance at detecting previously unknown intrusions.

3 IDS Terminology Revisited

Current classification efforts have several shortcomings, mainly because they are not widely accepted and their chosen categories have not been sufficiently motivated. Many of the surveyed taxonomies in Section 2 have used a similar terminology, but they differ when it comes to the details. Table 2 summarizes some of the terms previously used.

Table 2: Detection Method Terminology used in Different Surveys

Author	*Terms used in Surveys*		
Halme and Bauer [4]	anomaly		misuse
Debar et al. [1]	behavior-based		knowledge-based
Ko [5]	anomaly	specification-based	misuse
Lindqvist [9]	dynamic knowledge default deny		static knowledge default permit
Axelsson [2]	anomaly self-learned	signature-inspired	signature programmed
MAFTIA [3]	*no corresponding mapping*		

3.1 IDS Terminology Discussion

There is a strong connection but not a perfect match between the terms anomaly detection, behavior-based detection, and detection using dynamic knowledge/default deny. On the other end of the spectrum, the terms misuse detection, knowledge-based detection, signature detection, and detection using static knowledge/default permit are also very similar. The columns of Table 2 indicate the implied similarities between the terms. Let's have a closer look at each of these terms, to see whether we can merge them into a single taxonomy.

Debar et al. [1] define knowledge-based detection as an encoding of knowledge of attacks and vulnerabilities, which in our view corresponds well to the definition of misuse detection[2]. Taking the literal meaning of the term knowledge-based detection, it could also encompass an expert encoding normal behavior. With such a wider interpretation, we can then view Ko's specification-based detection [5] to be similar to knowledge-based detection. Furthermore, as the literal meaning of specification-based detection does not discern the source of the specification,

[2]We do not define misuse detection here, but the interested reader is welcome to follow one of the references, e.g. [5].

we can extend it to also include a specification of misuse. However, specification-based detection can also be viewed as anomaly detection per the definition of Halme and Bauer [4], because normal behavior is used as the basis for defining allowed behavior and thus how attacks are detected. For that reason we have listed specification-based detection in the middle column in Table 2, between anomaly and misuse detection.

Let's now consider the terms found in Lindqvist's classification [9]. In most cases, static knowledge is used in misuse (or knowledge-based) detection, while dynamic knowledge is more common for anomaly (behavior-based) systems. For example, normal behavior profiles can adapt to changes in behavior, while an encoded attack signature does not change once it has been inserted into the system. However, there are exceptions to these cases and we can only view such interpretations as rules of thumb. Now consider the orthogonal division into default deny and default permit. Default deny means that we specify what is acceptable behavior and generate alarms for anything that is outside this specification, while for default permit we list specific events we generate alerts for and consider all else normal. In our opinion these two classes correspond closely to anomaly and misuse detection.

Axelsson [2] defines signature detection as detection based on knowledge of a model of the intrusive process. Both legal and illegal behavior can be defined, thus also encompassing the specification-based approach. His signature-inspired detection class uses a mixed model of both intrusive and normal behavior. A more general name for this class would be *compound detectors*, but he did not use that term since the existing systems in this class focus on the intrusive model, and mainly use the normal model to avoid choosing features prone to false alarms.

MAFTIA [3] includes two binary attributes called *behavior-based* and *knowledge-based* in the *generic attributes* class, but these are not central in the detection component taxonomy. The taxonomy in MAFTIA is very interesting, but it is not easily mapped to previous efforts and has no obvious high level classes. Since we want to study the importance of commonly-used high-level IDS attributes for detection, we do not further discuss this work.

3.2 Consolidation of IDS Terminology

In the terminology discussion above, we showed that the taxonomy categories for detection method are quite similar (excluding the MAFTIA taxonomy) and can basically be summarized in two orthogonal concepts: the type of reference model used in the detection process and how this reference model is created. Let's have a closer look at these concepts.

- **Type of Reference Model** The reference model in the detection system, i.e. the "knowledge base" that input data is compared to, can define the normal user model, the attacker model, or both. Detection is basically about classifying observations as belonging to the normal model or the attack model.

  Now consider the terms summarized in Table 2. Misuse and anomaly detection refers to the underlying reference model. In misuse detection, the reference model is based on attack behavior, while anomaly detection use normal behavior as a reference. Furthermore, also the terms default permit and default deny belong here, as well as the categories signature, signature-inspired, and anomaly. Signature inspired models both attacker and normal user behavior. Knowledge-based and behavior-based as defined by Debar et al. [1] correspond to defining the attacker model versus defining the normal user model.

- **Creator of Reference Model** The other concept is about how the reference model is created, i.e. basically how much of the reference model that is knowledge encoded by experts and how much is learned from the data itself by an algorithm. This is a direct application of the concepts underlying the terms programmed versus self-learning.

The concept of static versus dynamic knowledge does not have a perfect match, and may form a third attribute. It handles the updating strategy of the reference model. However, it is rather close to the concept of programmed versus self-learning, which motivates our choice of only using the "creator of model" concept.

Axelsson [2] have used these two orthogonal concepts, but with different names. As mentioned, Lindqvist [9] used a similar scheme with two concepts. Furthermore, we believe that there are not two distinct categories for each concept, but a sliding scale. Axelsson's [2] third category, *signature-inspired detection*, can be seen as a simplified scheme of the sliding scale because he identified the need to have something between the two extremes previously discussed in literature.

3.3 Classification of Research IDSs

One of the main uses of previous taxonomies has been to survey intrusion detection systems. Focusing on the attributes *type of reference model* and the *creator of reference model*, we have classified several well-known intrusion detection systems and the result is shown in Figure 1. Completely self-learning systems are on one end of the scale as opposed to programmed systems that are found on the other. In between, there are systems using some prior (programmed) knowledge in combination with learning. The other axis shows whether the systems use training data

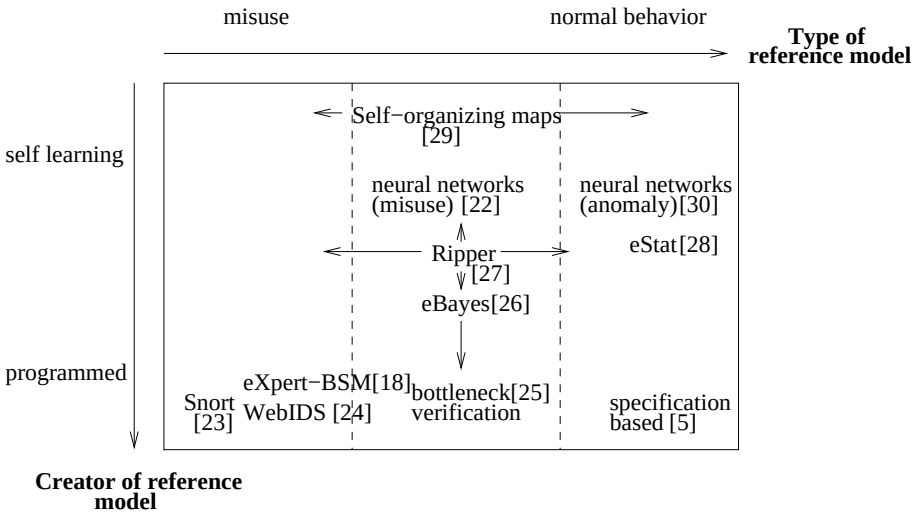

Figure 1: Detection method categorization

about attacks or misuse, or use training data based on normal behavior. Our classification of the IDSs is based on how we believe they are used, after studying the relevant research papers. The systems could be used differently with another knowledge base. The placement of each system is not exact, as we prioritized readability in the figure. Further descriptions of the systems can be found in Appendix A.

We find that the classified IDSs are well dispersed in Figure 1, making an empirical argument that the attributes *type of reference model* and the *creator of reference model* are relevant for survey classifications. In the next section, we consider the relevance of these attributes in a case study with attacks.

4 Case Study of Detection Method Suitability

In the previous sections we have looked at taxonomies relevant to intrusion detection and made an effort to consolidate the main categories into a single framework. Previous taxonomies have worked well in describing ongoing research prototypes, but they have not managed to convey why their chosen categories are better than other possible attributes. We have made a few observations here. Most taxonomies do include a division on the reference model, which seems to imply that it should be significant for classification. The other orthogonal division is not widely used, implying that it may have less relevance.

In this section, we are going to consider two examples of attacks and how well they can be detected by different intrusion detection systems classified as shown in Figure 1.

4.1 Preliminaries

We have chosen attacks and detection methods that we find are quite diverse. The methods and attacks are described in this section together with a motivation of why we chose these specific examples for the case study.

4.1.1 Detection Method Descriptions

Instead of discussing complete systems, we have chosen to concentrate on detection method paradigms. The three examples we use are chosen from different points in Figure 1 with respect to both attributes. In principle, these methods can be used with any reference model in mind, but in practice they differ in which model they use.

Stateless Pattern Matching One of the most common methods used to detect attacks is built on some kind of (stateless) pattern matching, and Kumar [8] describes some relevant aspects of such detection. Such detection is simple, and can be very efficient in terms of keeping up with high-speed event streams. Snort uses a form of stateless pattern matching for its signature rules, and we use it as an example in our discussion. We do recognize that Snort has many capabilities that we do not use for this study.

Expert Systems Forward-chaining rule-based systems have successfully been used for intrusions detection. They provide a powerful inference engine, and separate the knowledge base from the fact base. In our discussion, we are going to use rules from eXperts used in the EMERALD framework [17, 18].

Neural Networks Artificial neural networks are a powerful computational tool and maybe the most well-known artificial intelligence system capable of inductive learning. There has been several systems using neural networks for intrusion detection, but there is no research paper that describes experiences of a general system used for detection of a wide range of attacks. For that reason, the discussion of the suitability of neural network detection for the attacks is kept at a general level.

4.1.2 Sample Attack Descriptions

The attacks described here are chosen because we find them representative of the spectrum from the discrete to the continuous case, and successful detection should require the use of different reference models. The attacks are relatively old and should be well-known. The details are provided in the references.

echo/chargen attack [19] A DoS attack created by any UDP packet addressed to the destination port 7 (*echo*), and with the source port 19 (*chargen*), or the other way around.

syn flood [20] A DoS attack where the server receives *too many* SYN packets without accompanying ACKs, thus exhausting state in the server.

The *echo/chargen attack* is well-defined using only misuse information. It is what we call a discrete attack, because it consists of a single event and this event clearly distinguishes it from normal behavior. The *syn flood attack* is well-defined when it comes to the misuse model, but we need normal data to define *too many* in relation to the normal traffic at a specific site. This is an example of a continuous attack, consisting of several events and with no distinct decision boundary between normal and malicious behavior.

4.1.3 Analysis of the Attacks

Table 3 shows classifications of the attacks according to Lindqvist [9], Mell [11] and Kumar [8]. In Lindqvist's classification, the *echo/chargen* and the *syn flood* looks exactly the same. However, there is probably a difference in how they are best detected. Mell's classification shows differences for the attacks but does not give any more hints for detection than Lindqvist's classes. The difference is that Mell's classification provides more information, e.g. about what data to collect and which platforms that are vulnerable. Kumar's classification reveals that the detection of the *echo/chargen attack* may be simpler than the detection of the *syn flood attack*. None of these classifications seems to map to the detection method attributes that we are studying.

4.2 Detection Evaluation

For each of the detection methods we demonstrate how they can be used to detect the attacks, and discuss the suitability of the method for each attack.

4.2.1 Detection via Pattern Matching

Snort already comes equipped with some rules to detect the exploits described in the previous sections, as shown in Table 4. The rule used to detect the simplest of the attacks, the *echo/chargen attack*, is listed first. By matching on the right protocol and with specific port numbers, we have an efficient detection signature.

Finally, we have a signature in Snort to detect a *syn flood attack*. However, this signature is directed at a specific *syn flood attack* created by a Shaft agent where the sequence number is always the same. There is no simple way to write a pattern matching signature for a general *syn flood attack*. Either you can match on no one, or all instances of SYN packets. However, as SYN packets are normal to network traffic (and there is a significant number of them), we cannot match on every SYN packet the IDS sniffs. As a parenthesis, there are plugins to Snort that extends its detection capabilities to also include syn floods.

Summary of Packet Matching Detection Clearly, stateless pattern matching works well for the *echo/chargen attack*. It is efficient in that pattern matching can be done fast and with a small memory footprint. We do not expect to see any false positives or negatives from the signature either, based on the knowledge that no normal traffic should have the characteristics of the attack. The administrator does not need to tune the rule and the alerts are distinct with good information, such as the type of attack and the computers targeted. We lack the actual attacker, as the source address is spoofed but that can be captured at the network boundary.

Table 3: The attacks classified according to Lindqvist [9], Mell [11] and Kumar [8]

Lindqvist	**echo/chargen attack** *Intrusion technique:* Active misuse of resources → Resource exhaustion *Intrusion result:* Denial of service → Unselective → Affects all users **syn flood attack** *Intrusion technique:* Active misuse of resources → Resource exhaustion *Intrusion result:* Denial of service → Unselective → Affects all users
Mell	**echo/chargen attack** *Script goal:* DoS remote → Crash/freeze host *Target type:* Hosts → UNIX *Attacker platform:* All *Transmission method:* IP → UDP *Attacker requirements:* Network access *Target of attack:* Application/daemon **syn flood attack** *Script goal:* DoS remote → Crash/freeze application (or host) *Target type:* Hosts, Router, Firewall *Attacker platform:* All *Transmission method:* IP → TCP *Attacker requirements:* Network access *Target of attack:* Network protocol stack
Kumar	**echo/chargen attack** Existence pattern **syn flood attack** Sequence → Interval

Table 4: Snort signatures for the different attacks.

echo/chargen attack

```
alert udp any 19 <> any 7 (msg:"DOS UDP echo+chargen bomb"; reference:cve,CAN-1999-0635;
reference:cve,CVE-1999-0103; classtype:attempted-dos; sid:271; rev:3;)
```

syn flood attack

```
alert tcp $HOME_NET any <> $EXTERNAL_NET any (msg:"DDOS shaft synflood"; flags: S;
seq: 674711609; reference:arachnids,253; classtype:attempted-dos; sid:241; rev:2;)
```

When it comes to the *syn flood attack* we have reached the limits of simple pattern matching. There is an example signature in Snort for a specific kind of syn flood, but there is no general signature and we claim no such signature can be created without extending the pattern matching language.

4.2.2 Detection via Expert Systems

As the basis for our discussion of expert system detection we use the material found in [17], even though Lindqvist et al. do not list an explicit rule for the *echo/chargen attack*. The rule for detecting this attack is trivial to construct, and results in a single rule similarly to the Snort rule described above.

Contrary to the stateless pattern matching, we can create rules for detecting *syn flood attacks*, because the expert system keep state through the fact base. Such an example is shown in Table 5.

Summary of Expert System Detection With the expert system, we can design rules to detect both types of attacks. The *echo/chargen attack* is concisely described in a single rule, which should not create false positives or negatives.[3]

When it comes to the *syn flood attack*, the expert system excels over stateless pattern matching. By defining several constants that depend on the site in question, the expert system keeps track of ongoing connections and alerts when there are too many of them. However, the rules for the *syn flood attack* are relatively complex. The simple high-level description given in Section 4.2 requires seven codependent rules to successfully alert for the attack. The thresholds

[3] As this attack is also easily detected with stateless pattern matching, one might in practice choose to avoid describing such a rule within the expert system for performance reasons.

Table 5: Expert rules for detection of syn flood attacks (from [17])

```
1  rule[create_open_conn(*):
2    [+ev:conn_event| e_type == 0 ]
3  ==>
4    [+open_conn | seq_id=ev.seq_id,
5          sec = ev.sec,
6          expired = 0
7          client_ID = ev.client_ID]
8    [-|ev]
9  ]

10 rule[destroy_open_conn(*):
11   [+ev:conn_event| e_type == 1 ]
12   [+oc:open_conn | seq_id = (ev.seq_id - 1) ]
13 ==>
14   [-|oc]
15   [-|ev]
16 ]

17 rule|ignore_spurious_acks(*):
18   [+ev:conn_event| e_type == 1 ]
19   [+oc:open_conn | seq_id = (ev.seq_id - 1) ]
20 ==>
21   [-|ev]
22 ]

23 rule[first_bad_conn(*):
24   [+ts:time]
25   [-bad_conn]
26   [+oc:open_conn|expired == 0]
27   [?|(ts.sec - oc.sec) > 'expire_time]
28 ==>
29   [+bad_conn|count = 1]
30   [/oc| expired = 1]
31 ]
```

```
rule[add_to_bad_cons(*):
  [+ts:time]
  [+oc:open_conn| expired == 0 ]
  [?|(ts.sec - oc.sec) > 'expire_time]
  [+bc:bad_conn|count < 'max_bad_conns]
==>
  [/bc| count += 1]
  [/oc| expired = 1]
]

rule[max_open_cons_cons(*):
  [+ts:time]
  [+oc:open_conn| expired == 0 ]
  [?|(ts.sec - oc.sec) > 'expire_time]
  [+bc:bad_conn|count == 'max_bad_conns]
==>
  [?|syn_alert("SYN Attack:  Last Host %s.
       SeqID = %d.  Time = %d:,
        oc.client_ID, oc.seq_id, oc.sec)]
  [/bc|count = 1]
  [/oc|expired = 1]
]

rule[free_bad_open_cons(*):
  [+ts:time]
  [+bc:bad_conn]
  [+oc:open_conn|expired == 1]
  [?|(ts.sec - oc.sec) > 'bad_conn_life]
==>
  [-|oc]
  [/bc|count -= 1]
]
```

must be manually adapted *per installed site*.[4] The system administrator must not only be competent enough to be able to specify the traffic profile at his site but also learn enough of the expert system to correctly set the necessary variables. The addition of new detection rules might be beyond the reach of a user of the IDS.

4.2.3 Detection via Neural Networks

Neural networks use inductive learning, instead of specified rules, and are especially suited for tackling certain types of problems with the following characteristics, as listed by Mitchell [21].

- Instances have many attribute-value pairs.
- Training data may contain errors.
- A longer training time is acceptable, while the evaluation time is required to be fast.
- There is no need for transparency of learning, i.e. a human does not need to understand *what* the network has learned.

The key term when discussing neural networks is learning. Considering the *echo/chargen attack*, we would not design a network to detect that specific attack instance. Instead, we would look at the characteristics of that attack, i.e. the attack uses an anomalous combination of port numbers, and train a network with normal port combinations and possibly labeled instances of the *echo/chargen attack*. As it turns out, some other ports (*daytime* on 13 and *time* on 37) may also form a DoS attack. Our neural network might flag these combinations as anomalous.

The *syn flood attack*, seems to be a good match for a neural network. It is a continuous attack with a range of acceptable behavior that changes per site. By using the network's ability to manually adapt to local traffic, there is no need for the administrator to tune the detection rule. Cannady [22] claims a good detection of *syn floods* with a forward-feed neural network. A recurrent network providing short-term memory should also work well for such detection.

Summary of Detection via Neural Networks Self-learning systems offer an ability to adapt to the local site, or learn to correctly classify attacks. The key for successful self-learning is an abundance of training data. The

[4] In some rare cases, we expect it to be possible to define such thresholds once and then be applicable for most sites.

problem in intrusion detection is that both misuse and normal data is usually difficult to come by, and finding labeled data is even more difficult.

We may use a neural network for detection of anomalous port activity. Such a network might be able to alert for the *echo/chargen attack*, as well as the attack variants with other port combinations. Both the pattern matching system and the expert system miss the "unknown" variants of this attack. However, it is difficult to evaluate a more complex neural network and we expect there are both false negatives and false positives. Furthermore, as the neural network works like a black box, the system administrator might find it difficult to interpret the alerts and then track down the false positives.

4.3 Discussion of Attack – Detection Method Connection

For all detection methods described above, there are clear tradeoffs in terms of memory footprint, efficiency, and ease-of-use for the operator. However, in this case study we concentrate on evaluating how well detection methods classified with the two concepts *type of reference model* and *creator of reference model* can detect the two attack examples.

To reiterate, the *echo/chargen attack* consists of a single event, and any such event indicates malicious behavior. The *syn flood attack* consists of several integrated events. SYN packets are part of the normal traffic and even many SYN packets may not be an indication of an attack but just a peak of normal traffic.

Type of Reference Model Let's first consider the implications of the *type of reference model*. The *echo/chargen attack* is easily defined through a misuse model. In practice, both pattern matching systems and expert systems use misuse models (as seen in Figure 2). Both of these detection methods detect this specific attack. Neural networks, on the other hand, usually emphasizes the normal behavior model and this is reflected in our redefinition of detection of the *echo/chargen attack* given in Section 4.2.3. We do not detect the specific attack, but we redesigned the detection problem to detect any anomalous port combinations (including the *echo/chargen attack*).

Successful detection of the *syn flood attack* not only requires an attack model but also a normal user model to define a threshold (or a probability) that we are under attack. The stateless pattern matching technique mainly failed because it could not keep any state, thwarting its usefulness to this discussion. The expert system, with its powerful inference, manage to model the attack but its normal behavior model in the form of a set of user-defined constants seems limiting and demanding for correct tuning. The neural network automatically creates a model of normal behavior based on seen examples (collected at the local site). However, here we have problems in modeling the attack in sufficient detail, as we do not expect there to be many malicious instances available for training.

Different attacks require different reference models, in turn implying suitable detection methods. However, there are probably attacks that are difficult to describe with a certain reference model, even though our two examples do not include such an example. Such attacks would not be *well-formed* when it comes to the reference model axis.

Creator of Reference Model Let's now consider the attribute of the *creator of the reference model*. The first reflection is that self-learning systems need an abundance of data to learn, but data is scarce in intrusion detection. Normal data (where one simply assumes there are no attacks) can be collected. Misuse data is scarce, and we might have to generate synthetic data. The existence of data does not play any intrinsic role for the attacks, but only reflects a facet of reality.

For some attacks, we need to tune the detection signature with normal traffic. Again, having an human expert or a self-learning system do this tuning does not seem to make any difference when it comes to the end result. For practical reasons, a self-learning system might be preferable as it can automatically be tuned on the spot but that aspect only reflects that manual labor is expensive. As a parenthesis, the Lindqvist's dynamic versus static division seems more applicable here.

It seems like the attribute of the *creator of the reference model* does not play a significant role when trying to find an efficient detection method for a specific attack. There are reasons one would use an expert to encode the necessary knowledge (scarcity of data), or why one would use a self-learning algorithm (automatic continuous tuning possible). Neither of these reasons is a reflection of the underlying attack. If we look at Figure 2, we see that no self-learning system is located in the top left corner, and only one programmed system is located in the bottom right corner. This reflects the fact that there is little misuse data available, and normal behavior is quite diverse and difficult to specify formally in rules. For that reason, one could even argue that the attribute *creator of the reference model* is partly defined by the available data sources and maybe should be replaced by a better, more independent attribute.

5 Discussion and Future Work

In this paper, we have surveyed previous classifications done for intrusion detection. After having summarized the different terms used, we consolidated the key attributes relevant for detection methods into two orthogonal concepts: the *type of reference model* used by the system and the *creator of the reference model*. Not only does these two terms provide a uniform framework for defining the previously used terms, but these concepts also reflect the development of the field. For example, many systems today use mixed reference models, and such systems could not well be defined with the normal binary detection method attribute of misuse or anomaly detection.

To motivate these attributes, we used the taxonomy in two different ways. First we classified 11 research prototypes, resulting in most IDSs well dispersed in the defined space, thus making an empirical argument that the attributes *type of reference model* and the *creator of reference model* are relevant for surveys.

Second we evaluated the relevance of three classified detection methods on attack detection in a case study. We found that the first concept, *type of reference model*, is quite relevant. However, *creator of the reference model* does not appear as relevant for this particular use of the classification. Further study revealed that in practice the *creator of the reference model* is partly dependent on the *type of reference model* due to data scarcity. This makes us question its use as a key attribute, and we are currently revising the case study to incorporating the slightly different concept of static versus dynamic knowledge.

Even though the *type of reference model* used is relevant, it does not seem to reflect a choice for the system designer but an inherent consequence based on the specific attack you want to detect.

We have only used two dimensions in this study, and the work should of course be extended to also incorporating other important aspects, such as stateful versus stateless detection.

References

[1] Hervé Debar, Marc Dacier, and Andreas Wespi. Towards a taxonomy of intrusion-detection systems. *Computer Networks*, 31(8):805–822, April 1999.

[2] Stefan Axelsson. Intrusion detection systems: A taxomomy and survey. Technical Report No 00-4, Dept. of Computer Engineering, Chalmers Univerity of Technology, Sweden, March 2000.

[3] D. Alessandri, C. Cachin, M. Dacier, O. Deak, K. Julisch, B. Randell, J. Riordan, A. Tscharner, A. Wespi, and C. Wüest. Towards a taxonomy of intrusion detection systems and attacks. Technical Report RZ 3366, IBM Research, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland, 2001. MAFTIA project, report D3.

[4] Lawrence R. Halme and R. Kenneth Bauer. AINT misbehaving - a taxonomy of anti-intrusion techniques. In *Proceedings of the 18th National Information Systems Security Conference*, pages 163–172, Baltimore, MD, USA, October 1995. National Institute of Standards and Technology/National Computer Security Center.

[5] C. Ko, M. Ruschitzka, and K. Levitt. Execution monitoring of security-critical programs in distributed systems: A specification-based approach. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, Oakland, California, 1997.

[6] Kathleen A. Jackson. Intrusion detection system (IDS) product survey. Technical Report LA-UR-99-3883, Los Alamos National Lab, 1999.

[7] Håkan Kvarnström. A survey of commercial tools for intrusion detection. Technical Report TR99-8, Chalmers University of Technology, 1999.

[8] Sandeep Kumar. *Classification and detection of computer intrusions*. PhD thesis, Purdue University, West Lafayette, Indiana, August 1995.

[9] Ulf Lindqvist. *On the fundamentals of analysis and detection of computer misuse*. PhD thesis, Chalmers University of Technology, Göteborg, Sweden, 1999.

[10] Peter G. Neumann and Donn Parker. A summary of computer misuse techniques. In *Proceedings of the 12th National Computer Security Conference (NCSC*, pages 396–407, Baltimore MD, 10-13 October 1989.

[11] Peter Mell. Understanding the global attack toolkit: Using a database of dependent classifiers. In *2nd Workshop on Research with Security Vulnerability Databases*, January 21-22 1998.

[12] Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba, and Kumar Das. The 1999 darpa off-line intrusion detection evaluation. *Computer Networks*, Volume 34(Issue 4):579–595, October 2000. Elsevier Science B.V.

[13] Dorothy E Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, SE-13(2):222–232, February 1987.

[14] Carl E Landwehr, Alan R Bull, John P McDermott, and William S Choi. A taxonomy of computer program security flaws. *ACM Computing Surveys*, 26(3):211–254, September 1994.

[15] Ivan V Krsul. *Software vulnerability analysis*. PhD thesis, Purdue University, West Lafayette, Indiana, May 1998.

[16] John D Howard. *An analysis of security incidents on the Internet 1989-1995*. PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania, April 1997.

[17] Ulf Lindqvist and Phillip A Porras. Detecting computer and network misuse through the production-based expert system toolset (P-BEST). In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, pages 146–161, Oakland, California, May 9–12, 1999.

[18] Ulf Lindqvist and Phillip A Porras. eXpert-BSM: A host-based intrusion detection solution for Sun Solaris. In *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC 2001)*, pages 240–251, New Orleans, Louisiana, December 10–14, 2001.

[19] CERT. Cert advisory ca-1996-01 udp port denial-of-service attack. `http://www.cert.org/advisories/CA-1996-01.html`, February 1996.

[20] Christoph L Schuba, Ivan V Krsul, Markus G Kuhn, Eugene H Spafford, Aurobindo Sundaram, and Diego Zamboni. Analysis of a denial of service attack on TCP. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 208–223, Oakland, California, May 4–7, 1997.

[21] Tom M Mitchell. *Machine Learning*. McGraw, 1997. ISBN 0-07-042807-7.

[22] James Cannady. Artificial neural networks for misuse detection. In *Proceedings of the 1998 National Information Systems Security Conference (NISSC'98)*, pages 443–456, Arlington, VA, October 5-8 1998.

[23] Martin Roesch. SNORT - lightweight intrusion detection for networks. In *Proceedings of the 13th Systems Administration Conference - LISA '99*, Seattle, Washington, USA, November 7-12 1999. USENIX.

[24] M. Almgren, H. Debar, and M. Dacier. Lightweight tool for detecting web server attacks. In *Proceedings of the Network and Distributed System Security Symposium*, 2000.

[25] Robert K Cunningham, Richard P Lippman, and Seth E Webster. Detecting and displaying novel computer attacks with Macroscope. In *Proceedings of the 2000 IEEE Workshop on Information Assurance and Security*, United States Military Academy, West Point, NY, June 6–7 2000.

[26] Alfonso Valdes and Keith Skinner. Adaptive, model-based monitoring for cyber attack detection. In H. Debar, L. Me, and F. Wu, editors, *From Recent Advances in Intrusion Detection (RAID 2000)*, number 1907 in Lecture Notes in Computer Science, pages 80–92, Toulouse, France, October 2000. Springer-Verlag.

[27] Wenke Lee, Sal Stolfo, and Kui Mok. A data mining framework for building intrusion detection models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, Oakland, CA, May 1999.

[28] H.S. Javitz and A. Valdes. The NIDES statistical component description and justification. Technical report, Computer Science Laboratory, SRI International, Menlo Park, CA, March 1994.

[29] Kevin L. Fox, Ronda R. Henning, Jonathan H. Reed, and Richard P. Simonian. A neural network approach towards intrusion detection. In *Proceedings of the 13th National Computer Security Conference*, pages 125–133, Washington D.C., October 1990. National Institute of Standards and Technology/National Computer Security Center.

[30] Hervé Debar, Monique Becker, and Didier Siboni. A neural network component for an intrusion detection system. In *Proceedings of the IEEE Symposium on Research in Computer Security and Privacy*, 1992.

Appendix A

Here we have collected the descriptions of the classified intrusion detection systems. The descriptions refer to Figure 1 found in the main text.

Snort In the lower left corner of the map are systems like *Snort* [23]. These use rather simple string matching to detect known attacks and all knowledge is pre-programmed. This is the most common approach in commercial detection systems today.

WebIDS *WebIDS* [24] is found to the right of Snort. It is a pattern matching system but also uses thresholds and it is a little bit more complex than Snort. Setting the thresholds requires expert knowledge about the difference between normal behavior and attack behavior.

eXpert-BSM It is possible to make more general rules that may catch groups of intrusions instead of specific instances. This is done in *eXpert-BSM* [18], where an expert system[5] is used, which can handle more complex compound rules with several ordered or unordered events and negations. This is a pure programmed system, based on misuse knowledge, but it implicitly encodes knowledge about normal behavior in the general rules. This system is placed close to WebIDS in the map.

Bottleneck verification Even further to the right, is *Bottleneck verification* [25]. This system models the valid privilege transition paths and parses user commands to find out if high privilege operations are performed without going through the "bottleneck", i.e. login or the *su* command in a UNIX system, to get higher privilege. This means that they use programmed models of both normal and attack behavior.

Specification-based In the lower right corner of the map is the *specification-based* system by Ko et al. [5]. They manually specify valid sequences of execution events for important programs and detects deviations from these sequences. This is a pure programmed system using only normal behavior data as reference for detection.

eBayes In the middle of the map we find *eBayes* [26], which monitors TCP sessions. It uses Bayes nets to model different forms of normal and attack behavior. At least theoretically, these models can be either specified, learned or hybrid. The Bayes nets rely on conditional probability tables that can be learned either off-line or adaptively.

Ripper Above eBayes, also in the middle is *Ripper* [27]. Ripper needs labeled data containing both normal data and attacks. From a large number of features Ripper selects the most discriminating and generalizable features. It can be used to create both misuse and anomaly detection models, even though the best results are achieved for misuse models. The requirements of this approach is rather close to the neural network model. The advantages of Ripper is that it gives information about which features it uses and creates more readable "rules". However, the readability of the rules probably depend on the level of detail in the labeling of training data and should be better for misuse models than anomaly models. Higher level of detail in the labeling requires more knowledge, and thus brings Ripper more towards the programmed systems. If labels are only *attack* or *normal*, it is very close to the misuse neural network in [22].

eStat To the right of Ripper, we find *eStat*, which is a pure anomaly detection system based on NIDES statistical component [28]. It is close to (but below) the anomaly-based neural network, and it has been suggested that these kinds of systems can be interchanged. However, the statistics-based systems requires slightly more encoded knowledge and programming, since suitable statistical measures must be defined and adjusted, which is done automatically in the neural network.

Neural networks In the upper part of the map, we find several types of *neural networks*. They can implement unsupervised learning using Kohonens *self-organizing maps* as in [29], or they can use supervised learning with labeled data as in [22]. The unsupervised learning strategy is completely self-learning, except for some initial choice of parameters to use. The supervised learning strategy is a little bit more towards the programmed systems, since it needs chosen training examples to do its job well. The supervised systems may be pure *anomaly* detection systems (e.g. [30]) using only normal behavior data for training, or *"misuse"* detection systems (e.g. [22]), which use a mix of (labeled) normal and attack data. Misuse detection using neural networks is not very common, however, since it is difficult to obtain data labeled data with a high percentage of attacks. Cannady [22] uses about 30% simulated attacks mixed into normal data.

[5] More specifically, it is a forward-chaining production system.