A Preliminary Attempt to Apply Detection and Estimation Theory to Intrusion Detection

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

March 13, 2000

Abstract

Research into the automated detection of computer security violations is hardly in its infancy, yet little comparison has been made with the established field of detection and estimation theory, the results of which have been found applicable to a wide range of problems in other disciplines. This paper attempts such a comparison, studying the problem of intrusion detection by the use of the introductory models of detection and estimation theory. Examples are given from current intrusion detection situations, and it is concluded that there are sufficient similarities between the fields to merit further study.

1 Introduction

The field of automated computer security intrusion detection is coming of age. However, the method has not yet seen wide commercial use, and lately concerned voices have begun to be heard as to the actual benefits of the approach when compared with more and better perimeter defences, for example. This paper is a preliminary attempt to compare computer security intrusion detection—intrusion detection for short—and the field of detection and estimation theory. The latter is often thought to deal solely with problems in signal processing, such as are present in a radar detection situation or in digital radio communications. However, the general scientific methods that have been developed to deal with these problems also lend themselves to the study of problems in medical diagnosis etc. It is interesting to note that those working on intrusion detection have been slow to utilise these findings, and even though there are some disparities, we believe that the similarities warrant a closer look, and that many interesting results could be carried over to the field of computer security.

For an introduction to 'classical' detection theory see [Tre68], and for an overview of the intrusion detection field see [Axe00].

2 Classical detection theory

The problem of detecting a signal transmitted over a noisy channel is one of great technical importance, and has consequently been studied thoroughly for some time now. An introduction to detection and estimation theory is given in [Tre68], from which this section borrows heavily.

In classical binary detection theory (see figure 1) we should envisage a system that consists of a source from which originates one of two signals, *H0* or *H1*. This signal is transmitted via some channel that invariably adds noise and distorts the signal according to a probabilistic transition mechanism. The output—what we receive—can be described as a point in a finite (multidimensional) observation space, for example x in figure 1. Since this is a problem that has been studied

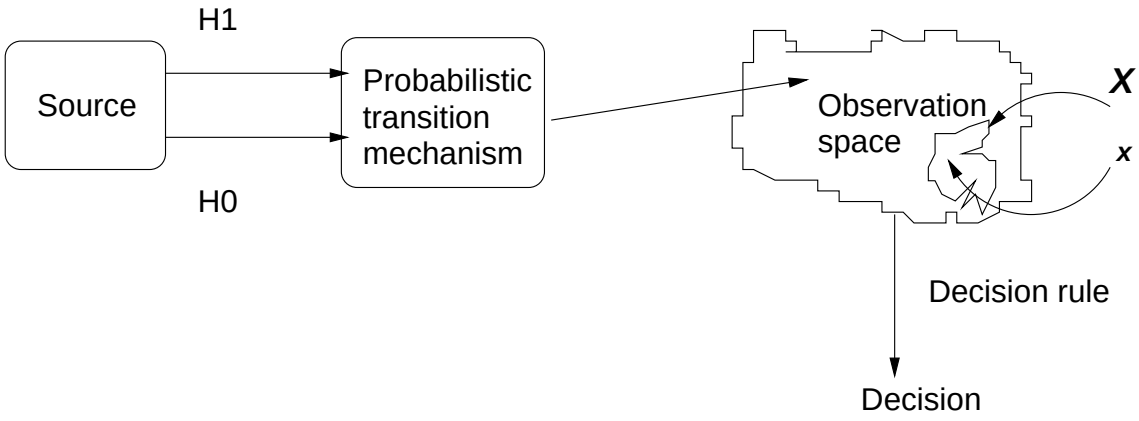

Figure 1: Classical detection theory model

by statisticians for some time, we have termed it a classical detection model. Based on an observation of the output of the source as transmitted through the probabilistic transition mechanism, we arrive at a decision. Our decision is based on a decision rule; for example, is or is not x in X, where X is the region in the observation space that defines the set of observations that we believe to be indicative of *H0* (or *H1*) (see figure 1). We then make a decision as to whether the source sent *H0* or *H1* based on the outcome of the comparison of x and X.

Note that the source and signal model *H0* and *H1* could represent any of a number of interesting problems, and not only the case of transmitting a one or a zero. For example, *H1* could represent the presence of a disease (and conversely *H0* its absence), and the observation space could be any number of measurable physiological parameters such as blood count. The decision would then be one of 'sick' or 'healthy.' In our case it would be natural to assign the symbol *H1* to some form of intrusive activity, and *H0* to its absence.

Our problem is then one of deciding the nature of our probabilistic transition mechanism. We must choose what data should be part of our observation space, and on this basis derive a decision rule that maximises the detection rate and minimises the false alarm rate, or settle for some desirable combination of the two.

When deciding on the decision rule the *Bayes criterion* is a useful measurement of success [Tre68, pp. 24]. In order to conduct a Bayes test, we must first know the a priori probabilities of the source output (see [Axe99] for further discussion). Let us call these P_0 and P_1 for the probability of the source sending a zero or a one respectively. Second, we assign a cost to each of the four possible courses of action. These costs are named C_{00}, C_{10}, C_{11}, and C_{01}, where the first subscript indicates the output from our decision rule—what we though had been sent—and the second what was actually sent. Each decision or experiment then incurs a cost, in as much as we can assign a cost or value to the different outcomes. For example, in the intrusion detection context, the detection of a particular intrusion could potentially save us an amount that can be deduced from the potential cost of the losses if the intrusion had gone undetected. We aim to design our decision rule so that the *average* cost will be minimised. The expected value—R for *risk*—of the cost is then [Tre68, p. 9]:

$$\begin{aligned} R = {} & C_{00}P_0P(\text{say } H0|H0 \text{ is true}) \\ & + C_{10}P_0P(\text{say } H1|H0 \text{ is true}) \\ & + C_{11}P_1P(\text{say } H1|H1 \text{ is true}) \\ & + C_{01}P_1P(\text{say } H0|H1 \text{ is true}) \end{aligned} \tag{1}$$

It is natural to assume that $C_{10} > C_{00}$ and $C_{01} > C_{11}$, in other words the cost associated with an incorrect decision or misjudgement is higher than that of a correct decision. Given knowledge of the a priori possibilities and a choice of C parameter values, we can then easily construct a Bayes optimal detector.

Though figure 1 may lead one to believe that this is a multidimensional problem, it can be shown [Tre68, p. 29] that a *sufficient statistic* can always be found whereby a coordinate transform

from our original problem results in a new point that has the property that only one of its coordinates contains all the information necessary for making the detection decision. Figure 2 depicts such a case, where the only important parameter of the original multidimensional problem is named *L*.

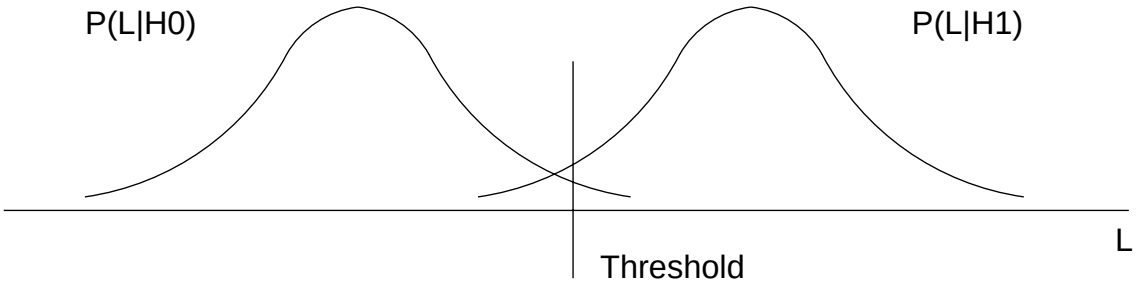

Figure 2: One dimensional detection model

It can furthermore be shown that the two main approaches to maximising the desirable properties of the detection—the Bayes or Neyman-Pearson criteria—amount to the same thing; the detector finds a likelihood ratio (which will be a function only of the sufficient statistic above) and then compares this ratio with a pre-set threshold. By varying the threshold in figure 2, it can be seen that the detection ratio (where we correctly say *H1*) and the false alarm rate (where we incorrectly say *H1*) will vary in a predictable manner. Hence, if we have complete knowledge of the probability densities of *H0* and *H1* we can construct an optimal detector, or at least calculate the properties of such a detector. We will later apply this theory to explain anomaly and signature detection.

Three recurring problems in detection and estimation theory are, in increasing order of difficulty:

Known signals in noise The simplest of the problems to be addressed. It arises in situations where we know what signal has (or has not) been transmitted. The problem is then one of filtering out the signal from the background noise. An example would be a simple Morse code radio transmitter, where we know that someone is intermittently transmitting a carrier wave of a certain frequency.

Signals with unknown parameters in noise Here the problem is more difficult, since we do not know the parameters (such as frequency, phase, amplitude etc.) of the transmitted signal. Such problems arise for instance in radar signal processing, where a signal is reflected from a target. The reflected signal from even the simplest of targets is at least phase shifted and attenuated compared with the original signal that was transmitted from the radar antenna.

Random signals in noise Here there is no known signal component to detect. Instead the source consists of two random processes, and our problem is that of deducing from the received signal which of the two processes is the likely source of the signal. These problems arise in certain satellite applications, as well as in the Search For Extra Terrestrial Intelligence (SETI) project.

We will use three intrusion detection situations as examples, making reference to these three categories.

3 Application to the intrusion detection problem

This section is a discussion of the way in which the intrusion detection problem may be explained in light of the classical model described above.

3.1 Source

Starting with the *source*, ours is different from that of the ordinary radio transmitter because it is human in origin. Our source is a human computer user who issues commands to the computer system using any of a number of input devices. In the vast majority of cases, the user is benevolent and non-malicious, and he is engaged solely in non-intrusive activity. The user sends only *H0*, that is, non-intrusive activity. Even when the user is malicious, his activity will still mostly consist of benevolent activity. Some of his activity will however be malicious, that is, he will send *H1*. Note that *malicious* has to be interpreted liberally, and can arise from a number of different types of activities such as those described by the taxonomies in for example [LBMC94, LJ97]. Thus, for example, the use of a pre-packed exploit script is one such source of intrusive activity. A *masquerading*[1] intruder can be another source of intrusive activity. In this case the activity that he initiates differs from the activity that the proper user would have originated.

It should be noted that we have only treated the binary case here, differentiating between 'normal' behaviour and one type of intrusion. In reality there are many different types of intrusions, and different detectors are needed to detect them. Thus the problem is really a multi-valued problem, that is, in an operational context we must differentiate between *H0* and *H1, H2, H3* . . . ,where *H1–Hn* are different types of intrusions. To be able to discriminate between these different types of intrusions, some statistical difference between a parameter in the *H0* and *H1* situation must be observable. This is simple, almost trivial, in some cases, but difficult in others where the observed behaviour is similar to benevolent behaviour. Knowledge, even if incomplete, of the statistical properties of the 'signals' that are sent is crucial to make the correct detection decision.

It should be noted that such classifications of computer security violations that exist [LBMC94, NP89, LJ97] are not directed at intrusion detection, and on closer study appear to be formulated on too high a level of representation to be directly applicable to the problem in hand. We know of only one study that links the classification of different computer security violations to the problem of detection, in this case the problem of what traces are necessary to detect intrusions after the fact [ALGJ98].

3.2 Probabilistic transition mechanism

In order to detect intrusive behaviour we have first to observe it. In a computer system context it is rare to have the luxury of observing user behaviour directly, looking over the user's shoulder while he provides a running commentary on what he is doing and intends to do. Instead we have to observe the user by some other means, often by some sort of security logging mechanism, although it is also possible by observing the network traffic emanating from the user. Other more direct means have also been proposed, such as monitoring the user's keystrokes.

In the usual application of detection theory, the probabilistic transition mechanism, or 'channel', often adds noise of varying magnitude to the signal. This noise can be modelled and incorporated into the overall model of the transmission system. The same applies to the intrusion detection case, although our 'noise' is of a different nature and does not in general arise from nature, as described by physics. In our case we observe the subject by some (imperfect) means where several sources of noise can be identified. One such source is where other users' behaviour is mixed with that of the user under study, and it is difficult to identify the signal we are interested in.

If, for example, our user proves to be malicious, and sends TCP-syn packets from a PC connected to a network of PCs to a target host, intended to execute a SYN-flooding denial-of-service attack on that host. Since the source host is on a network of PCs—the operating systems of which are known to suffer from flaws that make them prone to sending packet storms that look like SYN-flooding attacks to the uninitiated—it may be difficult to detect the malicious user. This is because he operates from under the cover of the noise added by the poorly implemented TCP/IP stacks of the computers on the same source network as he is. It can thus be much more difficult to build a model of our 'channel' when the noise arises as a result of a purely physical process.

[1] A *masquerader* is an intruder that operates under false identity. The term was first used by Anderson in [And80]. See section 5.3 for a more detailed explanation.

If source models are lacking in intrusion detection, then study of the transmission observation mechanisms is almost non-existent. As mentioned, we know of only one study of intrusions [ALGJ98] that takes this perspective.

3.3 Observation space

Given that the action has taken place, and that it has been 'transmitted' through the logging system/channel, we can make observations. The set of possible observations, given a particular source and channel model, makes up our *observation space*.

As said earlier, some results suggest that we can always make some sort of coordinate transformation that transforms all available information into one coordinate in the observation space. Thus in every detection situation we need to find this transformation.

In most cases the computer security audit data we are presented with will be discrete in nature, not continuous. This is different from the common case in detection theory where the signals are most often continuous in nature. In our case a record from a host-based security log will contain information such as commands or system calls that were executed, who initiated them, any arguments such as files read, written to, or executed, what permissions were utilised to execute the operation, and whether it succeeded or not. In the case of network data we will typically not have such high quality information since the data may not contain all security relevant information; for example, we will not know exactly how the attacked system will respond to the data that it is sent, or whether the requested operation succeeded or not [PN98]. As noted in section 3.2, the question of what data to log in order to detect intrusions of varying kinds is still open. We also know little of the way different intrusions manifest themselves when logged by different means.

Once again the literature is hardly extensive, although for example[ALGJ98, HL93, LB98] have touched on some of the issues presented in this section, albeit from different angles.

3.4 Decision rule

Having made the coordinate transformation in the previous step we then need to decide on a threshold to distinguish between *H0* and *H1*. Of course in the case of a discrete detector, the threshold is often a simple *yes* or *no*.

Thus our hope when we apply anomaly detection (see section 4.1) is that all that is not normal behaviour for the source in question—that cannot be construed as *H0*—is some sort of intrusive behaviour. The question is thus to what degree abnormal equates to intrusive. This is perhaps most likely in the case of a *masquerader* who one may presume is not trained to emulate the user whose identity he has assumed. There are some studies that suggest that different users indeed display sufficiently different behaviour for them to be told apart [LB98]. Whether anomaly detection can detect other types of intrusion is more uncertain, although there are some preliminary indications that it may [ALJ+95].

It is interesting to note that it is only recently the area of detection principle has seen much interest. Most of the earlier research presents an intrusion detection prototype without a discussion of how that detector would perform. The researchers seldom present any empirical data or theoretical calculations of the required or expected performance of their prototype—either in terms of detection accuracy, false alarm rate, or runtime performance—instead limiting their argument to purely architectural issues of the intrusion detection system.

4 Existing approaches to intrusion detection

For a complete survey of existing approaches to intrusion detection see [Axe00]. Here we will only outline the two major methods of intrusion detection: *anomaly detection* and *signature detection*. These have been with us since the inception of the field. In short, *anomaly detection* can be defined as looking for the unexpected—that which is unusual is suspect—at which point the alarm should be raised. *Signature detection*, on the other hand, relies on the explicit codifying of 'illegal' behaviour, and when traces of such behaviour is found the alarm is raised.

4.1 Anomaly detection

Taking the basic outline of detection and estimation theory laid out in section 2, we can elaborate upon it in describing these methods. In contrast to the model in figure 2, where we have knowledge of both *H0* and *H1*, here we operate without any knowledge of *H1*.

Thus we choose a region in our observation space—X in figure 1. To do so, we must transform the observed, normal behaviour in such a manner that it makes sense in our observation space context. The region X will contain the transformed normal behaviour, and typically also behaviour that is 'close' to it, in such a way as to provide some leeway in the decision, trading off some of the detection rate to lower the false alarm rate. The detector proper then flags all occurrences of x in X as no alarm, and all occurrences of x *not* in X as an alarm. Note that X may be a disjoint region in the observation space.

4.2 Signature detection

The signature detector detects evidence of intrusive activity irrespective of the model of the background traffic; these detectors have to be able to operate no matter what the background traffic, looking instead for patterns or signals that are thought by the designers to stand out against any possible background traffic.

Thus we choose a region in our observation space, as described in section 4.1, but in this instance we are only interested in known intrusive behaviour. Thus X will here only encompass observations that we believe stem from intrusive behaviour plus the same leeway as before, in this case trading off some of the false alarm rate to gain a greater detection rate in the face of 'modified' attacks (see section 5.2 for a description of such modifications). During detector operation we flag all occurrences of x in X as an alarm, and all other cases as no alarm. X here may also consist of several disjoint regions, of course.

4.3 Comparison with Bayesian detectors

It is an open question to what degree detectors in these classes can be made to, or are, approximate Bayesian detectors. In the case of non-parametric intrusion detectors— detectors where we cannot trade off detection rate for false alarm rate by varying some parameter of the detector—merely studying the receiver operating characteristics (ROC) curve cannot give us any clue as to the similarity to a Bayesian detector. This is because the ROC curve in this case only contains one point, and it is impossible to ascertain the degree to which the resulting curve follows the optimal Bayesian detector. (See [Axe99], for a brief introduction to ROC curves, [Tre68] for a thorough one).

5 Examples

To see how intrusion detection systems fit with the classical theory we will give a few examples of intrusion detection scenarios and explain them in light of the theory. A few typical examples, chosen to illustrate the model above, will be presented in order of increasing difficulty and sophistication.

The examples are drawn primarily from the UNIX environment, an environment with which we presume the reader is familiar, and from its basic operational and security behaviour. Numbers in parenthesis after program names, library procedures, and system calls refer to the relevant sections of the UNIX manual.

5.1 Setuid shell scripts

It has long been known that setUID shell scripts pose a serious security problem in UNIX systems. A number of different weaknesses present themselves, of which we are going to present one of the more straightforward here, since it is simple and illustrates our point well. It should be noted

that this flaw is of historical significance only, as current UNIX systems guard against it [Ste92]. A more detailed explanation of this flaw is presented in [ALGJ98].

The intrusion occurs when the potential intruder finds a setUID shell script on the system. He then proceeds to make a soft link to the script file. The soft link is named `-i`. If the script is invoked via the link by the attacker issuing the command `-i`, then he will be greeted with the shell prompt of a setUID shell. This attack is dependent on the shell script being run by the Bourne shell.

The specific flaw here stems from when the execution of shell scripts was moved from the command interpreter into the kernel; the *`exec(2)`* system call first opens the executable file, sees that it begins with the magic number '#!', reads the next argument as a command interpreter to call, and calls that command interpreter with the file name of the shell script as the first parameter. The idea is that the command interpreter will then open the file and read commands from it. The problem here is that when the Bourne shell is called with `-i` as the first parameter, it does not interpret this argument as a filename to open, but instead interprets it as a switch—or option—to start in interactive mode. Thus instead of executing the shell script, the attacker is rewarded with a setUID shell prompt. This then enables him to execute arbitrary commands with system administrator privileges.

In order to detect this attack with an ordinary, signature-based intrusion detection system, we might observe that few legitimate programs would be invoked by the command `-i`, and decide to search for all command invocations of this type. We would then have constructed an implicit *H0* model, i.e. that benign, *H0* behaviour will never consist of a command with the name `-i`. The *H1* model is quite clear: any program that is invoked with the command `-i` is an attack, even though this of course is not a problem in itself. Our logging mechanism could then focus on program invocation and for example log all *`exec(2)`* calls (see [ALGJ98] for an evaluation of this logging method).

This case is sufficiently simple for it to be likely that all security professionals would agree with the implicit *H0*, *H1* and transition mechanism models discussed above. However, as we will see in the next example, the moment the complexity of the intrusion increases, variation in the exploitation of flaws by an adept attacker becomes a factor, and value of stricter models begins to show.

5.2 Buffer overrun

Buffer overruns are a common form of vulnerability in current systems. They are often serious because when they are exploited the machine under attack often falls completely into the hands of the attacker. He gains administrative privileges, and typically all security of the computer system is then lost. A common flaw in 'C' programs is the reading of user input into a buffer—finite in length—without proper bounds checking. This is a very common occurrence because of language and library features that input or copy data without checking for overflow. Common culprits in the 'C' library are *`gets(3)`*, *`strcpy(3)`* and *`sprintf(3)`*. Even though these routines have been superseded with newer versions that check the length of the input data on programmer request, many programs have not been updated, and simple errors in the application of the new routines can cause serious problems. Given the culture within the 'C' world that favours performance before correctness, the problem is further aggravated.

In order to exploit a buffer overrun the attacker typically copies a machine code snippet into the buffer that is being overrun, and then continues to overwrite the return address of the currently executing subroutine with the address of the 'poisoned' code just inserted. When the current subroutine then returns, it will not jump back to the statement past the one that called it, but rather to the beginning of the attack code which it then executes. This code typically starts the execution of a command interpreter which often runs with system administrator privileges, and it is thus a trivial matter to execute general commands with system administrator privileges from this command interpreter. For detailed descriptions of this particular technique, see [One96].

When we wish to detect such violations, we start from the beginning and note a few characteristics of the particular exploit we wish to detect; in other words, we study the *H1* characteristics of the source. If we follow the reasoning in [LP99] we could note that the exploit code is often transferred to the buffer as a string that should be text, but since it is binary in origin, it instead

contains a lot of control characters. Further, we could conclude that it often involves an *exec(2)* call, and is longer than a certain length. This leads to the idea of logging *exec(2)* calls, i.e limiting our observation space to the *exec(2)* calls. Our detection rule is then simply to sound the alarm whenever the argument to the call exceeds a certain length and contains control characters, exactly the approach taken in [LP99].

Studying this approach from the perspective of our model a few shortcomings become clear. We operate without any strong model of the *H0* behaviour of the source, that is, we do not know the likelihood of benign traffic triggering our detection mechanism, causing a false alarm. Indeed, this has led some researchers to try a detection rule with only the length of the *exec(2)* arguments as the parameter, but in practice this provoked false alarms from benign invocation of *sendmail(8)*. Thus when operating without a good model of the benign behaviour of the source, we must write *robust* signatures that are not likely trigger false alarms, even when faced with the most general behaviour of the target system. Any benign behaviour, now or in the future, should pass through the system without raising an alarm.

This robustness must be accomplished while at the same time keeping the detection rate high, something we foresee being difficult. Indeed, even in the example given the attack can easily be generalised to render the detection completely ineffective (see [CWP+00] for an exposition of the problem of the variability of buffer overrun attacks). For example, we could easily do away with the exploit code altogether and instead choose to exploit code that is already in place in the running program. That leaves only the code pointer to change, and that would slip completely through the detection mechanism described above. Or we could perhaps find a data dependent flaw to exploit, deciding not to touch any function pointers at all. Of course if the attacker was aware that intrusion detection system 'X' was running on the protected system, and knew how that system operated, avoidance techniques like those described would be more the norm than the exception. The way intrusion detection systems that ignore the *H0* behaviour of the source will manage the robustness problem—having a high enough detection rate, especially in the face of such varied attacks, while still keeping the false alarm rate at bay—is still an open question.

5.3 Masquerading

A class of intrusions that are difficult to identify but have far-reaching effects are the *masqueraders*, or intruders who operate under stolen identity. Typically they have 'stolen' the password of a legitimate user, and log into the system assuming the identity of that user. This is of course a difficult situation to detect, but not impossible. To date, the proposed mechanisms have taken the path of constructing a profile of normal behaviour for the authorised user, and then detecting deviations from this profile. When we observe the user is behaving abnormally, we in fact take this as an indication not that it is the legitimate user acting abnormally, but rather that another, illegitimate user is acting in the guise of the authorised user.

This is not an uncommon strategy when doing manual detection and intervention. Often the first clue that something is amiss is the observation that the system is not behaving as it usually does [SSHW88]; for example, the system takes a longer time to complete a calculation. Upon further investigation we find that a legitimate user is logged in at an unusual time, say a hypothetical clerk in the financial department is logged on at four in the morning, and that the logs show that she has just used the 'C' compiler. If we know that the clerk in question is not a programmer, then the situation is starting to look questionable. Indeed the earliest intrusion detection systems attempted to automate this very process, with varying success [SSHW88]. The approach of constructing profiles of normal behaviour in particular for users of systems, however, has stood the test of time.

Of course, even the automated construction of such profiles is both error prone and time consuming. The detection problem is also one of adjusting the sensitivity of the system so that it does not flag small deviations from the norm as intrusions. Other problems include users who learn over time and thus change their behaviour, users who work in different capacities—they might spend the first half of the day programming and the second writing reports—and malicious users who deliberately teach the intrusion detection system their illegitimate behaviour so that it perceives it as 'normal', and hence not worthy of an alarm.

Translating this scenario into the model presented above is fairly straightforward. First we

have a source that is known and that sends only *H0*, that is, we have the original, legitimate user who only behaves 'in character.' We then observe the user's actions via the transition mechanism, typically some sort of detailed log of the commands the user has issued to the system, but other more exotic approaches have been proposed, such as measuring the time between successive keystrokes to build a profile of the user's typing behaviour. We then try to make the ordinary transformations to make the regions of interest in our observation space more tractable. In our case that will often consist of calculating descriptive statistics of things such as command usage frequencies to form a profile of the user's behaviour. We then decide on a detection rule, which in this case amounts to identifying a subset of the observed behaviour as being characteristic, in order to maximise the number of true alarms and minimise the number of false alarms.

The problem with the standard approach is highlighted by this detection model, namely that we do not have any idea of the characteristics of *H1*. Instead we try to 'detect' a completely unknown signal solely by forming an opinion on the characteristics of *H0*, and assuming that all we observe that does not fit our *H0* model well enough is by default *H1* behaviour. What we would like to detect is of course direct *H1* behaviour, there are no guarantees that whatever is *not* normal—or rather uncommon—*H0* behaviour is in fact *H1* behaviour. The naive approach of collecting statistics about all *possible* users is not practical, and we will instead have to find at least some information on the characteristics of general *H1* behaviour, or we will have no idea of how our detection rule would fare, since we will have no inkling as to how much the observed regions in the observation space would typically overlap, if at all.

There have been very little study of this area, although results by Lane and Brodley [LB98] seem to indicate that users differ sufficiently for this approach to be viable. Lane and Brodley also study the transition mechanism problem, and conclude that in this context it is mainly a problem of feature selection.

6 Discussion

The dichotomy between *anomaly detection* and *signature detection* that is present in the intrusion detection field, vanishes (or is at least weakened) when we study the problem from the perspective of classical detection theory. If we wish to classify our source behaviour correctly as either *H0* or *H1*, knowledge of both distributions of behaviour will help us greatly when making the intrusion detection decision. Interestingly, only one research prototype takes this view [Lee99, Axe00]; all others are firmly entrenched in either the *H0* or *H1* camp. It may be that further study of this class of detectors will yield more accurate detectors, especially in the face of attackers who try to modify their behaviour to escape detection. A detector that operates with a strong source model, taking both *H0* and *H1* behaviour into account, will most probably be better able to qualify its decisions by stating strongly that this behaviour is not only known to occur in relation to certain intrusions, and further is not a known benign or common occurrence in the supervised system. Such detectors would of course be of great help to SSOs in their daily work.

Furthermore, the three examples in sections 5.1, 5.2, and 5.3 fit neatly into the three categories detailed in section 2:

- First we have the class 'known signals in noise' that equates to the 'setUID shell script' attack. It is simple, straightforward, and 'static', meaning it is not amenable to simple variation by the attacker to avoid detection.
- Second we have the 'buffer overrun' attack that fits into the 'known signals with unknown parameters in noise' class. Here the general outline of the attack is known, and indeed particular details have lured some into over-specifying patterns for the detection of such attacks. However, since the specific attack can easily be modified along several, seemingly different paths, it would be difficult to detect if one were to treat the problem as one of the previous class. We need to view the problem as fundamentally more difficult, where we know the general structure of the signal but not the specific parameters in this particular instance.
- Third we have 'masquerading' , analogous to the 'random signals in noise' class. Here the problem is of several orders of magnitude greater. We do not even know the general outline

of the signal we are trying to detect, and the intrusion detection community has hitherto relied solely on a negative formulation of the problem: all that is not known to be non intrusive is classified as intrusive.

However, fundamental questions remain that are not addressed by the simple binary model presented here. Having differentiated between different types of intrusion, we face the question of the relative merit of detecting those types of intrusion with an automatic intrusion detection system. Not only are certain types of intrusion more serious in their potential impact from a system owner's perspective, but the same owner has a varying capacity to correct security flaws in different situations, depending for instance on whether the flaw is one of design, installation, or use. Thus, when used pro-actively the system owner is probably most interested in defending against attacks that have a high potential loss associated with them, that are difficult to defend against by other means, and for which there is little possibility of fixing, at least beforehand. When used in a post-mortem fashion, analysing old data for evidence of intrusive activity, the demands are likely to be slightly different. Here it is likely we would be interested in trying to find evidence of attacks that we did not know about when the system was operating real time on the same security data.

Thus we believe we have shown that a closer look at classical detection and estimation theory field would prove fruitful.

7 Conclusions

It is interesting to note the relative absence in the field of intrusion detection of references to strict classical detection and estimation theory. This paper attempts a preliminary explanation of the problem of computer security intrusion detection from a classical detection theory viewpoint. There is a sufficiently good fit between introductory models of classical detection and estimation theory and the intrusion detection problem to merit further study. The comparison also points to new and exciting areas of research in intrusion detection, where intrusion detectors, contrary to what has previously been the case, could operate with combined models of benign and malicious behaviour.

References

[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.

[And80] James P. Anderson. Computer security threat monitoring and surveillance. Technical Report Contract 79F26400, James P. Anderson Co., Box 42, Fort Washington, PA, 19034, USA, 26 February revised 15 April 1980.

[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.

[Axe00] Stefan Axelsson. Intrusion-detection systems: A taxonomy and survey. Technical Report 99–15, Department of Computer Engineering, Chalmers University of Technology, SE–412 96, Göteborg, Sweden, March 2000.

[CWP+00] Crispin Cowan, Perry Wagle, Calton Pu, Steve Beattie, and Jonathan Walpole. Buffer overflows: Attacks and defenses for the vulnerability of the decade. In *DARPA Information Survivability Conference & Exposition (DISCEX 00)*, pages 119–129, Hilton Head, South Carolina, USA, 25–27 January 2000. DARPA, IEEE Computer Society, IEEE Computer Society Customer Service Center, 10662 Los Vaqueros Circle, P.O. Box 3014, Los Alamitos, CA 90720-1314, USA.

[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.

[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.

[LBMC94] Carl E Landwehr, Alan R Bull, John P McDermott, and William S Choi. A taxonomy of computer program security flaws. *ACM Computing Surveys*, 26(3):211–254, September 1994.

[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.

[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.

[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.

[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.

[One96] Aleph One. Smashing the stack for fun and profit. *Phrack*, 49–14, November 1996. Avaliable from http://www.phrack.com.

[PN98] Thomas H. Ptacek and Timothy N. Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical report, Secure Networks Inc., January 1998. Available via the web: http://www.secnet.com/papers/IDS.PDF at the time of writing.

[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.

[Ste92] W Richard Stevens. *Advanced Programming in the UNIX Environment*. Addison-Wesley, 1992.

[Tre68] Harry L. Van Trees. *Detection, Estimation, and Modulation Theory, Part I, Detection, Estimation, and Linear Modulation Theory*. John Wiley and Sons, Inc., 1968.