Intrusion Detection Systems: A Survey and Taxonomy

Stefan Axelsson
Department of Computer Engineering
Chalmers University of Technology
Göteborg, Sweden
email: *sax@ce.chalmers.se*

14 March 2000

Abstract

This paper presents a taxonomy of intrusion detection systems that is then used to survey and classify a number of research prototypes. The taxonomy consists of a classification first of the detection principle, and second of certain operational aspects of the intrusion detection system as such. The systems are also grouped according to the increasing difficulty of the problem they attempt to address. These classifications are used predictively, pointing towards a number of areas of future research in the field of intrusion detection.

1 Introduction

There is currently a need for an up-to-date, thorough taxonomy and survey of the field of intrusion detection. This paper presents such a taxonomy, together with a survey of the important research intrusion detection systems to date and a classification of these systems according to the taxonomy. It should be noted that the main focus of this survey is intrusion detection systems, in other words major research efforts that have resulted in prototypes that can be studied both quantitatively and qualitatively.

A taxonomy serves several purposes [FC93]:

Description It helps us to describe the world around us, and provides us with a tool with which to order the complex phenomena that surround us into more manageable units.

Prediction By classifying a number of objects according to our taxonomy and then observing the 'holes' where objects may be missing, we can exploit the predictive qualities of a good taxonomy. In the ideal case, the classification points us in the right direction when undertaking further studies.

Explanation A good taxonomy will provide us with clues about how to explain observed phenomena.

We aim to develop a taxonomy that will provide at least some results in all areas outlined above. With one exception [DDW99], previous surveys have not been strong when it comes to a more systematic, taxonomic approach. Surveys such as [Lun88, MDL+90, HDL+90, ESP95] are instead somewhat superficial and dated by today's standards. The one previous attempt at a taxonomy [DDW99] falls short in some respects, most notably in the discussion of detection principles, where it lacks the necessary depth.

2 Introduction to intrusion detection

Intrusion detection systems are the 'burglar alarms' (or rather 'intrusion alarms') of the computer security field. The aim is to defend a system by using a combination of an alarm that sounds whenever the site's security has been compromised, and an entity—most often a site security officer (SSO)—that can respond to the alarm and take the appropriate action, for instance by ousting the intruder, calling on the proper external authorities, and so on. This method should be contrasted with those that aim to strengthen the perimeter surrounding the computer system. We believe that both of these methods should be used, along with others, to increase the chances of mounting a successful defence, relying on the age-old principle of defence in depth.

It should be noted that the intrusion can be one of a number of different types. For example, a user might steal a password and hence the means by which to prove his identity to the computer. We call such a user a *masquerader*, and the detection of such intruders is an important problem for the field. Other important classes of intruders are people who are legitimate users of the system but who abuse their privileges, and people who use pre-packed exploit scripts, often found on the Internet, to attack the system through a network. This is by no means an exhaustive list, and the classification of threats to computer installations is an active area of research.

Early in the research into such systems two major principles known as *anomaly detection* and *signature detection* were arrived at, the former relying on flagging all behaviour that is abnormal for an entity, the latter flagging behaviour that is close to some previously defined pattern signature of a known intrusion. The problems with the first approach rest in the fact that it does not necessarily detect *undesirable* behaviour, and that the false alarm rates can be high. The problems with the latter approach include its reliance on a well defined security policy, which may be absent, and its inability to detect intrusions that have not yet been made known to the intrusion detection system. It should be noted that to try to bring more stringency to these terms, we use them in a slightly different fashion than previous researchers in the field.

An intrusion detection system consists of an audit data collection agent that collects information about the system being observed. This data is then either stored or processed directly by the detector proper, the output of which is presented to the SSO, who then can take further action, normally beginning with further investigation into the causes of the alarm.

3 Previous work

Most, if not all, would agree that the central part of an intrusion detection system is the detector proper and its underlying principle of operation. Hence the taxonomies in this paper are divided into two groups: the first deals with detection principles, and the second deals with system characteristics (the other phenomena that are necessary to form a complete intrusion detection system).

In order to gain a better footing when discussing the field, it is illustrative to review the existing research into the different areas of intrusion detection starting at the source, the intrusion itself, and ending with the ultimate result, the decision.

Obviously, the source of our troubles is an action or activity that is generated by the intruder. This action can be one of a bewildering range, and it seems natural to start our research into how to construct a detector by first studying the nature of the signal that we wish to detect. This points to the importance of constructing taxonomies of computer security intrusions, but, perhaps surprisingly, the field is not over-laden with literature. Such classifications of computer security violations that exist [LJ97, NP89] are not directed towards intrusion detection, and on closer study they appear to be formulated at too high a level of representation to be applicable to the problem in hand. We know of only one study that connects the classification of different computer security violations to the problem of detection, in this case the problem of what traces are necessary to detect intrusion after the fact [ALGJ98].

From the nature of the source we move to the question of how how to observe this source, and what problems we are likely to have in doing so. In a security context, we would probably perform some sort of security audit, resulting in a security audit log. Sources of frustration when undertaking logging include the fact that we may not be able to observe our subject directly in isolation; background traffic will also be present in our log, and this will most likely come from benign usage of the system. In other words, we would have an amount of traffic that is to varying degrees similar to the subject we wish to observe. However, we have found no study that goes into detail on the subject of what normal traffic one might expect under what circumstances. Although one paper states that in general it is probably not possible [HL93], we are not as pessimistic. With a sufficiently narrow assumption of operational parameters for the system, we believe useful results can be achieved.

This brings us to the results of the security logging—in other words what can we observe—and what we suspect we should observe given an idea of the nature of the security violation, background behaviour, and observation mechanism. One issue, for example, is be precisely what data to commit to our security log. Again the literature is scarce, although for instance [ALGJ98, HL93, LB98] address some of the issues, albeit from different angles.

How then to formulate the rule that governs our intrusion detection decision? Perhaps unsurprisingly given the state of research into the previous issues, this also has not been thoroughly addressed. More often than not we have to reverse engineer the decision rule from the way in which the detector is designed, and often it is the mechanism that is used to implement the detector rather than the detection principle itself. Here we find the main body of research: there are plenty of suggestions and implementations of intrusion detectors in the literature, [AFV95, HDL+90, HCMM92, WP96] to name but a few. Several of these detectors employ several, distinct decision rules, however, and as we will see later, it is thus often impossible to place the different research prototypes into a single category. It becomes more difficult to classify the detection principle as such because, as previously noted, it is often implicit in the work cited. Thus we have to do the best we can to classify as precisely as possible given the principle of operation of the detector. Some work is clearer in this respect, for example [LP99].

In the light of this, the main motivation for taking an in-depth approach to the different kinds of detectors that have been employed is that it is natural to assume that different intrusion detection principles will behave differently under different circumstances. A detailed look at such intrusion detection principles is thus in order, giving us a base for the study of how the operational effectiveness is affected by the various factors. These factors are the *intrusion detector*, the *intrusion* we wish to detect, and the *environment* in which we wish to detect it.

Such a distinction has not been made before, as often the different intrusion detection systems are lumped together according to the principle underlying the mechanism used to implement the detector. We read of detectors based on an *expert system*, an *artificial neural network*, or—least convincingly—a *data mining approach*.

4 A taxonomy of intrusion detection principles

The taxonomy described below is intended to form a hierarchy shown in table 1. A general problem is that most references do not describe explicitly the decision rules employed, but rather the framework in which such rules could be set. This makes it well nigh impossible to classify down to the level of detail that we would like to achieve. Thus we often have to stop our categorisation when we reach the level of the framework, for example the *expert system*, and conclude that while the platform employed in the indicated role would probably have a well defined impact on the operational characteristics of the intrusion detection principle, we cannot at present categorise that impact. Due both to this and the fact that the field as a whole is still rapidly expanding, the present taxonomy should of course be seen as a first attempt.

Since there is no established terminology, we are faced with the problem of finding satisfactory terms for the different classes. Wherever possible we have tried to find new terms for the phenomena we are trying to describe. However, it has not been possible to avoid using terms already in the field that often have slightly differing connotations or lack clear definitions altogether. For this reason we give a definition for all the terms used below, and we wish to apologise in advance for any confusion that may arise should the reader already have a definition in mind for a term used.

Table 1: Classification of detection principles

anomaly	self-learning	non time series	rule modelling	W&S A.4[a]
			descriptive statistics	IDES A.3, NIDES A.14, EMERALD A.19, JiNao A.18, Haystack A.1
		time series	ANN	Hyperview(1)[b] A.8
	programmed	descriptive stat	simple stat	MIDAS(1) A.2, NADIR(1) A.7, Haystack(1)
			simple rule-based	NSM A.6
			threshold	ComputerWatch A.5
		default deny	state series modelling	DPEM A.12, JANUS A.17, Bro A.20
signature	programmed	state-modelling	state-transition	USTAT A.11
			petri-net	IDIOT A.13
		expert-system	NIDES A.14, EMERALD A.19, MIDAS-direct A.2, DIDS A.9, MIDAS(2) A.2	
		string-matching	NSM A.6	
		simple rule-based	NADIR A.7, NADIR(2) A.7, ASAX A.10, Bro A.20, JiNao A.18, Haystack(2) A.1	
signature inspired	self-learning	automatic feature sel	Ripper A.21	

[a] Letter and number provide reference to section where it is described [b] Number in brackets indicates level of two tiered detectors.

4.1 Anomaly detection

Anomaly In anomaly detection we watch not for known intrusion—the signal—but rather for abnormalities in the traffic in question; we take the attitude that something that is abnormal is probably suspicious. The construction of such a detector starts by forming an opinion on what constitutes *normal* for the observed subject (which can be a computer system, a particular user etc.), and then deciding on what percentage of the activity to flag as *abnormal*, and how to make this particular decision. This detection principle thus flags behaviour that is unlikely to originate from the normal process, without regard to actual intrusion scenarios.

4.1.1 Self-learning systems

Self-learning Self-learning systems learn by example what constitutes *normal* for the installation; typically by observing traffic for an extended period of time and building some model of the underlying process.

Non-time series A collective term for detectors that model the normal behaviour of the system by the use of a stochastic model that does not take time series behaviour into account.

Rule modelling The system itself studies the traffic and formulates a number of rules that describe the normal operation of the system. In the detection stage, the system applies the rules and raises the alarm if the observed traffic forms a poor match (in a weighted sense) with the rule base.

Descriptive statistics A system that collects simple, descriptive, mono-modal statistics from certain system parameters into a profile, and constructs a distance vector for the observed traffic and the profile. If the distance is great enough the system raises the alarm.

Time series This model is of a more complex nature, taking time series behaviour into account. Examples include such techniques such as a hidden Markov model (HMM), an artificial neural network (ANN), and other more or less exotic modelling techniques.

ANN An *artificial neural network* (ANN) is an example of a 'black box' modelling approach. The system's normal traffic is fed to an ANN, which subsequently 'learns' the pattern of normal traffic. The output of the ANN is then applied to new traffic and is used to form the intrusion detection decision. In the case of the surveyed system this output was not deemed of sufficient quality to be used to form the output directly, but rather was fed to a second level expert system stage that took the final decision.

4.1.2 Programmed

Programmed The *programmed* class requires someone, be it a user or other functionary, who teaches the system—programs it—to detect certain anomalous events. Thus the user of the system forms an opinion on what is considered abnormal enough for the system to signal a security violation.

Descriptive statistics These systems build a profile of normal statistical behaviour by the parameters of the system by collecting descriptive statistics on a number of parameters. Such parameters can be the number of unsuccessful logins, the number of network connections, the number of commands with error returns, etc.

Simple statistics In all cases in this class the collected statistics were used by higher level components to make a more abstract intrusion detection decision.

Simple rule-based Here the user provides the system with simple but still compound rules to apply to the collected statistics.

Threshold This is arguably the simplest example of the *programmed—descriptive statistics* detector. When the system has collected the necessary statistics, the user can program predefined thresholds (perhaps in the form of simple ranges) that define whether to raise the alarm or not. An example is '(Alarm if) number of unsuccessful login attempts > 3.'

Default deny The idea is to state explicitly the circumstances under which the observed system operates in a security-benign manner, and to flag all deviations from this operation as intrusive. This has clear correspondence with a default deny security policy, formulating, as does the general legal system, that which is permitted and labelling all else illegal. A formulation that while being far from common, is at least not unheard of.

State series modelling In state series modelling, the policy for security benign operation is encoded as a set of states. The transitions between the states are implicit in the model, not explicit as when we code a state machine in an expert system shell. As in any state machine, once it has matched one state, the intrusion detection system engine waits for the next transition to occur. If the monitored action is described as allowed the system continues, while if the transition would take the system to another state, any (implied) state that is not explicitly mentioned will cause the system to sound the alarm. The monitored actions that can trigger transitions are usually security relevant actions such as file accesses (reads and writes), the opening of 'secure' communications ports, etc.
The rule matching engine is simpler and not as powerful as a full expert system. There is no unification, for example. It does allow fuzzy matching, however—fuzzy in the sense that an attribute such as 'Write access to any file in the `/tmp` directory' could trigger a transition. Otherwise the actual specification of the security-benign operation of the program could probably not be performed realistically.

4.2 Signature detection

Signature In signature detection the intrusion detection decision is formed on the basis of knowledge of a model of the intrusive process and what traces it ought to leave in the observed system. We can define in any and all instances what constitutes *legal* or *illegal* behaviour, and compare the observed behaviour accordingly.

It should be noted that these detectors try to detect evidence of intrusive activity irrespective of any idea of what the background traffic, i.e. normal behaviour, of the system looks like. These detectors have to be able to operate no matter what constitutes the normal behaviour of the system, looking instead for patterns or clues that are thought by the designers to stand out against the possible background traffic. This places very strict demands on the model of the nature of the intrusion. No sloppiness can be afforded here if the resulting detector is to have an acceptable detection and false alarm rate.

Programmed The system is programmed with an explicit decision rule, where the programmer has himself prefiltered away the influence of the channel on the observation space. The detection rule is simple in the sense that it contains a straightforward coding of what can be expected to be observed in the event of an intrusion.

Thus, the idea is to state explicitly what traces of the intrusion can be thought to occur uniquely in the observation space. This has clear correspondence with a default permit security policy, or the formulation that is common in law, i.e. listing *illegal* behaviour and thereby defining all that is not explicitly listed as being permitted.

State-modelling State-modelling encodes the intrusion as a number of different states, each of which has to be present in the observation space for the intrusion to be considered to have taken place. They are by their nature time series models. Two subclasses exist: in the first, *state transition*, the states that make up the intrusion form a simple chain that has to be traversed from beginning to end; in the second, *petri-net*, the states form a petri-net. In this case they can have a more general tree structure, in which several preparatory states can be fulfilled in any order, irrespective of where in the model they occur.

Expert-system An expert system is employed to reason about the security state of the system, given rules that describe intrusive behaviour. Often forward-chaining, production-based tool are used, since these are most appropriate when dealing with systems where new facts (audit events) are constantly entered into the system. These expert systems are often of considerable power and flexibility, allowing the user access to powerful mechanisms such as unification. This often comes at a cost to execution speed when compared with simpler methods.

String matching String matching is a simple, often case sensitive, substring matching of the characters in text that is transmitted between systems, or that otherwise arise from the use of the system. Such a method is of course not in the least flexible, but it has the virtue of being simple to understand. Many efficient algorithms exist for the search for substrings in a longer (audit event) string.

Simple rule-based These systems are similar to the more powerful expert system, but not as advanced. This often leads to speedier execution.

4.3 Compound detectors

Signature inspired These detectors form a compound decision in view of a model of both the normal behaviour of the system and the intrusive behaviour of the intruder. The detector operates by detecting the intrusion against the background of the normal traffic in the system. At present, we call these detectors 'signature inspired' because the intrusive model is much stronger and more explicit than the normal model. These detectors have—at least in theory—a much better chance of correctly detecting truly interesting events in the supervised system, since they both know the patterns of intrusive behaviour and can relate them to the normal behaviour of the system. These detectors would at the very least be able to qualify their decisions better, i.e. give us an improved indication of the quality of the alarm. Thus these systems are in some senses the most 'advanced' detectors surveyed.

Self learning These systems automatically learn what constitutes intrusive and normal behaviour for a system by being presented with examples of normal behaviour interspersed with intrusive behaviour. The examples of intrusive behaviour must thus be flagged as such by some outside authority for the system to be able to distinguish the two.

Automatic feature selection There is only one example of such a system in this classification, and it operates by automatically determining what observable features are interesting when forming the intrusion detection decision, isolating them, and using them to form the intrusion detection decision later.

4.4 Discussion of classification

While some systems in table 1 appear in more than one category, this is not because the classification is ambiguous but because the systems employ several different principles of detection. Some systems use a two-tiered model of detection, where one lower level feeds a higher level. These systems (MIDAS, NADIR, Haystack) are all of the type that make *signature—programmed—default permit* decisions on anomaly data. One could of course conceive of another type of detector that detects anomalies from signature data (or alarms in this case) and indeed one such system has been presented in [MCZH99], but unfortunately the details of this particular system are so sketchy as to preclude further classification here.

It is probable that the detection thresholds of these systems, at least in the lower tier, can be lowered (the systems made more sensitive) because any 'false alarms' at this level can be mitigated at the higher level.

It should be noted that some of the more general mechanisms surveyed here could be used to implement several different types of detectors, as is witnessed by some multiple entries. However, most of the more recent papers do not go into sufficient detail to enable us draw more precise conclusions. Examples of the systems that are better described in this respect are NADIR, MIDAS, and the P-BEST component of EMERALD.

4.4.1 Orthogonal concepts

From study of the taxonomy, a number of orthogonal concepts become clear: *anomaly/signature* on the one hand, and *self-learning/programmed* on the other. The lack of detectors in the *signature—self-learning* class is conspicuous, particularly since detectors in this class would probably prove useful, combining as they do the advantages of self-learning systems—they do not have to perform the arduous and difficult task of specifying intrusion signatures—with the detection efficiency of signature based systems.

4.4.2 High level categories

We see that the systems classified fall into three clear categories depending on the type of intrusion they detect most readily. In order of increasing difficulty they are:

Well known intrusions Intrusions that are well known, and for which a 'static', well defined pattern can be found. Such intrusions are often simple to execute, and have very little inherent variability. In order to exploit their specific flaw they must be executed in a straightforward, predictable manner.

Generalisable intrusions These intrusions are similar to the *well known intrusions*, but have a larger or smaller degree of variability. These intrusions often exploit more general flaws in the attacked system, and there is much inherent opportunity for variation in the specific attack that is executed.

Unknown intrusions These intrusions have the weakest coupling to a specific flaw, or one that is very general in nature. Here, the intrusion detection system does not really know what to expect.

4.4.3 Examples of systems in the high level categories

A few examples of systems that correspond to these three classes will serve to illustrate how the surveyed systems fall into these three categories.

Well known intrusions First we have the simple, signature systems that correspond to the *well known intrusions* class. The more advanced (such as IDIOT) and general (such as P-BEST in EMERALD) move towards the *generalisable intrusions* class by virtue of their designers' attempts to abstract the signatures, and take more of the expected normal behaviour into account when specifying signatures [LP99]. However, the model of normal behaviour is still very weak, and is often not outspoken; the designers draw on experience rather than on theoretical research in the area of expected source behaviour.

Generalisable intrusions The *generalisable intrusions* category is only thinly represented in the survey. It corresponds to a signature of an attack that is 'generalised' (specified on a less detailed level) leaving unspecified all parameters that can be varied while still ensuring the success of the attack. Some of the more advanced signature based systems have moved towards this ideal, but still have a long way to go. The problem is further compounded the systems' lack of an clear model of what constitutes normal traffic. It is of course more difficult to specify a general signature, while remaining certain that the normal traffic will not trigger the detection system. The only example of a self-learning, compound system (RIPPER) is interesting, since by its very nature it can accommodate varying intrusion signatures merely by being confronted by different variations on the same theme of attack. It is not known how easy or difficult it would be in practice to expand its knowledge of the intrusive process by performing such variations. It is entirely possible that RIPPER would head in the opposite direction and end up over-specifying the attack signatures it is presented with, which would certainly lower the detection rate.

Unknown intrusions The third category, *unknown intrusions*, contains the anomaly detectors, in particular because they are employed to differentiate between two different users. This situation can be modelled simply as the original user (stochastic process) acting in his normal fashion, and a masquerader (different stochastic process) acting intrusively, the problem then becoming a clear example of the uncertainties inherent detecting an unknown intrusion against a background of normal behaviour. This is perhaps the only case where this scenario corresponds directly to a security violation, however. In the general case we wish to employ anomaly detectors to detect intrusions that are novel to us, i.e. where the signature is not yet known. The more advanced detectors such as the one described in [LB98] do indeed operate by differentiating between different stochastic models. However, most of the earlier ones, that here fall under the descriptive statistics heading, operate without a source model, opting instead to learn the behaviour of the background traffic and flag events that are unlikely to emanate from the known, non-intrusive model.

In the case where we wish to detect intrusions that are novel to the system, anomaly detectors would probably operate with a more advanced source model. Here the strategy of flagging behaviour that is unlikely to have emanated from the normal (background) traffic probably results in less of a direct match. It should be noted that research in this area is still in its infancy. Anomaly detection systems with an explicit intrusive source model would probably have interesting characteristics although we will refrain from making any predictions here.

4.4.4 Effect on detection and false alarm rates

It is natural to assume that the more difficult the problem (according to the scale presented in section 4.4.2), the more difficult the accurate intrusion detection decision, but it is also the case that the intrusion detection problem to be solved becomes more interesting as a result. This view is supported by the classification above, and the claims made by the authors of the classified systems, where those who have used 'well known intrusions' detection are the first to acknowledge that modifications in the way the system vulnerability is exploited may well mean the attack goes undetected [Pax88, HCMM92, HDL+90]. In fact, these systems all try to generalise their signature patterns to the maximum in order to avoid this scenario, and in so doing also move towards the 'generalisable intrusions' type of detector. The resulting detector may well be more robust, since less specific information about the nature of its operation is available to the attacker.

RIPPER is the only tool in this survey that can claim to take both an intrusion model and normal behaviour model into account. Thus it is able to make a complex, threshold based decision, not two tiered as is the case with some others. These detectors will by their very nature resemble signature based systems since we still have a very incomplete knowledge of what intrusive processes look like in computer systems. These systems will in all likelihood never detect an intrusion that is novel to them. However, by employing a complete source model, they promise the best performance with regard to detection and false alarm rates. Indeed, if the source models can be specified with sufficient (mathematical) accuracy, a theoretically optimal, or near optimal, detector can be constructed, and its optimal detection and false alarm rates calculated.

The main interest in intrusion detectors is of course in the degree to which they can correctly classify intrusions and avoid false alarms. We call this parameter *effectiveness*. There has not been much study of the effectiveness of intrusion detection systems to date, although in the past year some work in this field has been presented [LGG+98, WFP99]. These studies are still weak when it comes to the classification both of intrusions and of signal and channel models, so it is difficult to draw any conclusions from them. The one paper which is both strong on the modelling and the claims it makes is [HL93], and interestingly the work presented there suggests that some types of intrusion detectors—those that fall into the *anomaly—self learning—non-time series* category above—could have a difficult time living up to realistic effectiveness goals, as claimed in [Axe99].

Furthermore, the *anomaly—programmed—descriptive stat* class of detectors, is probably not as interesting, especially in light of how easily they are avoided once the attacker has learned of their existence. They are probably useful as a first step on which to base further intrusion detection decisions. It should be noted that the systems surveyed that use this technique are all old by today's standards. We do not know why this line of research was not continued.

5 A taxonomy of system characteristics

Turning to the research systems, it became apparent that they consist of more than just a detector. For this reason we have built a taxonomy of 'system characteristics,' i.e those characteristics that do not pertain directly to the detection principle. Having thus divided the groups of systems based on their approach to detecting an intrusion in audit data, a closer study reveals the following dichotomies in other system characteristics.

Time of detection Two main groups can be identified: those that attempt to detect intrusions in *real-time* or near real-time, and those that process audit data with some delay, postponing detection (*non-real-time*), which in turn delays the time of detection. Without any real exceptions, the surveyed systems that fall into the real-time category can also be run, off-line, on historical audit data. This is most likely because it makes it possible to simplify the verification process as the system is being developed, but of course it can sometimes be valuable to run an otherwise real-time capable system on previously saved data to establish past, security-critical events.

Granularity of data-processing This is a category that contrasts systems that process data *continuously* with those that process data in *batches* at a regular interval. This category is linked with the *time of detection* category above, but it should be noted that they do not overlap, since a system could process data continuously with a (perhaps) considerable delay, or could process data in (small) batches in 'real-time'.

Source of audit data The two major sources of audit data in the surveyed systems are *network data* (typically data read directly off a multicast network such as Ethernet) and *host based* security logs. The host based logs can include operating system kernel logs, application program logs, network equipment (such as routers and firewalls) logs, etc.

Response to detected intrusions *Passive* versus *active*. Passive systems respond by notifying the proper authority, and they do not themselves try to mitigate the damage done, or actively seek to harm or hamper the attacker. Active systems may be further divided into two classes:

1. Those that exercise control over the *attacked* system, i.e. they modify the state of the attacked system to thwart or mitigate the effects of the attack. Such control can take the form of terminating network connections, increasing the security logging, killing errant processes, etc.
2. Those that exercise control over the *attacking* system, i.e they in turn attack in an attempt to remove the attacker's platform of operation. Since this is difficult to defend in court, we do not envision much interest in this approach outside military or law enforcement circles.

Of the systems surveyed, one severs network connections in response to suspected attacks, and one blocks suspect system calls, terminating the process if this option fails. This mode of defence is generally difficult to field since it opens up the system to obvious denial of service attacks.

Locus of data-processing The audit data can either be processed in a *central* location, irrespective of whether the data originates from one—possibly the same—site or is collected and collated from many different sources in a *distributed* fashion.

Locus of data-collection Audit data for the processor/detector can be collected from many different sources in a *distributed* fashion, or from a single point using the *centralised* approach.

Security The ability to withstand hostile attack against the intrusion detection system itself. This area has been little studied. A basic classification would use a *high*—*low* scale. The surveyed systems, with one exception, all fall into the latter category.

Degree of inter-operability The degree to which the system can operate in conjunction with other intrusion detection systems, accept audit data from different sources, etc. This is not the same as the number of different platforms on which the intrusion detection system itself runs.

In fairness it should be said that not all of these categories are dichotomies in the true sense. However, the author believes that many of the surveyed systems display sufficient differences for it to be meaningful to speak of a dichotomy.

Table 2: Classification of the surveyed systems according to system characteristics

Name of system	Publ. year	Time of detection	Granularity	Audit source	Type of response	Data-processing	Data-collection	Security	Inter-oper.
Haystack [Sma88]	1988	non-real	batch	host	passive	centralised	centralised	low	low
MIDAS [SSHW88]	1988	real	continuous	host	passive	centralised	centralised	low	low
IDES [LJL+88]	1988	real	continuous	host	passive	centralised	distributed	low	low
W&S [VL89]	1989	real	continuous	host	passive	centralised	centralised	low	low[a]
Comp-Watch [DR90]	1990	non-real	batch	host	passive	centralised	centralised	low	low
NSM [HDL+90]	1990	real	continuous	network[b]	passive	centralised	centralised [c]	low	low[d]
NADIR [JDS91]	1991	non-real	continuous	host[e]	passive	centralised	distributed	low	low
Hyperview [DBS92]	1992	real	continuous	host	passive	centralised	centralised	low	low
DIDS [SSTG92]	1992	real	continuous	both[f]	passive	distributed	distributed	low	low[g]
ASAX [HCMM92]	1992	real[h]	continuous[i]	host	passive	centralised	centralised	low	higher[j]
USTAT [Ilg93]	1993	real	continuous	host	passive	centralised	centralised	low	low[k]
DPEM [KFL94]	1994	real	batch	host	passive	distributed	distributed	low	low
IDIOT [KS94b]	1994	real[l]	continuous	host	passive	centralised	centralised	low	higher
NIDES [AFV95]	1995	real[m]	continuous	host[n]	passive	centralised	distributed	low[o]	higher[p]
GrIDS [fCCCf+96]	1996	non-real	batch	both[q]	passive	distributed	distributed	low	low
CSM [WP96]	1996	real	continuous	host	active[r]	distributed	distributed	low	low
Janus [GWTB96]	1996	real	continuous	host	active[s]	centralised	centralised	low	low
JiNao [JGS+97]	1997	real	batch	'host'[t]	passive	distributed	distributed	low	low
EMERALD [PN97]	1997	real	continuous	both	active	distributed	distributed	moderate	high
Bro [Pax88]	1998	real	continuous	network	passive	centralised	centralised [u]	higher	low

[a] The authors clearly intend the method to be generally useful. [b] The first system to utilise the raw network traffic as a source of audit data. [c] One central network tap. [d] The use of network protocols common to several computer and operating system architectures lends some inter-operability from that perspective. [e] The hosts record information about network events, but the network is not used directly as a source of audit data. [f] DIDS has components that monitor both individual hosts (Hyperview) and network traffic (NSM) for audit data. [g] The network monitoring component lends some inter-operability from a platform and/or operating system perspective. [h] Deduced from the proposed architecture. [i] Deduced from the proposed architecture. [j] The authors discuss the issue in some detail, and present a proposed architecture to remedy the problem somewhat. [k] The prototype presented is platform specific, and the authors discuss the general applicability of the method. [l] The user of the system can make the choice between real or non-real time processing. [m] Both types of processing are available. [n] NIDES can use host-based logs of network activity as input. [o] However, the authors are the first to discuss the problems with attacks against the intrusion detection system at length. [p] NIDES contains user support for easing the conversion of proprietary audit trails to NIDES format. [q] Primarily a network monitoring tool. [r] The prototype version is passive, but the authors foresee a future version to sever the connection that the intruder appears to be utilising. [s] The most active system surveyed. [t] The audit logs originate from network *equipment*, not the network directly, through network 'sniffing' hence the 'host' classification. [u] One central network tap.

5.1 A classification of system characteristics

Applying our taxonomy to the surveyed systems leads to the classification given in table 2 on the preceding page.

5.2 A short discussion of trends and constants

Since not enough is known about all the systems surveyed some have been left out of the classification. Since we have tabulated the system features according to the year of publication it becomes easy to note a number of trends and constants in the features surveyed.

5.2.1 Trends

Reading from left to right, the first trend we encounter is in the *type of response* category, where we see an interest in active response of late. It should be pointed out that research here is still young, and security concerns about the intrusion detection system itself raise the possibility of the attacker exploiting an active response to effect a denial of service attack on the system being monitored, or an innocent third party. There are also the legal ramifications of active response.

We also see a trend from centralised to distributed intrusion detection. This probably follows the trend in computing in general during the period surveyed, and the fact that distributed intrusion detection is viewed as a way to make the intrusion detection system scale as the monitored system grows [PN97].

Interest in increasing the security of the intrusion detection system itself also seems to be on the rise; as intrusion detection comes of age this is perhaps only natural. There is also a trend towards increasing inter-operability, which would coincide with the increased commercial interest in intrusion detection in the past few years.

5.2.2 Constants

We note that real time intrusion detection has always been a goal, which is hardly surprising. Perhaps more remarkable is that the *audit source* category has such an overwhelming number of *host* entries. One might have thought that network intrusion detection, often much discussed, would have had a bigger impact on the research prototypes, but this is evidently not the case. Problems with network intrusion detection such as ever increasing network speeds and, more recently, encryption, are probably at least a partial reason for this.

6 Conclusions

The construction of a taxonomy of intrusion detection principles proves to be a fruitful exercise that provides us with many insights into the field, and a number of lines for further research.

When completing the survey one is struck by the lack of research earlier in the detection chain, into taxonomies and models of intrusive and normal behaviour, and into what information to observe in order to detect intrusions, etc.

The lack of detectors in the *signature—self-learning* class is conspicuous, particularly since detectors in this class would probably prove very useful, combining the advantages of self-learning systems—not having to perform the arduous and difficult task of specifying intrusion signatures—with the detection efficiency of signature based systems.

Two tiered detectors also show promise. Some of the oldest systems are designed according to these principles, but there is little research into the specific abilities of such systems.

In more general terms, signature based systems with a more explicit normal behaviour model and anomaly based systems with a better formed intrusion/attack model are of interest, if only for the fact that current systems are, almost without exception, firmly entrenched in one camp or the other.

Furthermore, when we divide current detectors into three groups according to the difficulty of the problem they address, it would be interesting to see to what degree those that fall into the higher classes are able to live up to the requirements of detection and false alarm rates. These problems are that much more difficult to solve, especially in view of more advanced attackers who may actively try to avoid being detected.

A few non-detection parameters were also studied and certain trends and constants were identified in past and present research, in particular the recent interest in *security*, *active response*, *distributed systems* and *inter-operability*.

Appendix A Details of the surveyed systems

In the following, the term 'authors' is used to refer to the authors of the work being surveyed, while we refer to ourselves as the 'present author.'

A.1 Haystack

The Haystack prototype [Sma88] was developed for the detection of intrusions in a multi-user Air Force computer system, then mainly a Unisys (Sperry) 1100/60 mainframe running the OS/1100 operating system. This was the standard Air Force computing platform at the time.

To detect intrusions the system employs two methods of detection: anomaly detection, and signature based detection. The anomaly detection is organised around two concepts; per user models of how users have behaved in the past, and pre-specified generic user group models that specify generic acceptable behaviour for a particular group of users. The combination of these two methods solves many of the problems associated with the application of any one of them in intrusion detection systems.

A.2 MIDAS: Expert systems in intrusion detection, a case study

MIDAS [SSHW88] was developed by the National Computer Security Centre (NCSC), in co-operation with the Computer Science Laboratory, SRI International, to provide intrusion detection for the NCSC's networked mainframe, Dockmaster (a Honeywell DPS-8/70). This computer was primarily used for electronic communication within the employee community at NCSC and affiliated agencies. The authors acknowledge previous work by Denning et al. and work at Sytek as their main sources of inspiration.

MIDAS is built around the idea of heuristic intrusion detection. The authors take as their example the way in which a human site security officer would go about analysing audit logs manually to find evidence of intrusive behaviour.

MIDAS applies the Production Based Expert System Toolset (P-BEST) for intrusion detection. P-BEST is written in Lisp, and produces Lisp code that can be compiled and run on a dedicated Symbolics Lisp machine. The compilation of the expert system code into object code provides for the efficient execution of the expert system shell.

In MIDAS, P-BEST's rule-base is populated with rules in three distinct categories. The structure of the rule base is two tiered. The first, lowest layer handles the immediate deduction about certain types of events such as 'number of bad logins', and asserts a fact to the effect that a particular threshold of suspicion has been reached when they fire. These suspicions are then processed by second layer rules that decide whether to proceed to raise an alarm based on the suspicion facts asserted by the lower level rules; for example, 'This user is a masquerader because he has made 40 command errors in this session, he has tried the invalid commands `suid` and `priv`, and he is logged in at an unusual time.' Taken together, this would be a strong indication that something is amiss, and the second level rule—representing a masquerader—would trigger, alerting the site security officer.

A.3 IDES—A real-time intrusion detection expert system

IDES is a classic intrusion detection system [LJL$^+$88, LTG$^+$92], and thus far one of the best documented. Strictly speaking there is no one IDES system, however, since the IDES project went on for a number of years, merging into the Next-Generation Intrusion Detection Expert System (NIDES) once the IDES project was officially complete. The IDES system thus underwent sometimes fundamental change as the research project progressed.

The basic motivation behind IDES is that users behave in a consistent manner from time to time, when performing their activities on the computer system, and that the manner in which they behave can be summarised by calculating various statistics for the user's behaviour. Current activity on the system can then be correlated with the calculated profile, and deviations flagged as (possibly) intrusive behaviour.

The 1988 prototype of IDES differed from the original prototype in many respects. It runs on two Sun-3 workstations, one Sun-3/260 that maintains the audit database and the profiles, and one Sun-3/60 that manages the SSO user interface. The audit database is implemented using a COTS Oracle DBMS. The monitored system is a DEC-2065 that runs a local version of the TOPS-20 operating system. The audit data is transmitted (securely) to IDES via the network one record at a time, and processed to provide a real-time response.

IDES process each new audit record as it enters the system, and verifies it against the known profile for both the subject, and the group of the subject should it belong to one. IDES also verifies each session against known profiles when the session is completed. To further distinguish different but authorised behaviour, the prototype was extended to handle two sets of profiles for monitored subjects depending on whether the activity took place on an 'on' or 'off' day. The SSO defines which days are in effect 'normal' working days for a particular subject, in this case mostly users, and those which are not. This further helps to increase the true detection rate since a finer notion of what is 'normal' for a particular subject, based on real-world heuristics, can be developed.

A.4 Wisdom & Sense—Detection of anomalous computer session activity

W&S [VL89] is another seminal anomaly detection system. Development began as early as 1984, with the first publication in 1989. It is interesting to note that W&S was not at first intended for application to computer security, but rather to 'a related problem in nuclear materials control and accountability.'[1] W&S is unique in its approach to anomaly detection: it studies historic audit data to produce a forest of rules describing 'normal' behaviour, forming the 'wisdom' of the title. These rules are then fed to an expert system that evaluates recent audit data for violations of the rules, and alerts the SSO when the rules indicate anomalous behaviour, thus forming the 'sense'.

W&S reads historic audit records from a file. The authors state that more is better when it comes to the creation of rules from previous audit records, of which there are about 10,000 records per user. The authors consider a figure of around 500-1,000 audit records per user a good target to aim for. The audit records that are used typically record one event for each process execution, doing so at the end of the execution of the process.

The 'sense' element reads audit records, evaluates the thread class of which they are part against the rule base, and triggers an alarm if sufficient numbers of rules report enough of a discrepancy—a high enough score—with the profile. The score of the thread (named figure of merit (FOM) by the authors) is a sum of the FOMs for that thread's audit records. This sum is aged. Thus several events that are slightly anomalous across several sessions will eventually accumulate to form an anomaly for that thread.

The 'sense' element of W&S then reads the rule-base, dictionary, and new audit records, either in batches or as they become available. The inference engine then processes each audit record, finding the rules that apply, and computes its transaction score. In so doing, the inference engine basically sums all contributions from the different failed rules—bear in mind that we are searching for anomalies—and the rules that describe 'normal' behaviour, taking the thread of which the audit record is part into account. The thread score is updated, and aged by the previous process. W&S then reports an anomaly whenever the thread score, or individual audit record score, exceeds an operator defined threshold.

A.5 The ComputerWatch data reduction tool

The ComputerWatch [DR90] data reduction tool was developed as a commercial venture by the Secure Systems Department at AT&T Bell Laboratories as an add-on package for use with the AT&T System V/MLS. System V/MLS is a B1 evaluated version of System V UNIX that provides multi-level security features that comply with the NCSC orange book B1 security criteria.

The ComputerWatch tool operates on the host audit trail to provide the SSO with a summary of system activity, from which he can decide whether to take action by investigating further those statistics that seem anomalous. The tool then provides the SSO with the necessary mechanisms for making specific inquiries about particular users and their activities, based on the audit trail.

[1] The authors were working at Los Alamos National Laboratory and Oak Ridge National Laboratory at the time of publication. These facilities have strong ties with the US nuclear weapons program.

ComputerWatch would normally be used by the SSO with some periodicity, in an attempt to establish an overview of the type of system activity that has occurred. The tool provides the SSO with a summary report of this activity. The report can be perused as is, or certain entries can be automatically highlighted by the system according to a set of predefined rules to provide a form of threshold highlighting capacity. The SSO then decides what type of activity, if any, merits further study, and can make specific enquiries about users and their activities, using the audit trail database.

A.6 NSM—Network security monitor

The most recent version of NSM [HDL+90, MHL94] is discussed here. NSM was the first system to use network traffic directly as the source of audit data. NSM listens passively to all network traffic that passes through a broadcast LAN, and deducts intrusive behaviour from this input. This approach stems from the observation that even though several other intrusion detection systems try to ameliorate the problems of having several different forms of audit trails available from different platforms, the network traffic between these systems typically takes place on broadcast networks, utilising standard protocols such as TCP/IP, `telnet`, `ftp`, etc. Thus NSM can monitor a network of heterogeneous hosts without having to convert a multitude of audit trail formats into a canonical format.

NSM follows a layered approach, termed the Interconnected Computing Environment Model (ICEM). The *connection layer* is responsible for studying the network data, and attempts to form pairs of bi-directional communication channels between sets of hosts. These connections are condensed into a *connection vector*, pruning away some of the data gained from the lower layers. In the system described, only the host vectors and connection vectors are used as input to a simple expert system that analyses the data for intrusive behaviour. The expert system draws on several other inputs, such as the profiles of expected traffic behaviour. These profiles consist of expected data-paths that describe which systems are expected to communicate with which, using what protocols. Another type of profile is constructed for each kind of higher-level protocol, for example what a typical `telnet` session looks like.

Other types of input are knowledge of the various capabilities of the protocols; for example, `telnet` is a powerful protocol that enables the user to perform a variety of tasks—and of how well these protocols authenticate their requests. `Telnet` authenticates its requests, while `sendmail` requests identification, but does not authenticate this identification. The data from these sources is combined to make a decision about the likelihood that a particular connection represents intrusive behaviour, based on anomaly reasoning. This is combined into a concept of the *security state* of the connection.

The default presentation of the data to the SSO takes the form of a sorted list, where each row in the list consists of a connection vector and the computed suspicion level. The results are also stored in a database, enabling the SSO to select specific events he would like to study more closely.

A.7 NADIR—An automated system for detecting network intrusion and misuse

NADIR [JDS91, HJS+93] was developed at the Los Alamos National Laboratory, for use by the laboratory in its internal computer security.[2] Thus NADIR was conceived with the problems and organisational needs of the Los Alamos National Laboratory in mind.

NADIR is implemented on a Sun SPARCstation II using the Sybase relational database management system. NADIR collects audit information from three different kinds of service nodes. The audit data is collected and subjected to extensive processing before being entered into the relational database as audit information. The audit data consists of data pertaining to the different kinds of service nodes, and the network traffic that they generate and receive.

Each audit record entered into NADIR pertains to a specific event. The information for the event is logged: whether the corresponding action succeeded or not and contains a unique ID for the ICN user, the date and time, an accounting parameter, and an error code. The remainder

[2] It is not known what influence W&S (see section A.4) had on the development of NADIR.

of the record describes the event itself. This description varies depending on the kind of service node from which it originates.

NADIR calculates an individual user profile on a weekly basis, where each record summarises the user's behaviour. The profile contains static information about the user, historic information about the user (the number and a list of the different source machines from which the user has attempted to login to the ICN), blacklist (the number of times and the date upon which a user was last blacklisted[3]), and so on. The user activity field contains account statistics for different types of activity during the week for the three classes of service nodes; in other words, from the eight counters that tally all attempted logins from source machines in each of the four partitions.

These profiles are then compared using a set of expert system rules. These rules were derived from a number of different sources. First and foremost, security experts were interviewed, and the security policy of the laboratory was encoded as a set of expert system rules. Second, statistical analysis of the audit records from the system was performed, and the results from this analysis were hard coded into the system as rules in the expert system.

NADIR produces several sets of reports about system activity that the SSO can inspect for indications of intrusive behaviour.

A.8 Hyperview—A neural network component for intrusion detection

Hyperview [DBS92] is a system with two major components. The first is an 'ordinary' expert system that monitors audit trails for signs of intrusion known to the security community. The second is a neural network based component that learns the behaviour of a user adaptively and raises the alarm when the audit trail deviates from this already 'learned' behaviour.

The decision to attempt to employ a neural network for the statistical anomaly detection function of the system stemmed from a number of hypotheses about what the audit trail would contain. The fundamental hypothesis was that the audit trail constitutes a multivariate time series, where the user constitutes a dynamic process that emits a sequentially ordered series of events. The audit record that represents such an event consists of variables of two types; the values of the first being chosen from a finite set of values—for instance the name of the terminal the command was issued on—the second having a continuous value, for example CPU usage or some such.

The authors proposed the then untested approach of mapping the time series to the inputs of the neural network. At the time, the usual approach was to map N inputs to a window of time series data, shifting the window by one between evaluations of the network. The authors acknowledged that this would make for a simple model that could be easily trained, for example. However, since there were a number of problems with this approach the authors decided on a different tack.

The designers of Hyperview chose to employ a recurrent network, where part of the output network is connected to the input network, forming input for the next stage. This creates an internal memory in the network. Between evaluations, the time series data is fed to the network one at a time rather than as a shifting time window. The purpose is the same, namely to provide the network with a perception of the past. It is interesting to note that the recurrent network has long term memory about the parameters of the process in the form of the weights of the connections in the network, and short term memory about the sequence under study in the form of the activation of the neurones. At the time the system was designed these kinds of networks had been studied far less than non-recurring networks.

The design of the system as a whole is a complex one. The authors chose to connect the artificial neural network to two expert systems. One monitors the operation and the training of the network—to prevent the network from learning anomalous behaviour for instance—and evaluates its output. The other expert system scans the audit trail for known patterns of abuse, and together with the output from the first expert system (and hence from the artificial neural network) forms an opinion on whether to raise the alarm or not. The decision expert system also provides the artificial neural network with 'situation awareness' data—data that the audit trail itself does not contain—from the simple 'current time and date,' to the complex 'state of alert, or state of danger for the system,' defined by the SSO.

[3] Blacklisted individuals lose their ICN privileges in the event of specific types of unauthorised behaviour.

A.9 DIDS—Distributed intrusion detection prototype

DIDS [SSTG92] is a distributed intrusion detection system that incorporates Haystack (see section A.1, page 15) and NSM (see section A.6, page 17) in its framework.

DIDS is made of up of three main components. On each host, a *host monitor* performs local intrusion detection, as well as summarising the results and parts of the audit trail for communication to the DIDS director. Furthermore each (broadcast) network segment houses its own *LAN monitor* that monitors traffic on the LAN, and reports to the DIDS director. Finally, the centralised *DIDS director* analyses material from the host monitors and the LAN monitors that report to it, and communicates the results to the SSO.

The director consists of two main parts: the communications manager, and the expert system. The communications manager is responsible for collecting the data sent to it from the host managers and LAN managers, and for communicating this data to the expert system for further processing.

The expert system draws inferences about the security state of the system and each individual host, and assembles the information for presentation to the SSO. The expert system is an ordinary rule-based (or *production* based) expert system. It is implemented in CLIPS, a C language expert system implementation from NASA.

A.10 ASAX—Architecture and rule-based language for universal audit trail analysis

The paper outlining ASAX [HCMM92] only describes a suggested prototype of the system, and hence it cannot be fully surveyed.

ASAX is a rule-based intrusion detection system with a specialised, efficient language (RUSSEL) for describing the rules. ASAX first converts the underlying (UNIX-like) operating system's audit trail into a canonical format—named NADF by the authors—and then processes the resulting audit trail in one pass by evaluating rules in the RUSSEL language.

The RUSSEL language is a declarative, rule-based language that is specifically tailored to audit trail analysis. The authors state that: 'a general purpose rule-based language should not necessarily allow encoding of any kind of declarative knowledge or making a general reasoning about that knowledge.' This is in contrast to more general expert systems, such a P-BEST (see sections A.2, A.3, and A.14), that the authors argue are more cumbersome for the SSO to use. Rules in the RUSSEL language are applied to each audit record sequentially. They encapsulate all the relevant knowledge about past results of the analysis in the form of new rules, and they are active only once, requiring explicit re-instantiation when they have fired.

A.11 USTAT—State transition analysis

USTAT [Ilg93, IKP95] is a mature prototype implementation of the state transition analysis approach to intrusion detection. State transition analysis takes the view that the computer initially exists in a secure state, but as a result of a number of penetrations—modelled as state transitions—it ends up in a compromised target state. USTAT reads specifications of the state transitions necessary to complete an intrusion, supplied by the SSO, and then evaluates an audit trail in the light of the specifications. The C2 audit trail produced by the computer is used as the source of information about the system's state transitions.

The SSO specifies the intrusive behaviour he wishes the IDS to detect as a sequence of specific state transitions. For example he could state that a particular file would have to exist, with a certain name, permissions and so on, in order for a state to exist. The system could then transition from this state to another state via a specified transition, for example the writing of the said file.

To apply state transition analysis to intrusion detection the authors make two provisions:

1. The intrusion must have a visible effect on the system state, and

2. The visible effect must be recognisable without knowledge external to the system, such as the attacker's true identity.

Not all possible intrusions meet these provisions. For instance, the passive monitoring of broadcast network traffic might be difficult to detect from outside the resource employed to perform the monitoring. Another intruder who is difficult to detect is the masquerader. However, if that masquerader then goes on to attempt any one of a number of intrusions to gain greater privileges, state transition will have a chance to catch him.

A.12 DPEM—Distributed program execution monitoring

The authors note that previous work on the the detection of the exploitation of previously known intrusions has focused on the patterns of use that arise from these exploitations [Ko96, KFL94, KRL97]. Instead they suggest that the opposite approach be taken, and that the intrusion detection system should focus on the correct security behaviour of the system, or more particularly a security privileged application that runs on the system as specified. The authors have designed a prototype—the distributed program execution monitor (DPEM) - that reads the security specifications of acceptable behaviour of privileged UNIX programs, and checks audit trails for violations of these security specifications.

The DPEM prototype, as its name suggests, monitors programs executed in a distributed system. This is accomplished by collecting execution traces from the various hosts, and where relevant distributing them across the network for processing. DPEM consists of a *director*, a *specification manager*, *trace dispatchers*, *trace collectors*, and *analysers* that are spread across the hosts of the network.

A.13 IDIOT—An application of Petri-nets to intrusion detection

IDIOT [KS94b, KS94a, Kum95, CDE+96, KS95] is a system that was developed at COAST (now the Center for Education and Research in Information Assurance and Security (CERIAS), http://www.cerias.purdue.edu The basic principle behind IDIOT is to employ coloured Petri-nets for signature based intrusion detection. The authors suggest that a layered approach should be taken when applying signature (pattern matching) based techniques to the problem of intrusion detection.

The authors argue that of the many available techniques of pattern matching, coloured Petri-nets (CP-nets) would be the best technique to apply since it does not suffer from a number of shortcomings common in other techniques. The latter do not allow conditional matching of patterns, and do not lend themselves to a graphical representation, etc.

The patterns play a major role in IDIOT. They are written in an ordinary textual language and then parsed, resulting in a new pattern matching engine. As noted, this engine can then be dynamically added to an already running IDIOT instance via the user interface. Furthermore, the user can extend IDIOT to recognise new audit events, for example.

A.14 NIDES—Next generation intrusion detection system

NIDES [AFV95, ALJ+95] is the successor to the IDES project (see section A.3). Like its predecessor it is very well documented, and there are many more references available than the two given here.

It is incorrect to speak of one single NIDES system—a feature it shares with its predecessor—as in fact there are four different systems, each constructed on top of the previous system. NIDES follows the same general principles as the later versions of IDES: it has a strong anomaly detection foundation, complemented with a signature based expert system component. The latter is implemented using the P-BEST expert system shell, a later version of P-BEST than the one presented in the survey of MIDAS (see section A.2), implemented in C and generating C as output.

The NIDES system is highly modularised, with well defined interfaces between components, built on a client-server architecture. The system is centralised to the extent that the analysis runs on a specific host - named the *NIDES host* by the authors—and collects data from various hosts through a computer network. The *target hosts* collect audit data from various host-based logs—there is a provision to utilise TCP WRAPPER [Ven92], viz. host-based network traffic logs—convert them into the canonical NIDES format, and transmits them to the NIDES host. The SSO interacts with the system through the NIDES host.

A.15 GrIDS—A graph based intrusion detection system for large networks

The authors suggest a method for constructing graphs of network activity in large networks to aid in intrusion detection [fCCCf+96]. The graphs typically codify hosts on the networks as nodes, and connections between hosts as edges between these nodes. The choice of traffic taken to represent activity in the form of edges is made on the basis of user supplied rule sets. The graph and the edges have respectively global and local attributes, including time of connection etc., that are computed by the user supplied rule sets. The authors argue that these graphs present network events in a graphic fashion that enables the viewer to determine if suspicious network activity is taking place.

Reporting all network activity in a single graph would be unwieldy. Therefore, the system allows several rule sets that define one graph apiece. All the collected data is considered for inclusion in all the rule sets, and thus two different rule sets could render the same audit data as two different graphs.

A.16 CMS—Co-operating security managers

The authors of co-operating security managers [WP96] note that as networks grow larger, centralised intrusion detection will not scale up well. To alleviate this problem they suggest that several intrusion detection agents, one at each computer connected to the network, co-operate in a distributed fashion, where the computer from which a user first entered the system is made responsible for all that user's subsequent actions. This, the authors claim, results in the load being evenly distributed among the co-operating entities.

A.17 Janus—A secure environment for untrusted helper applications

Janus [GWTB96] is a security tool inspired by the reference monitor concept that was developed at the University of California, Berkeley. While it is not an intrusion detection system per se, it shows many interesting similarities with *specification based* intrusion detection. Furthermore, its high degree of active influence over running applications makes it an interesting case-in-point when studying active response.

Janus is a user-space, per application reference monitor intended to supervise the running of potentially harmful web-browsing helper applications. It does this by enclosing the application in a restricted environment, a so-called 'sand box.'

When the framework detects that a policy module would disallow a certain system call, it aborts the system call with an *EINTR* (system call interrupted) error before it can be executed. To the supervised program this is indistinguishable from an interrupted system call, and some programs are designed to retry the system call when this condition becomes true. Janus detects this situation when a hundred invocations of the same system call have been denied, and then opts to kill the application completely.

A.18 JiNao—Scalable intrusion detection for the emerging network infrastructure

The authors have developed a prototype implementation of JiNao [JGS+97], a network intrusion detection system that aims to protect the network infrastructure itself, rather than the individual hosts on that network. The threat model assumes that certain routing entities in a network can be compromised, causing them to misbehave or stop routing altogether. The prototype assumes that the routers communicate via the OSPF protocol. It is hoped that the project will eventually detect both external and internal intrusion attempts on the network infrastructure, in a comprehensive and scalable fashion, inter-operating with other intrusion detection systems.

The authors state that intrusion detection in JiNao is operated using three different paradigms: misuse based detection, anomaly based detection, and protocol based (misuse) detection.

A.19 EMERALD—Event monitoring enabling responses to anomalous live disturbances

EMERALD [PN97, PV98] is intended as a framework for scalable, distributed, inter-operable computer and network intrusion detection. The authors begin by describing a situation in which large, organic computing and network resources provide critical and costly service to their operators, yet offer little in the way of specific security policies, or indeed organisational support for the specification of such policies. These large computing resources typically contain commercial off-the-shelf (COTS) components, as well as non-COTS components and legacy systems integrated with current technology. These infrastructures clearly need to be protected, and yet there is little in the way of widely available, robust tools to detect and track intruders. It is intended that EMERALD will contain components to enable the system to respond actively to the threats posed, principally by an attacker external to the organisation or at least external on some level. However, the proposed architecture does not preclude the detection of internal attackers.

The authors envisage a distributed system that operates on three different levels in a large enterprise network, made up of administratively more or less separate domains. These domains trust each other to a greater or lesser extent; two domains could operate in a peer-to-peer relationship, while third might trust virtually no-one else, only allowing out-bound connections. The enterprise network would typically be made up of thousands of entities.

EMERALD specifies well-defined interfaces on many levels, both internal to the EMERALD monitor and external to it, to enable other existing intrusion detection components to inter-operate with it. These components could be plugged in as an internal module in the monitor, or join in the intrusion detection effort via the network interface. To resolve these two situations, EMERALD defines a two layered, subscription based, message passing communication system and interface. The idea is that this will enable a completely implementation-neutral path of communication—both internally in the EMERALD monitor and externally—between monitors in the distributed EMERALD system.

A.20 Bro

Bro [Pax88] is, in the words of its authors, 'A stand-alone system for detecting network intruders in real-time by passively monitoring a network link over which the intruder's traffic transits.' The designers envisaged their tool meeting the following design goals and requirements (from [Pax88]):

1. It would make high-speed, large volume monitoring of network traffic possible.
2. It would not drop packets, i.e. it would be able to process incoming packets at a sufficient rate not to have to discard input packets before they had been processed.
3. Bro would provide the SSO with real-time notification of ongoing, or attempted, attacks.
4. Bro would be careful to separate mechanism from policy, making it simple to specify new security policies, and aiding in the realisation of both simplicity and flexibility in the system.
5. The system would be extensible, and above all it would be easy to add knowledge of new types of attack.
6. Bro would aid the user in avoiding simple mistakes in the specification of the security policy.

The intrusion detection system would have to operate in an environment in which it could come under attack. The construction of resilient security systems has attracted little research, so the designers chose to simplify matters by assuming that only one of two systems communicating would be subverted. The authors note that this assumption would cost practically nothing, since if the intruder had both systems under his control, he could establish intricate covert channels between them, and hence avoid detection anyway.

Bro is realised as a single-point, network monitoring, policy based system, that contains both default deny and default permit elements in its detection. It is capable of monitoring the full data stream of an Internet access point consisting of an FDDI interface.

The paper describing Bro [Pax88] is notable for being the first paper to address the problem of the kinds of attacks the monitor itself must be capable of withstanding. Previous work in this field had not tackled the resistance of the intrusion detection mechanism against malicious attacks other than in theory.

The authors divide network attacks into three categories to use in a discussion of the success with which Bro withstands an attack.

Overload attacks The aim of the attack is to overload the monitor with data to the point where it fails to keep up with the stream of data which it has to handle.

Crash attacks The purpose of the attack is to make the network monitor stop working altogether.

Subterfuge attacks The attacker sets out to mislead the monitor as to the meaning of the traffic it is analysing.

Bro contains mechanisms to withstand attacks in all these categories that meet with varying success.

A.21 RIPPER

RIPPER [Lee99] is tool inspired by data mining for the automatic and adaptive construction of intrusion detection models. The central idea is to utilise auditing programs to extract an extensive set of features that describe each network connection or host session, and to apply data mining programs to learn rules that accurately capture the behaviour of intrusions and normal activities. These rules can then be used for signature detection and anomaly detection.

Data mining generally refers to the process of extracting descriptive models from large stores of data. The data mining field has utilised a wide variety of algorithms, drawn from the fields of statistics, pattern recognition, machine learning, and databases. Several types of algorithms are particularly useful for mining audit data:

Classification places data items into one of several predefined categories. The algorithms normally output *classifiers*, for example, in the form of decision trees. In intrusion detection one would want to gather sufficient *normal* and *abnormal* data and then apply a classification algorithm to teach a classifier to label or predict new, unseen audit data as belonging to the normal or abnormal class

Link analysis determines relations between fields in the database records. Correlation of system features in audit data—for example, between *command* and *argument* in the shell command history data of a user—can serve as the basis for constructing normal usage profiles. A programmer may have *emacs* strongly associated with *C* files, for instance.

Sequence analysis models sequential patterns. These algorithms can determine what time-based sequence of audit events occur frequently together. These frequent event patterns provide guidelines for incorporating temporal statistical measures into intrusion detection models. For example, patterns from audit data containing network-based denial of service attacks suggest that several per host and per service measures should be included.

RIPPER is a framework that contains programs implementing several algorithms from all these classes. The user can pre-process audit data to provide the algorithms with a representation on a natural level; we want to present data to the system at the highest possible level of abstraction that still accurately captures all the technical intricacies of the problem. The user then gives RIPPER the task of automatically mining patterns of abusive or normal behaviour.

RIPPER was part of the DARPA evaluation [LGG+98], and the authors make strong claims for its performance. It is said to have had the best overall performance of all the tested tools. The authors concur that in order to detect new or novel intrusions, anomaly detection has to be employed. RIPPER as a signature detection tool does not produce signatures of a sufficiently general nature.

References

[AFV95] D Anderson, T Frivold, and A Valdes. Next-generation intrusion-detection expert system (NIDES). Technical Report SRI-CSL-95-07, Computer Science Laboratory, SRI International, Menlo Park, CA 94025-3493, USA, May 1995.

[ALGJ98] Stefan Axelsson, Ulf Lindqvist, Ulf Gustafson, and Erland Jonsson. An approach to UNIX security logging. In *Proceedings of the 21st National Information Systems Security Conference*, pages 62–75, Crystal City, Arlington, VA, USA, 5–8 October 1998. NIST, National Institute of Standards and Technology/National Computer Security Center.

[ALJ+95] Debra Anderson, Teresa F. Lunt, Harold Javitz, Ann Tamaru, and Alfonso Valdes. Detecting unusual program behavior using the statistical component of the next-generation intrusion detection system (NIDES). Technical Report SRI-CSL-95-06, Computer Science Laboratory, SRI International, Menlo Park, CA, USA, May 1995.

[Axe99] Stefan Axelsson. The base-rate fallacy and its implications for the difficulty of intrusion detection. In *6th ACM Conference on computer and communications security*, pages 1–7, Kent Ridge Digital Labs, Singapore, 1–4 November 1999.

[CDE+96] Mark Crosbie, Bryn Dole, Todd Ellis, Ivan Krsul, and Eugene Spafford. *IDIOT—Users Guide*. The COAST Project, Dept. of Computer Science, Purdue University, West Lafayette, IN, USA, 4 September 1996. Technical Report TR-96-050.

[DBS92] Herve Debar, Monique Becker, and Didier Siboni. A neural network component for an intrusion detection system. In *Proceedings of the 1992 IEEE Computer Sociecty Symposium on Research in Security and Privacy*, pages 240–250, Oakland, CA, USA, May 1992. IEEE, IEEE Computer Society Press, Los Alamitos, CA, USA.

[DDW99] Hervé Debar, Marc Dacier, and Andreas Wespi. Towards a taxonomy of intrusion-detection systems. *Computer Networks*, 31(8):805–822, April 1999.

[DR90] Cheri Dowel and Paul Ramstedt. The computer watch data reduction tool. In *Proceedings of the 13th National Computer Security Conference*, pages 99–108, Washington DC, USA, October 1990. NIST, National Institute of Standards and Technology/National Computer Security Center.

[ESP95] M Esmaili, R Safavi, Naini, and J Pieprzyk. Intrusion detection: A survey. In *Proceedings of ICCC'95. (12th International Conference on Computer Communication)*, volume xxxxii+862, pages 409–414. IOS Press, Amsterdam, Netherlands, 1995.

[FC93] Robert Flood and Ewart Carson. *Dealing with complexity—An introduction to the theory and application of systems science*, chapter 8, pages 151–158. Plenum Press, New York, second edition, 1993.

[fCCCf+96] S. Stani ford Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K Levitt, C. Wee, R. Yip, and D. Zerkle. GrIDS—A graph based intrusion detection system for large networks. In *Proceedings of the 19th National Information Systems Security Conference*, 1996.

[GWTB96] Ian Goldberg, David Wagner, Randi Thomans, and Eric Brewer. A secure environment for untrusted helper applications (confining the wily hacker). In *Proceedings of the Sixth USENIX UNIX Security Symposium*, San Jose, California, USA, July 1996. USENIX, USENIX Association.

[HCMM92] Jani Habra, Baudouin Le Charlier, Abdelaziz Mounji, and Isabelle Mathieu. ASAX: Software architecture and rule-based language for universal audit trail analysis. In Yves Deswarte et al., editors, *Computer Security – Proceedings of ESORICS 92*, volume 648 of *LNCS*, pages 435–450, Toulouse, France, 23–25 November 1992. Springer-Verlag.

[HDL+90] Todd Heberlein, Gihan Dias, Karl Levitt, Biswanath Mukherjee, Jeff Wood, and David Wolber. A network security monitor. In *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy*, pages 296–304. IEEE, IEEE Comput. Soc. Press, Los Alamitos, CA, USA, 1990.

[HJS+93] Judith Hochberg, Kathleen Jackson, Cathy Stallings, J. F. McClary, David DuBois, and Josehpine Ford. NADIR: An automated system for detecting network intrusion and misuse. *Computers & Security*, 12(3):235–248, 1993.

[HL93] Paul Helman and Gunar Liepins. Statistical foundations of audit trail analysis for the detection of computer misuse. *IEEE Transactions on Software Engineering*, 19(9):886–901, September 1993.

[IKP95] Koral Ilgun, Richard A Kemmerer, and Phillip A Porras. State transition analysis: A rule-based intrusion detection approach. *IEEE Transactions on Software Engineering*, 21(3):181–199, March 1995.

[Ilg93] Koral Ilgun. USTAT: A real-time intrusion detection system for UNIX. In *Proceedings of the 1993 IEEE Symposium on Security and Privacy*, pages 16–28, Oakland, California, 24–26 May 1993. IEEE Computer Society Press.

[JDS91] Kathleen A Jackson, David H DuBois, and Cathy A Stallings. An expert system application for network intrusion detection. In *Proceedings of the 14th National Computer Security Conference*, pages 215–225, Washington, D.C., 1–4 October 1991. National Institute of Standards and Technology/National Computer Security Center.

[JGS+97] Y. Frank Jou, Fengmin Gong, Chandru Sargor, Shyhtsun Felix Wu, and Cleaveland W Rance. Architecture design of a scalable intrusion detection system for the emerging network infrastructure. Technical Report CDRL A005, Dept. of Computer Science, North Carolina State University, Releigh, N.C, USA, April 1997.

[KFL94] Calvin Ko, George Fink, and Karl Levitt. Automated detection of vulnerabilities in privileged programs by execution monitoring. In *Proceedings of the 10th Annual Computer Security Applications Conference*, volume xiii, pages 134–144. IEEE, IEEE Computer Society Press, Los Alamitos, CA, USA, 1994.

[Ko96] Calvin Ko. *Execution Monitoring of Security-critical Programs in a Distributed System: A Specification-based Approach*. PhD thesis, Department of Computer Science, University of California at Davis, USA, 1996.

[KRL97] Calvin 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*, volume ix, pages 175–187, Oakland, CA, USA, May 1997. IEEE, IEEE Computer Society Press, Los Alamitos, CA, USA. IEEE Cat. No. 97CB36097.

[KS94a] Sandeep Kumar and Eugene H. Spafford. An application of pattern matching in intrusion detection. Technical Report CSD-TR-94-013, The COAST Project, Dept. of Computer Sciences, Purdue University, West Lafayette, IN, USA, 17 June 1994.

[KS94b] Sandeep Kumar and Eugene H. Spafford. A pattern matching model for misuse intrusion detection. In *Proceedings of the 17th National Computer Security Conference*, pages 11–21, Baltimore MD, USA, 1994. NIST, National Institute of Standards and Technology/National Computer Security Center.

[KS95] Sandeep Kumar and Eugene H. Spafford. A software architechture to support misuse intrusion detection. Technical report, The COAST Project, Dept. of Comp. Sciences, Purdue Univ.,West Lafayette, IN, 47907–1398, USA, 17 March 1995.

[Kum95] Sandeep Kumar. *Classification and Detection of Computer Intrusions*. PhD thesis, Purdue University, West Lafayette, Indiana, August 1995.

[LB98] Terran Lane and Carla E. Brodie. Temporal sequence learning and data reduction for anomaly detection. In *5th ACM Conference on Computer & Communications Security*, pages 150–158, San Francisco, California, USA, 3–5 November 1998.

[Lee99] Wenke Lee. A data mining framework for building intrusion detection MOdels. In *IEEE Symposium on Security and Privacy*, pages 120–132, Berkeley, California, May 1999.

[LGG+98] Richard P. Lippmann, Isaac Graf, S. L. Garfinkel, A. S. Gorton, K. R. Kendall, D. J. McClung, D. J. Weber, S. E. Webster, D. Wyschogrod, and M. A. Zissman. The 1998 DARPA/AFRL off-line intrusion detection evaluation. Presented to The First Intl. Workshop on Recent Advances in Intrusion Detection (RAID-98), Lovain-la-Neuve, Belgium, *No printed proceedings*, 14–16 September 1998.

[LJ97] Ulf Lindqvist and Erland Jonsson. How to systematically classify computer security intrusions. In *Proceedings of the 1997 IEEE Symposium on Security & Privacy*, pages 154–163, Oakland, CA, USA, 4–7 May 1997. IEEE, IEEE Computer Society Press, Los Alamitos, CA, USA.

[LJL+88] Teresa F. Lunt, R. Jagannathan, Rosanna Lee, Sherry Listgarten, David L. Edwards, Peter G. Neumann, Harold S. Javitz, and Al Valdes. IDES: The enhanced prototype, A real-time intrusion detection system. Technical Report SRI Project 4185-010, SRI-CSL-88-12, CSL SRI International, Computer Science Laboratory, SRI Intl. 333 Ravenswood Ave., Menlo Park, CA 94925-3493, USA, October 1988.

[LP99] Ulf Lindqvist and Porras Phillip. Detecting computer and network misuse through the production-based expert system toolset (P-BEST). In *1999 IEEE Symposium on Security and Privacy*, pages 146–161, May 1999.

[LTG+92] Teresa F. Lunt, Ann Tamaru, Fred Gilham, R. Jagannathan, Caveh Jalali, and Peter G. Neuman. A real-time intrusion-detection expert system (IDES). Technical Report Project 6784, CSL, SRI International, Computer Science Laboratory, SRI Intl. 333 Ravenswood Ave., Menlo Park, CA 94925-3493, USA, February 1992.

[Lun88] Teresa F Lunt. Automated audit trail analysis and intrusion detection: A survey. In *Proceedings of the 11th National Computer Security Conference*, pages 65–73, Baltimore, Maryland, 17–2 October 1988. NIST.

[MCZH99] Stefanos Manganaris, Marvin Christensen, Dan Zerkle, and Keith Hermiz. A data mining analysis of RTID alarms. In *2nd International workshop on recent advances in intrusion detection*, West Lafayette, Indiana, USA, 7–9 September 1999. Purdue University, CERIAS, CERIAS.

[MDL+90] N. McAuliffe, Wolcott. D, Schaefer. L, Kelem. N, Hubbard. B, and Haley. T. Is your computer being misused? A survey of current intrusion detection system technology. In *Proceedings of the Sixth Annual Computer Security Applications Conference*, volume xx+451, pages 260–272. IEEE, IEEE Comput. Soc. Press, Los Alamitos, CA, USA, 1990. Cat.No.90TH0351–7.

[MHL94] Biswanath Mukherjee, L Todd Heberlein, and Karl N Levitt. Network intrusion detection. *IEEE Network*, 8(3):26–41, May/June 1994.

[NP89] Peter G Neumann and Donn B Parker. A summary of computer misuse techniques. In *Proceedings of the 12th National Computer Security Conference*, pages 396–407, Baltimore, Maryland, 10–13 October 1989.

[Pax88] Vern Paxon. Bro: A system for detecting network intruders in real-time. In *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX, USA, January 1988. USENIX, USENIX Association. Corrected version, §7 overstated the traffic level on the FDDI ring by a factor of two.

[PN97] Philip A Porras and Peter G Neumann. EMERALD: Event monitoring enabling responses to anomalous live disturbances. In *Proceedings of the 20th National Information Systems Security Conference*, pages 353–365, Baltimore, Maryland, USA, 7–10 October 1997. NIST, National Institute of Standards and Technology/National Computer Security Center.

[PV98] Phillip A Porras and Alfonso Valdes. Live traffic analysis of TCP/IP gateways. In *Proceedings of the 1998 ISOC Symposium on Network and Distributed Systems Security*, San Diego, California, 11–13 March 1998.

[Sma88] S. E. Smaha. Haystack: An intrusion detection system. In *Proceedings of the IEEE Fourth Aerospace Computer Security Applications Conference*, Orlando, FL, USA, December 1988. IEEE, IEEE Computer Society Press, Los Alamitos, CA, USA.

[SSHW88] Michael M. Sebring, Eric Shellhouse, Mary E. Hanna, and R. Alan Whitehurst. Expert systems in intrusion detection: A case study. In *Proceedings of the 11th National Computer Security Conference*, pages 74–81, Baltimore, Maryland, 17–20 October 1988. NIST.

[SSTG92] Steven R Snapp, Stephen E Smaha, Daniel M Teal, and Tim Grance. The DIDS (distributed intrusion detection system) prototype. In *Proceedings of the Summer USENIX Conference*, pages 227–233, San Antonio, Texas, 8–12 June 1992. USENIX Association.

[Ven92] Wietse Venema. TCP WRAPPER: Network monitoring, access control and booby traps. In *Proceedings of the 3rd USENIX UNIX Security Symposium*, pages 85–92, Baltimore, Maryland, 14–16 September 1992. USENIX Association.

[VL89] H S Vaccaro and G E Liepins. Detection of anomalous computer session activity. In *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, pages 280–289, Oakland, California, 1–3 May 1989.

[WFP99] Christina Warrender, Stephanie Forrest, and Barak Perlmutter. Detecting intrusions using system calls: Alternative data models. In *IEEE Symposium on Security and Privacy*, pages 133–145, Berkeley, California, May 1999.

[WP96] G. White and V. Pooch. Cooperating security managers: Distributed intrusion detection systems. *Computers & Security*, Vol. 15(No. 5):441–450, 1996. Elsevier Science Ltd.