Inf Syst Front manuscript No.
(will be inserted by the editor)

Information Systems Frontiers, Volume 15, March 2013, pp. 17–34
DOI:10.1007/s10796-010-9268-7

Knowing Who to Watch: Accumulating Evidence of Subtle Attacks

Howard Chivers · John A. Clark · Philip Nobles · Siraj A. Shaikh · Hao Chen

the date of receipt and acceptance should be inserted later

Abstract Insider attacks are often subtle and slow, posing the problem of integrating a large volume of event data from multiple sources over a long period. This paper proposes a scalable solution to combining evidence from multiple sources, by maintaining long-term estimates that individuals or nodes are subverted, rather than retaining event data for post-facto analysis. These estimates are then used as triggers for more detailed investigation. We identify essential attributes of event data, allowing the use of a wide range of indicators, and show how to apply Bayesian statistics to maintain incremental estimates without global updating. The paper provides a theoretical account of the process, a worked example, and a discussion of its practical implications. The work is couched in terms of networks and identifying subverted nodes, but is equally applicable in a social and behavioral context and the identification of suspect individuals.

1 Introduction

Insider attacks pose a particular threat because of the knowledge, access, and authority of their perpetrators (Randazzo et al, 2004). Such attacks often involve violations of physical or operational security, or the misuse of authority; they may also involve electronic attacks, in which case the 'electronic insider' is as big a threat as a person. It may be safer for a sophisticated external attacker to subvert an electronic system, often via social engineering, than directly subvert an employee. Such attackers may use technical means to camouflage an attack, such as indirection or address spoofing (spo, 1998); however, their most potent weapon in avoiding detection is patience – the world's largest credit card fraud was achieved with a subverted internal system that avoided discovery for over 17 months (Goodin, 2007).

Subtle attackers are unlikely to launch large-scale scans or use known exploits; they will seek to avoid any action that can be immediately identified as an attack. However, they are likely to cause minor security events: an attacker may test known passwords, probe for services, or test new exploits, expecting to hide within the background of user errors, mistakes and other 'noise'. The problem of detecting such an attacker is therefore one of accumulating relatively weak evidence over a long period. This issue is one of the 'grand challenges' of the internal attacker problem: "to combine events from one or more sensors, possibly of various types" while "reduce[ing] data without adversely impacting detection" (Brackney and Anderson, 2004). This paper provides a solution to this critical problem.

The work presented here is couched in terms of networks and systems, and the identification of a subverted node, which is part of a system that is used by a corrupt insider, or is acting as an electronic insider for some other party. However, the approach to characterizing and combining diverse sources of weak evidence is equally applicable to other problems in the insider space, such as identifying criminal or espionage threats from behavioral indicators.

This paper provides a process for combining evidence from various sources based on the application of Bayesian statistics, identifies attributes that must be available to allow the combination of evidence from different types of sensor, and demonstrates the effectiveness of this approach with a simulated slow-attack on a network.

This paper presents the the results of substantially more research than its workshop predecessor (Chivers et al, 2009).

Howard Chivers, Philip Nobles, Siraj A. Shaikh
Department of Information Systems, Cranfield University, Shrivenham, UK
E-mail: h.chivers/p.nobles/s.shaikh@cranfield.ac.uk

John A. Clark, Hao Chen
Department of Computer Science, University of York, York, UK
E-mail: jac/chenhao@cs.york.ac.uk

Although the principles and aims are the same, the hypothesis on which the updating algorithm is based has been changed, resulting in a improved updating factor, which is effective at resolving some marginal discrimination observed in the previous results. This paper also includes a significantly larger realistic simulation, explicit results on the limits of evidential accumulation, and a discussion on normalization that justifies the stance that it is not necessary to update scores at every node following each event.

The paper is organized as follows: Section 2 provides an overview of the proposed approach, section 3 describes related work, and the evidential accumulation process is developed and described in section 4. After a brief explanation of the simulation approach in section 5, section 6 shows that the proposed process is well behaved in simple cases, and that it gives the same estimate of behaviour, regardless of the size group an individual is associated with; section 7 then explores the effective limits to updating evidence. Section 8 simulates a challenging insider detection problem, contrasts the effectiveness of the evidence accumulation process with a common, but naive, alternative approach, and shows how the results vary with increasing uncertainty of identification of nodes that originate events. Section 9 discusses results and open issues, and the paper is concluded in section 10.

2 Overview

Consider how a human investigator might approach the problem of accumulating evidence in the network of Figure 1. The network consists of nodes *(A...J)* with interconnectivity as shown. Two minor security events are detected *E1,* and *E2*; they may originate from an operating system alert, intrusion detection system, or other form of event detector.

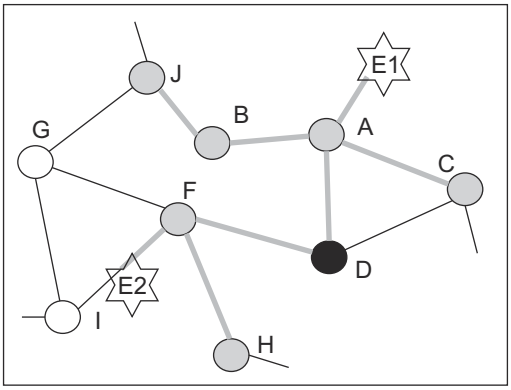

Fig. 1 Intersecting Evidence

Given information about event *E1* and the traffic in the network at the time, the investigator may determine that the nodes most likely to have originated the event are *J, B, A, C or D*. Similarly, when *E2* occurs, at a much later date, the possible originating nodes are *D, F* and *H*. Intersecting these observations suggests node *D* as a common factor, and this may be sufficient to trigger intensive monitoring to determine if it is behaving maliciously.

The data used to identify these security events and their possible sources is necessarily transient; it may not be possible to record sufficient traffic to allow this analysis retrospectively. However, it is initially sufficient to just identify nodes that score differently; in the long, slow, game, it is only necessary to 'tip off' a further investigation by identifying one or more nodes whose behaviour may be unusual. It is not essential to record the events, the traffic from which they were identified, or even the graphs that identify possible sources, provided it is possible to somehow accumulate a 'score' for each node in the system.

This approach solves one of the critical issues in identifying slow attacks: how to maintain long-term state. Systems that try to model the behaviour of individuals, systems or protocols, are forced to retain large amounts of data, which limits their scalability. In the approach described here, the state size is a small multiple of the number of nodes in the network; this state is readily distributed, and its storage is feasible, even for organizations with global networks.

The 'score' that we propose for each node is the probability that the node is subverted, based on the application of Bayesian statistics. This naturally allows incremental updating, and translation of the problem frame from events, which are related to behaviour, to individual attackers. Simpler schemes, such as the event counting used to introduce this section, can be shown to be inferior, as demonstrated in section 8.

In summary, we propose that to identify subtle or inside attackers:

- The primary objective is to identify nodes for further investigation.
- Long-term state is restricted to an incremental estimate of the probability that each node is an attacker.
- Node estimates are updated following every security event, taking account of transient network information that may be available at the time of the event.

This process is complementary to conventional intrusion detection using signatures or heuristics. There is no need to gradually accumulate evidence if the attack is evident. For example, a high level of network activity due to a worm or virus provides compelling evidence of an attack, and in this type of case the secondary investigation is concerned with incident management, rather than confirmation.

Section 4 describes how node scores are calculated and maintained, following a brief summary of related work.

3 Related Work

The use of a tiered approach to insider threat detection, detection followed by a more detailed forensic investigation, is proposed by Bradford et al (2004). Users are profiled according to their function, and deviation from normal behaviour triggers more intensive data collection. Sequential hypothesis testing is proposed to determine whether a process is anomalous and more intensive data collection should be initiated. However, the authors do not show an implementation of their approach, and remark that it could not be carried out for "every user regardless", but itself requires a "triggering process".

The problem is the volume of data that must be maintained, and this is also a issue with datamining approaches, which are often proposed as an adjunct to intrusion detection or audit. Research proposals to alleviate the scalability issue include improving the quality of the raw data, by discovering better behavioral indicators (Nguyen et al, 2003) or classifying input features (Chebrolua et al, 2004), the latter using a Bayesian classifier. An alternative approach by Staniford et al (2002) is to selectively retain anomalous network data, with the aim of identifying slow network scans. Anomalous packets are identified based on heuristics developed from real scans. Other approaches include statistical filtering, primarily to reduce false alarm rates and support visualization (Colombe and Stephens, 2004). In essence, however, all these approaches require the storage of large volumes of event data for later analysis, and the authors themselves often identify scalability as a problem (Nguyen et al, 2003).

Aggregation as a means of detecting slow or stealthy attacks has been proposed by Heberlein (2002). His assumption is that slow attacks are still systematic, and the attacker will eventually repeat the attack many times, possibly against different targets. Alerts are classified, accumulated, and displayed on a visualization grid, and any persistent activity which raises alerts of the same type over a long period, can be identified. Although similarly motivated, our work differs by accumulating evidence of attackers, not of incidents, removing the restriction that attackers need to repeat similar attacks. Heberlein's algorithm is also a counting process, which we show to be inferior to statistical reasoning.

Other work directed toward the insider problem is focussed on characterising an attacker's behaviour. The security indicators ('events') used may range from an individual's buying and travel preferences, to electronic alerts. For example, Buford et al (2008) propose a comprehensive framework of 'observables' that are used to build a model of individuals' behaviour via graph theory. Eberle and Holder (2009) develop graphs of behavioral events, such as phone calls, to identify sub-graphs of normal behaviour, which are used to search for similar but anomalous occurrences. These approaches offer the advantage of modeling the potential attacker, and providing interesting insights into observable behaviour; however, their application may be limited by the computational cost of graph matching over large datasets, as well as by data scalability.

Most of the work described above is still formative; network intrusion detection, however, is established in the literature and supported by both open and propriety products (Bace and Mell, 2001). An intrusion detection system (IDS) uses a behavioral model of a system or protocol and detects anomalous events by either recognizing predefined signatures, or by heuristics. Both approaches have strengths and weaknesses, but despite the usefulness of IDSs in practice, they are hampered by a lack of scalability, and tend to generate large numbers of false positive alerts (Bace and Mell, 2001). From the perspective of this paper, IDSs are an effective way of generating events which may indicate an attack, but are unable to maintain sufficient state to identify slow attacks.

An IDS is not the only possible source of security events; for example, the behavioral events referenced above, operating system audit trails, and even Honeypots (Spitzner, 2003), which are security traps with no operational functionality, are all possible event sources.

In summary, the challenge of integrating information from many sources in a manageable and scalable fashion, in order to identify patient internal attackers, is still an important open question (Brackney and Anderson, 2004).

4 Accumulating Evidence

This section develops the detailed theory necessary to achieve the method outlined in section 2: to collapse the problem of attacker identification to updating a single score for each network node, or user. The section first outlines the evidential scenario, and the attributes required to characterize security events. Standard Bayesian updating is summarized, followed by the development of the process for updating evidence of insider attacks. Finally, the practical issue of relating this process to real security events is discussed.

Definitions

Node: This paper uses network terminology, without loss of generality to broader types of human or attack behavior. A node is a network component, such as a user's end system, a router, or a user. The equivalent in behaviour modeling or social networks is an individual.

Event: An event is an alert that indicates a possible security violation; it may be an anomalous phone call, a failed connection, or something more certain, such as a known electronic exploit.

The evidential scenario is presented in Figure 2. Node a is a network node, and x is an event which is detected some-

where in the network; there is some evidence that identifies the nodes that may have originated the event.

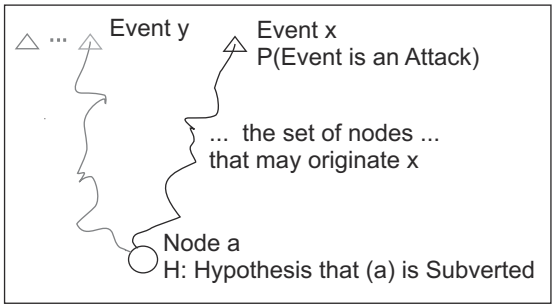

Fig. 2 The Evidential Scenario

Event *x* may indicate an attack. Some security events are almost certainly attacks; however, there are many more that may be user mistakes, backscatter, or other forms of network 'noise'. For example, an attempt to connect to a non-existent webserver is often a simple mistake, but could also be an attack probe.

In addition to uncertainty about the extent that an event is an attack, there may also be uncertainty about the origin of the event. For example, the attacker may be able to spoof its network address, or the event may only be traceable to a subnetwork; in a social context only a proportion of the individuals associated with a particular event (behavioral indicator) may be identifiable. In order to accumulate evidence from a wide range of different sources, events must be characterized by uniform parameters that describe these various attributes. We propose that security events can be characterized by three parameters:

- *P(Attack):* the probability that an particular event is caused by an intentional attack. For an event generated by network intrusion sensors this is the ratio of true positive alerts to all alerts, which is a standard figure of merit. For behavioral indicators (e.g. presence at a street demonstration) and other system alerts (e.g. failed logins) it is necessary to estimate the value based on likely event frequencies. In a similar way to estimating risk likelihoods, it may be sufficient to quantify these frequencies to an accuracy of an order of magnitude.
- *The Causal Node Set:* the set of nodes or individuals that could have originated the event. In a network it may be possible to associate the event with a packet stream that originated from an identifiable subnetwork or node. This may be a static feature of the sensor's location, or it may be deduced from the data (the packet source address places it in a particular subnetwork), or from dynamic system information (e.g. current routing tables). In the case of behavioral information it is likely to be an identifiable set of individuals (e.g. visitors to an internet cafe during a specific period).
- *P(Causal):* the probability that the event originator is within the causal node set. It will not always be possible to identify with certainty the set of nodes or individuals that contain the event originator. For example, routing tables or traffic records may suggest that most packets came from a specific network, but a few came from elsewhere; the traceability of a packet to a particular subnetwork may depend on the correct functioning of routers and firewalls, which themselves have a finite possibility of being subverted. In the case of behavioral indicators it may be possible to know the number of people involved in an incident, but only positively identify a fraction of the individuals. All these factors suggest the need for a quality metric which describes the certainty that the actual event originator is in the causal set.

Given a sequence of events characterized by these parameters, we wish to investigate the hypothesis that a particular node is subverted, or acting as the agent of an attacker. We will first summarize the standard approach to Bayesian updating, then show how it can be applied in this case.

4.1 Bayesian Updating

Bayesian updating provides an estimate of the probability that hypothesis H is true, given an event, *x*.

$$P(H|x) = \frac{P(x|H) \cdot P(H)}{P(x)} \tag{1}$$

This theorem uses *P(x|H)*, the probability of event *(x)* given that the hypothesis is true, to update the initial ('prior') estimate of the probability that the hypothesis is true, *P(H)*. Simple updating of this type is often used in medical diagnosis; given knowledge of the probability of a symptom (the event) given a disease (the hypothesis), it provides a principled estimate of the likelihood of the disease given the symptom. It is essentially this change of reference frame – from symptom to cause – that is needed to identify internal attackers from their behaviour.

The denominator, *P(x)*, the probability of the event, is effectively a normalizing factor which ensures that the probabilities of all the possible hypotheses sum to unity. In many cases, including ours, it is difficult to estimate *P(x)*, and this is resolved by explicitly normalizing over the hypotheses; the problem of normalization is discussed further in section 4.2.3, below.

Assuming conditional independence (i.e. that the probability of an event is conditioned by the hypothesis, but not by other events that are observed, see section 4.3), the evidence from several events (e.g. x,y) is combined as follows:

$$P(H|x,y) = \frac{P(x|H) \cdot P(y|H) \cdot P(H)}{N} \quad (2)$$

Where N is a normalizing factor.

4.2 Combining Evidence from Security Events

The evidential scenario is described at the start of this section; in detail, we define:

S	The set of all nodes in the system.
$\#S$	The total number of nodes in the system.
$a,b...$	Particular network nodes. $a,b,... \in S$
H_a	The Hypothesis that we wish to update: that node (a) is the node which is subverted, or being used to mount an attack within the system.
$x,y...$	Particular events that may provide evidence of an attack.
$P(Attack_x)$	The probability that a particular event, x, originates from an intentioned attack.
C_x	The causal set of nodes that are likely to have originated event x.
$\#C_x$	The number of nodes in set C_x
$P(C_x)$	The probability that C_x includes the node that originated the event.

The parameters $P(Attack_x)$, the casual node set, C_x, and the likelihood that the originator is in this set, $P(C_x)$, were described in the introduction to this section; they are the three attributes necessary to characterize an event.

The hypothesis, H_a, assumes that only one node in the system is subverted; this provides improved discrimination and normalization over the alternative hypothesis that several nodes may be subverted. This is a technical issue and does not inhibit the practical use of the resulting scores to identify the (several) most likely attackers.

In order to carry out Bayesian updating as specified in equation 2, it is necessary to calculate the update factor $P(x|H_a)$, the prior probability $P(H_a)$, and when required, to normalize the result in such a way that the probabilities across all nodes sum to unity.

4.2.1 The prior probability

The prior probability is a function of the network node, and may be estimated in advance for the type of node, or if there is no basis for distinguishing nodes (see section 4.3, below), it can be set to *1/#S*.

4.2.2 The Bayesian update factor

The update factor $P(x|H_a)$ is the likelihood of the event, given the hypothesis H_a. Given that event x has been observed, and the hypothesis that a is the only attacker, then either:

- Event x is an attack, and it came from node a, or
- Event x is not an attack, and it may have originated from any node.

The probability that x is an attack is the parameter $P(Attack_x)$ that characterizes the event, as described above. The probability that the event is not an attack (i.e. it is a false positive) is therefore $(1 - P(Attack_x))$.

Each event is associated with a set of nodes, C_x that is expected to include the originator of the event. This set divides the population of nodes in the system into two; node a, which is the subject of the hypothesis, may be a member of C_x, or may fall outside that set. If a is in C_x then the probability that x originated from a is the probability that C_x includes the originating node, $P(C_x)$, divided by the number of nodes in the set, $\#C_x$. If a is not in C_x then the probability that x originated from a is $(1 - P(C_x))$, divided by the number of nodes outside C_x, $(\#S - \#C_x)$.

This allows us to calculate the probability that x is an attack, and it came from node a. As described above, we must add the possibility that x is not an attack to obtain the required update factor:

$if a \in C_x$:

$$P(x|H_a) = \frac{P(Attack_x) \cdot P(C_x)}{\#C_x} + (1 - P(Attack_x)) \quad (3)$$

$if a \notin C_x$:

$$P(x|H_a) = \frac{P(Attack_x) \cdot (1 - P(C_x))}{\#S - \#C_x} + (1 - P(Attack_x)) \quad (4)$$

These factors can be used directly, but they do not quite meet the need for efficient evidence recording, since applying them in this form would require the probability estimate associated with every node to be updated for each event. For efficiency, and to allow distributed calculation if necessary, it is very desirable to update only the estimates of nodes that are within C_x - that is to update only the scores of nodes that are indicated as possible originators of a particular event. This is possible by partially normalizing these update factors, as described in the next section.

4.2.3 Normalizing the result and localizing the update factor

The naive Bayesian updating process calculates a score which is the numerator of equation 2 for each node in the system:

$$Score_a = P(x|H_a) \cdot P(y|H_a) \cdot P(H_a) \quad (5)$$

Since there is a score for each node in the system, normalising to obtain the required probability for each node is carried out by dividing by the sum of all the node scores:

$$P(H_a|x,y) = \frac{Score_a}{\sum_{i=1}^{\#S} Score_i} \quad (6)$$

This allows probabilities to be recovered from node scores, when required.

As described above, it is desirable to localize the updating process; to do this we observe that multiplying all the update factors arising from a given event by a constant has no effect on the normalized probability. This is trivial to prove: multiply $P(x|H_i)$ in equation 5 by an arbitrary constant, K, for all nodes i in S. This multiplies both the numerator and the denominator of equation 6 by K, which cancel to give the same normalized probability that would result if the constant multiplier had not been used.

Our objective is to avoid updating node scores outside C_x, and this can therefore be achieved by choosing a constant K which sets the update factor for these nodes to unity. The required constant is the reciprocal of equation 4; by multiplying equation 3 by this factor we obtain an update factor that is applied to only the sores of those nodes within C_x.

$$\Delta_x = \frac{\frac{P(Attack_x) \cdot P(C_x)}{\#C_x} + (1 - P(Attack_x))}{\frac{P(Attack_x) \cdot (1 - P(C_x))}{\#S - \#C_x} + (1 - P(Attack_x))} \quad (7)$$

4.3 Evidence accumulation in practice

The forgoing sections provide the necessary theory to allow the details of security events to be discarded, while retaining a single score for each node which summarizes the evidence that the node is an attacker. The algorithm to achieve this is to:

1. Initialize each node score with its prior probability, $P(H_a)$.
2. For each security event:
   (a) Establish the distinguishing parameters: the probability that the event is an attack, the set of nodes that are likely to have originated the event (C_x), and the probability that C_x contains the event originator.
   (b) Calculate Δ from equation (7).
   (c) Multiply the score for each node in C_x by Δ; do not update the scores for nodes outside C_x.
3. When required, normalize the resulting node scores using equation 6, to obtain the probability that each node is an attacker.

The prior probability is of value if different nodes have significantly different priors; for example, the difference between a router and a laptop. In this case the relative difference between the types of node may be suggested by survey information. If no information is available, then the priors can be set to $1/\#S$.

The three parameters that characterize an event were discussed in the introduction to section 4.

The assumption of event independence, which is part of standard Bayesian theory, has some practical consequences for the choice of event. In many of the fields in which Bayes applies, items of evidence are not perfectly independent, but nearly enough so to allow the results to be useful. The choice of event type should therefore reflect this issue. For example, in a network attack, a particular sequence of actions may be closely related (e.g. a known exploit, followed by an outgoing connection that downloads specific malicious software). Such chains of actions are clearly not independent events, but are close enough in time to be correlated by an intrusion system and regarded as a single event (with a high certainty of being an attack). On the other hand, in a network scan, which is a series of probes to different network locations, the individual probes are only interdependent to the extent that addresses scanned will depend to some extent on past history. In these situations the designer has a choice whether to regard them as a separate events, with rather low P(Attack), or if they occur within a short time interval to regard them as a single event with a much higher P(Attack).

We are primarily concerned with comparative scores, in order to identify nodes that are distinctive and require further investigation. In practice, then, it is sufficient to use Logarithmic scores, simply adding $\mathrm{Log}(\Delta)$ to each node indicated by an event. Equation 6 can still be reconstructed from this information, but more usually, the highest node score or set of scores is chosen for further investigation.

The reader may be wondering about the value of calculating Δ at all at this stage, since we simply add its logarithm to the score for indicated nodes. However, this differs significantly from a counting algorithm, where the score for each node is incremented when it is identified as the possible source of a security event. The update value, Δ, characterizes exactly how much evidence is provided by each event. This important distinction is illustrated in the worked example presented in section 8.

5 Simulation Approach

The sections that follow evaluate the evidence accumulation process described above, partly by further exploration of equation (7), and partly by simulation. This section briefly describes the simulation rationale and approach.

It is rare to obtain useful effective network traces from real systems, especially large systems with subtle attackers.

Therefore, in order to explore a wide range of different scenarios, we use network simulation. In this paper, simulation is first used to demonstrate properties of the proposed evidence accumulation process, then used to demonstrate its effectiveness in a complex network whose overall structure is typical of those we encounter in practice.

The structure of the simulator is presented in fig 3

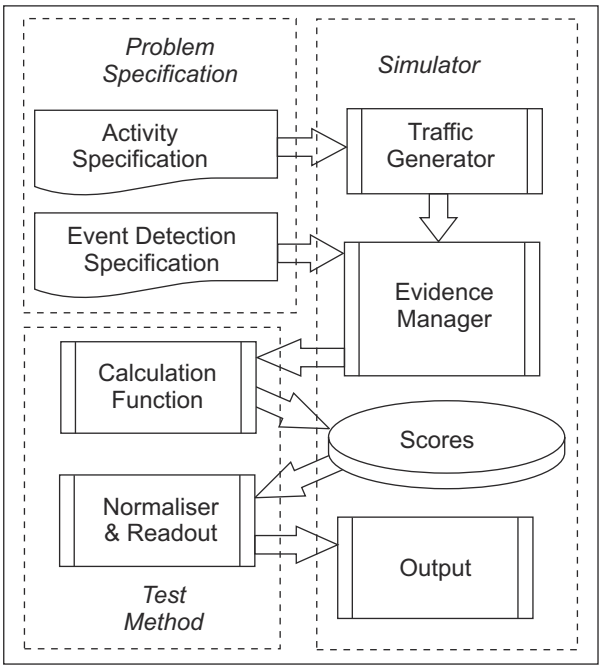

Fig. 3 The structure of the security event simulator

The simulator behaviour is controlled by two inputs: a problem specification and a test method. The problem specification contains two main parts, a description of network traffic including security related events, and a specification for how event detection behaves. Traffic is generated by the simulator at random, within rates set by the specification, and can be typed to allow different sorts of detectors and traffic to be simulated simultaneously. The event detection section specifies what traffic events can be detected, and how to determine the three key parameters: $P(Attack_x)$, C_x, and $P(C_x)$ for each event.

The test method provides score calculation and normalization functions. Usually these functions implement equations 7 and 6 respectively, but they can be exchanged with other methods, allowing exactly comparable results to be obtained; this feature is used to contrast the updating process proposed here with a counting approach, in section 8.2, below.

The simulator maintains complete separation between the calculation method and problem specification. It generates traffic according to the activity specification, which is then screened for events using the rule structure provided by the event detection specification. Events detected are provided to the calculation method, which updates the associated scores, and when output is required the normalizer can be employed to recover actual probabilities. In all the examples in this paper the normalizer is not used, since Log scores are displayed, as described in section 4.3.

6 Behaviour of evidential accumulation

Before showing a representatively difficult example of insider attack detection in section 8.1, this section explores if the evidential accumulation process has intuitively appealing behaviour. Two examples are given, the first explores a simple network with variable rate attackers, and the second is concerned with multiple overlapping groups of nodes.

6.1 A simple network

Given a single subnetwork, in which the sender can be readily identified, we explore some key questions for evidence accumulation:

- Does the evidential process identify an attacker sending at a slightly higher rate than the background of errors from normal nodes?
- If the rate of attack increases, is the process stable, and does it enable the attackers to be identified earlier?
- Does the process accommodate multiple attackers with different rates of attack (i.e. can one node hide behind another's attack)?

We simulate a single subnetwork of 50 nodes, in which the originating node of an event can be identified with certainty (i.e. $\#C_x = 1$, and $P(C_x) = 1$); we assign $P(Attack)$ an arbitrary probability of 0.083. Time is divided into slots (e.g. single minutes) and the average background rate of random innocent events that may be misinterpreted as attacks is 1/50 per node – in other words, one event per minute. Three nodes within the subnetwork are designated attackers, and they generate random attacks at rates of 2, 4 and 8 times the total background rate.

The scores resulting from this scenario are shown in Fig. 4. All three attack nodes are well distinguished from the background level of events, which is indicated by the 'control' result, which is the score for a typical innocent node. As would be expected, if the attack rate is higher, the discrimination improves. The accumulation of evidence is well behaved, and the higher rate nodes do not interfere with the accumulation of evidence relating to attackers operating at a lower rate.

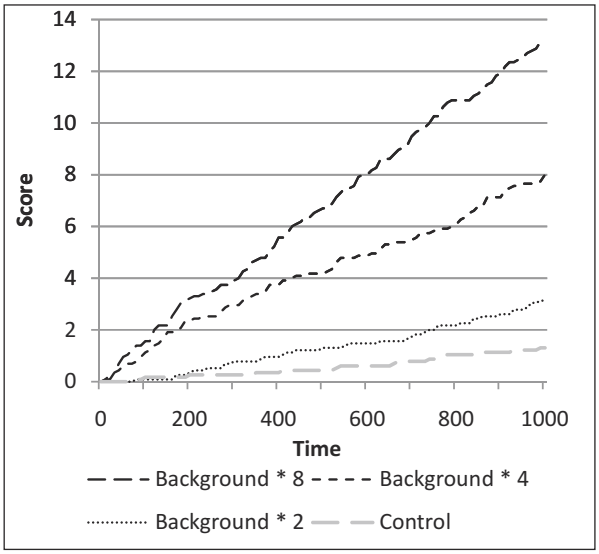

Fig. 4 Scores resulting from three attackers, at different rates, in a single subnetwork

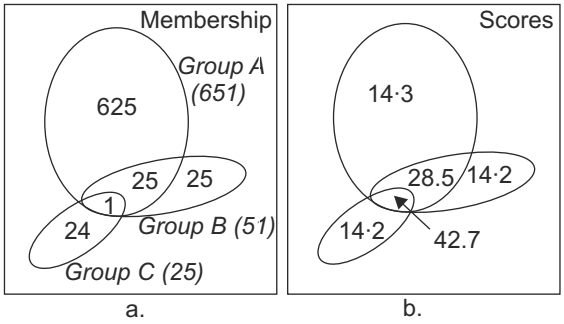

Fig. 5 Investigating the scores for individuals in different size groups, with the same behaviour. (a) shows the sizes of the three groups and their overlaps; (b) gives the final accumulated score for representative single individuals.

6.2 Evidence from variable sized overlapping groups of individuals

Electronic networks are likely to have stable structures, resulting in relatively simple and consistent groups of nodes (i.e. the sets of C_x) that can be identified as the originator of an event: for example the individual node, the logical subnetwork, or a local facility network. Under these circumstances it is important that the same evidence is accumulated against two nodes whose behaviour is the same, but who are usually identified with different size groups of nodes.

This issue becomes more significant when dealing with social networks where innocent individuals associated with 'events' (e.g. individuals visiting a specific internet cafe, or taking a flight to a particular destination) are present by accident, rather than a result of a fixed network architecture. These groups of individuals are much more ad-hoc, and the attackers lie somewhere in their many random intersections.

To investigate how well disparate groups are handled by the evidential accumulation process, we simulate three events that identify three radically different sized groups of individuals (651, 51 and 25). Fig. 5 (a) illustrates the resulting sets of individuals, and the size of two overlaps. All the individuals in the system generate events at the same rate (P(event)=0.014), which results in a total of 10 events per minute from the 700 individuals). The exceptions are the individuals in the overlaps, who generate events in all the groups with which they are identified at the same rate as other group members, so they generate additional events pro-rata to their group memberships. P(Attack) is .099 for all events, and the simulation was run for 100,000 minutes. The resulting scores are given in Fig. 5 (b).

The results show that individuals generating 'false alarms' at the same rate, but ascribed by the detection process to different sized groups, receive the same score; the small differences are attributed only to small differences in the random generation of events. This effective normalization between different group sizes is an important feature of the proposed evidence accumulation process, because the objective is to identify individual behaviour, and avoid simply identifying individuals on the basis of the groups to which they are ascribed. Section 8.2, below, provides a dramatic example of how badly alternative accumulation processes perform if they do not possess this attribute.

In this case the scores for the individuals in the intersection of these groups is predictable, as it arises simply from the superposition of event rates. The reader should bear in mind that this section is concerned only with basic behaviour. A real social network problem is likely to identify many more groups, with considerable uncertainty about their membership, making it rather more difficult to identify which individuals are potentially significant.

7 Limiting factors for event evidence

The equation for updating evidence can also be used as a measure of effectiveness for event detectors. Specifically, we can enquire under what circumstances does an event add information to our estimate that an individual is an attacker.

The threshold of usefulness of event detection occurs when Δ from equation (7) is unity; a value above unity adds evidence to some hypothesis, below reduces evidence against any of the identified nodes. We require:

$$\frac{\frac{P(Attack_x)\cdot P(C_x)}{\#C_x} + (1 - P(Attack_x))}{\frac{P(Attack_x)\cdot (1-P(C_x))}{\#S - \#C_x} + (1 - P(Attack_x))} >= 1 \qquad (8)$$

Multiplying by the denominator of the left hand side, then subtracting $(1 - P(Attack_x))$ from both sides gives:

$$\frac{P(Attack_x) \cdot P(C_x)}{\#C_x} >= \frac{P(Attack_x) \cdot (1 - P(C_x))}{\#S - \#C_x} \quad (9)$$

Dividing out $P(Attack_x)$, and multiplying out the denominators, we obtain:

$$(\#S - \#C_x) \cdot P(C_x) >= \#C_x \cdot (1 - P(C_x)) \quad (10)$$

Adding $\#C_x \cdot P(C_x)$ to both sides, then re-arranging, gives:

$$P(C_x) >= \frac{\#C_x}{\#S} \quad (11)$$

This result is shown graphically in Fig. 6.

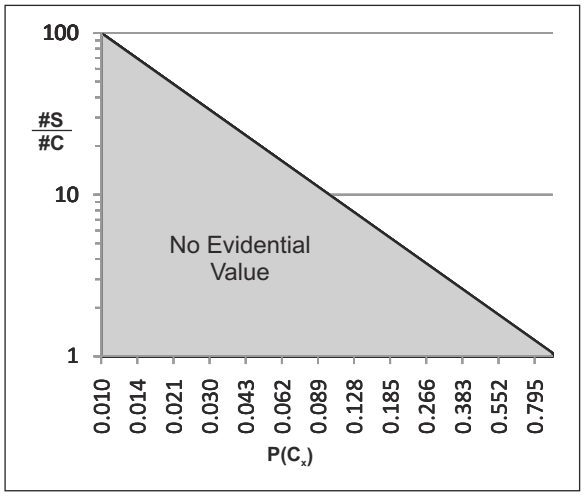

Fig. 6 Evidential value is limited by the effective size of the set of individuals implicated by an event

This result is valid provided $P(Attack_x)$ is not zero. Values of $P(Attack_x)$ close to unity provide most evidence,and as $P(Attack_x)$ tends toward zero the update factor approaches unity, which is to be expected since the weight of evidence is reducing.

Assuming that $P(Attack_x)$ is not approaching zero, then the factor that limits the value of evidence is the certainty of estimation of the set of nodes responsible for an event, in relation to the relative size of that set to the whole population. In simple terms the limit can be regarded as the point at which the whole population is implicated by an event. A practical illustration of this threshold is given in section 8.3, which shows how evidence in a representative network behaves with decreasing certainty of event attribution.

8 Insider Attacks

This section shows that an insider attack in a representative network can be identified by the proposed evidence accumulation process, contrasts the principled accumulation of evidence with a simple counting scheme, and explores how evidential accumulation behaves as the attribution of nodes that originate events becomes less certain. All the examples in this section use the same network and the same simulation seeds, to provide comparable results.

8.1 A difficult detection problem

The network used in this section is a medium-sized system (3000 endpoints) with features that are representative of the problem space, including:

- Sensors with different capabilities; for example, certainty of detection and ability to identify source nodes.
- Attackers whose rate of attack is significantly below the background rate of false positive alerts for the system.
- Attacks that employ address spoofing.

An important practical issue is the estimation of the three parameters that characterize a security event; relating these to actual systems and assessing the need for accuracy is subject to ongoing study. To date it has been possible to achieve realistic results by assigning *P(Attack)* as a fixed value for a given sensor within a deployment context, and by creating a simple rule-set that maps the network connection associated with an event to a set of nodes, giving C_x and $P(C_x)$, depending on the configuration and protocol.

The network used in this example is given in Fig. 7. This network has 3000 nodes, most of which are user systems located in eleven separate client subnetworks, in sizes that range from 33 nodes to 500. Two of these subnetworks have nodes that are subverted and are attacking the system. The purpose of dividing the clients into several subnetworks (apart from the fact that this is a standard configuration) is to contrast the detectability of attackers in different sized subnetworks, given that in many cases it will be possible to identify only the subnetwork from which an attack originated. This arrangement allows us to investigate the scores accrued for an attack node (3 or 403) versus normally-behaving nodes in the same subnetwork, and nodes in a control subnetwork of a significantly different size.

Most of the traffic in the system is between the clients and servers, via the core network. Router and firewall detail is not shown, and because the object is to investigate evidence accumulation rather than event generation we model two unspecified types of security event: those that can be detected within client subnetworks, and events in the server farm. For example, an event could be an attempt to connect to an exploitable network port.

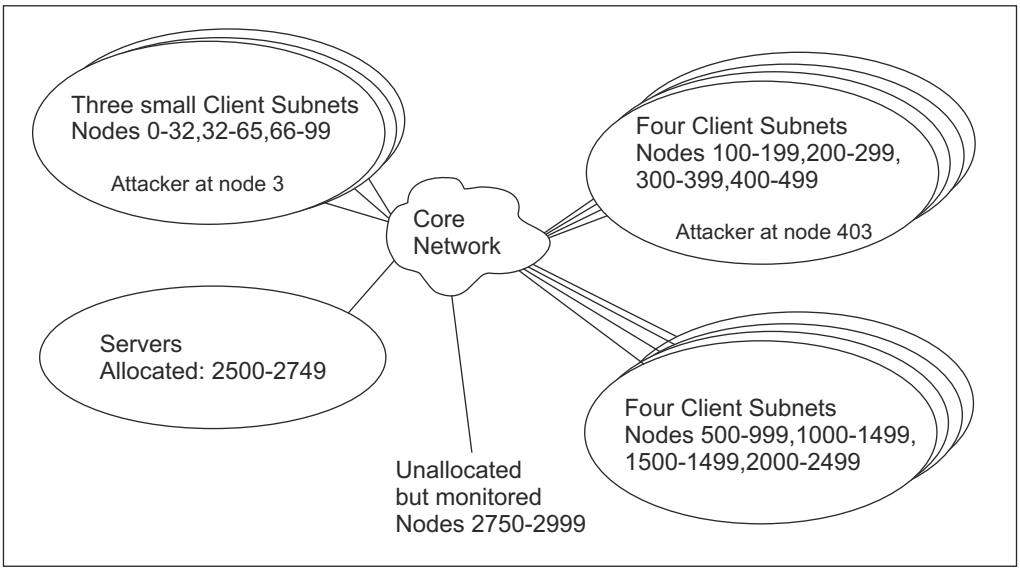

Fig. 7 Test Network

Attackers are expected to generate security events at a rate that is much lower than the background rate of 'mistakes' by normal clients, in order to remain undetected. In the simulation below, time is measured in arbitrary clocks (e.g. minutes), and the probability of a normal client generating a security alert in any time slot is 1/20; in other words the system suffers an average of 150 false alarms every minute. In contrast, attackers generate events at a rate of 1/10; one event every 10 minutes.

In addition to the low attack rate, to further avoid detection, attackers use address spoofing. Events detected outside the subnetwork containing the attacker can only be assigned to the whole subnetwork. Only events identified within the subnetwork containing the attacker (i.e. directed toward nodes within that subnetwork) can be traced to a specific node.

An outline calculation illustrates the difficulty of this problem. Consider the attacker at node 3. Viewed from outside, the subnetwork can be expected to generate innocent background events (false alarms) at a rate of 1.6 events per minute ($33 * 1/20$). The events generated by the attacker are distributed at random across the network, so of these, 33/3000 are towards the attacker's own subnetwork; these are the only events that can be identified to a particular attacker, and they occur at a rate of one every 909 minutes ($P() = 1/10 * 33/3000$). The simulation is over 10^4 minutes; in this time we expect a total of 1.5 million events in the system as whole ($10^4 * 3^3/20$) of which 16000 ($10^4 * 1.6$) can be ascribed to the attacker's subnetwork, and just 11 ($10^4/909$) to the attack node.

Given this information the reader could devise a solution to identify the attacker, but the problem addressed here is how to use all the available information when the location of the attacker and the traffic patterns are unknown in advance.

In summary, the event parameters used in the simulation are:

C_x contains all the nodes in the source subnetwork, unless the destination of the network message that caused the event is in the same subnetwork as the source, in which case C_x contains just the source node.

$P(C_x)$ is set to unity, since C_x includes the node which originates the traffic. (The effect of varying this parameter is discussed in section 8.3, below.)

$P(Attack)$ is set to 0.043 for all used locations except the server nodes, for which a value of 0.0099 is assigned, and events from a client to its own subnetwork, which are given a value of 0.076.

These are arbitrary, for the sake of demonstration, although they do reflect likely differences in expectation. For example, it seems plausible that an incident at a location to which most of the traffic is directed is less likely to be an attack, but in practice that is dependent on the actual event. The only special feature in the choice of value is avoiding fractions such as 30/3000 that match the system topology, although we have not detected any problems arising from such choices. Varying these parameters results in different scores, but not at the expense of overall discrimination.

The network simulator was used to generate random traffic as specified above, and the scores for the resulting security events were accumulated as described in section 4.3. The results are shown in Fig. 8.

Fig. 8 shows node scores as they are accumulated. The nodes shown are attackers (3,403), representative nodes in the same subnetworks (4,404), and a control node in a large subnetwork with no attackers (1000). Nodes (3,4) are from

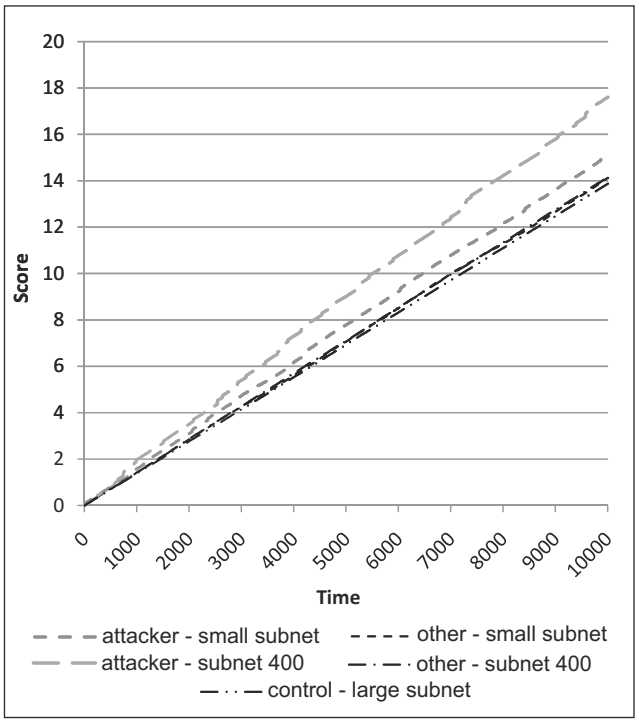

Fig. 8 Network Simulation Results

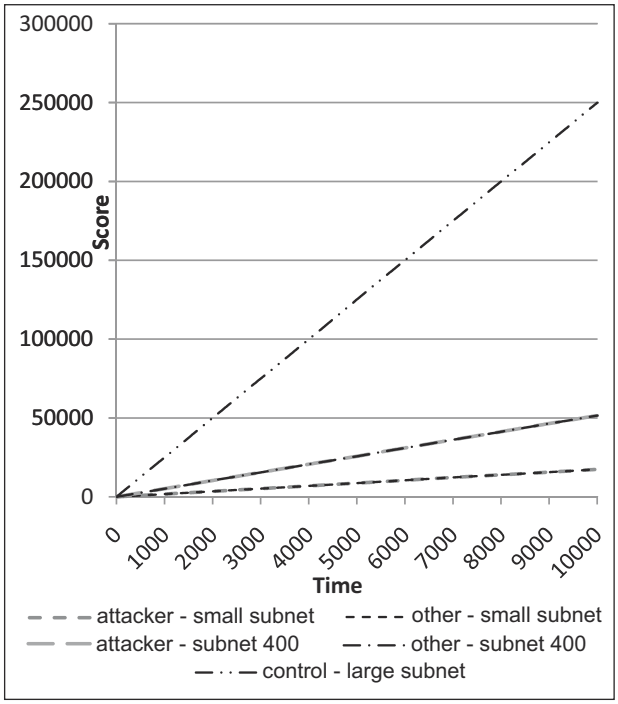

Fig. 9 Counting Algorithm Performance

33-node subnetworks, nodes (403, 404) are from 100-node subnetworks, and node 1000 is from a 500 node subnetwork, of which there are 4, which together contain a significant proportion of the nodes in the network.

The results show that insider attacks can be clearly distinguished from background noise in the system.

For each size of subnetwork the proposed scoring clearly distinguishes the attacker as an individual from other nodes within the same subnetwork. Nodes are similarly scored regardless of the size of the subnetwork in which they reside, and there there is only a small difference in score between other nodes in the subnetworks containing attackers and the control node (approximately 2%), which can be attributed to attackers slightly raising the score of their own network.

8.2 Contrasting evidence accumulation with event counting

The effectiveness of the approach to evidence accumulation presented in this paper can be judged by comparison to the counting algorithm used to introduce section 4, and adopted by some researchers. The same events are generated with the same characteristics as described in the previous section, but the calculation function uses counting rather than evidence accumulation, by simply incrementing node scores if the node is identified as a possible source of an event (i.e. is in C_x). The results are presented in in Fig. 9.

On a realistic problem, the counting approach fails in almost every respect. Attackers are not distinguished from other nodes in their subnetwork. Instead, the primary distinction is between nodes on the basis of network size; essentially the larger subnetworks generate more background traffic, so receive a proportionately higher score.

8.3 Uncertainty in identifying the nodes that originate an attack

The network example, above, assumed that the set of nodes associated with an attack could be identified with certainty. Fig. 10 shows the results of a series of simulations, with varying degree of certainty of attribution of the nodes that originate each event. For the sake of illustration all events were given the same $P(C_x)$; in practice this would vary depending on the type of the event and the position in the network where it was detected.

As would be expected, the greater the uncertainty the lower the overall score. Importantly, the scores remain correctly ordered; even with high degrees of uncertainty of attribution, the attacking nodes would be identified for further investigation.

An interesting feature of this simulation is the control node, whose score decreases more quickly than the others, eventually becoming negative. Section 7 showed that the limit of evidential value is at the point where the degree of uncertainty approximately encompasses the whole network. The control node in this case is in a subnetwork of 500 nodes, in a network with 3000 nodes; equation 11 gives the

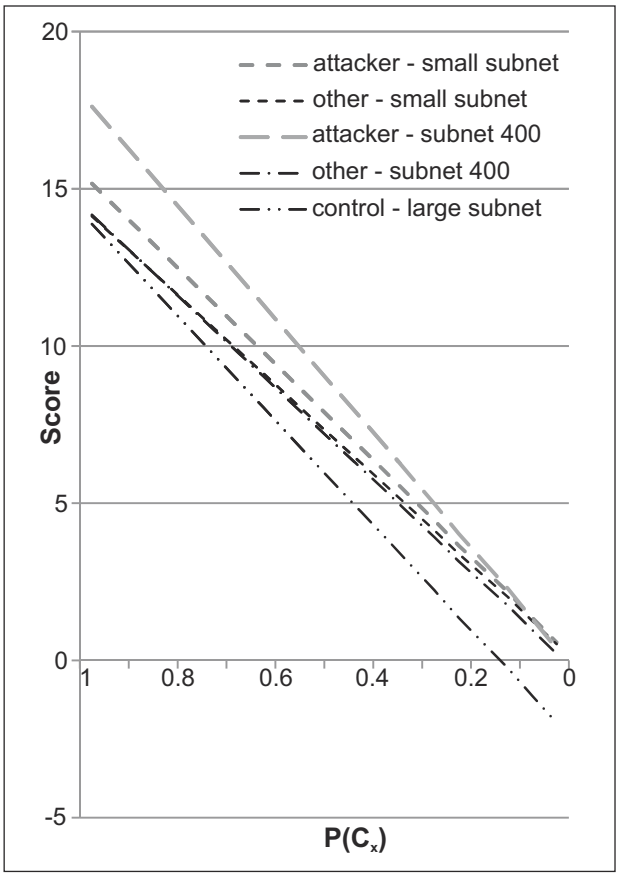

Fig. 10 The variation in final scores of the network simulation with $P(C_x)$

threshold value as $P(C_x) = 500/3000 = 0.17$, which is consistent with the point at which the simulated score is zero.

The decreasing score of the control node in this example is therefore a result of the attribution of the origins of events approaching the size of the whole system. This effect is also the most likely cause of the small divergence between non-attacker nodes (subnet 400 diverges a little); however, given the random nature of the simulation this difference is too small to justify a claim of significance.

9 Discussion

The proposed evidential updating process is effective because it relates event evidence to the hypothesis that a node (or user) is the attacker. This change of reference frame allows event data to be discarded, while retaining the weight of evidence for attackers. The process scales linearly with the number of nodes in the system, and is applicable to a very wide range of systems and circumstances.

Bayesian statistics has been used, rather than the simple probability ratios that would be suggested if information theory was employed, in order to effect the change of viewpoint from the event to the attacker. The update factor, Δ, importantly takes account of ancillary information such as the number of nodes that are indicated by the event, and the degree of certainty in their estimation.

Δ can be used as a figure of merit for sources of information; essentially, if Δ is consistently fractional for a sensor, then the resulting events will degrade the quantity of available information, rather than improve it. We show that this depends on the uncertainty of estimation of the set of event originators, compared with the size of the overall system. Essentially, if it is not possible to distinguish between nodes in the network, the event adds no value. This has important practical consequences for intrusion system and network design: as much attention should be given to identifying the source of events as to the false-alarm rates of sensors.

The attributes described in section 4 (probability of attack, the likely originating nodes of an event, and the probability that the event originator is in this set) are not specific to any particular type of event detector, and can be applied at different levels of abstraction, if necessary within the same system.

There are a number of practical considerations that are subject to ongoing study. The first implementation decision is which real components are regarded as 'nodes': should nodes model all network components, just routing components and endpoints, or just endpoints such as clients, servers or users? To date, only endpoint nodes have been considered; this decision is based on the prior probability of network components originating attacks, and the convenience in associating events with their possible sources. Further practical work is also needed to relate the three event parameters to actual intrusion sensors and networks.

A key practical issue is how to determine which nodes are a potential source of any particular event, and to what degree. Ideally this assessment would be evidence-based using recent network history, but although this is feasible in principle, it is an open question if this can be achieved in practice. However, even simple strategies, such as the one used in section 8.1, provide demonstrable benefit.

10 Conclusion

This paper provides a solution to a critical problem in insider attacker discovery: how to combine events from multiple sensors, and manage the data explosion that is otherwise needed to support the identification of long-running attacks.

The key concept is to move away from maintaining models and evidence of behaviour, and instead maintain an incremental assessment for every user/node in the system that the node is an attacker. This approach is extremely scalable; the updating algorithm is soundly based in Bayesian statistics, and avoids the need for global updating after each event. The

approach is well behaved, in the sense that higher volumes of attack make detection easier, and in a worked example which includes several of the difficulties faced in practice, it significantly outperforms counting algorithms (see section 8.1).

In addition, this work identifies the attributes or parameters that need to be standardized for disparate sources of security event to be combined, allowing the use of a wide range of different sensors at different levels of abstraction. The key criteria for a sensor (see section 7) is that it tends to provide information rather than add confusion, and a side effect of the updating process presented here is a criteria for deciding when this is the case.

Research on this approach is ongoing, both using simulation and relating the work to real sensors. The updating process described in this paper reflects a change of base hypothesis from our earlier publications, and resolves some of the open questions and marginal discrimination observed previously; remaining open questions are described in section 9.

References

(1998) CERT incident note IN-98-05: Probes with spoofed IP addresses

Bace R, Mell P (2001) Intrusion detection systems (IDS). Tech. Rep. SP 800-31, National Institute of Standards and Technology (NIST)

Brackney RC, Anderson RH (2004) Understanding the insider threat. Tech. Rep. Proceedings of March 2004 Workshop, RAND National Security Research Division

Bradford PG, Brown M, Perdue J, Self B (2004) Towards proactive computer-system forensics. In: International Conference on Information Technology: Coding and Computing (ITCC 2004), IEEE Computer Society, pp 648 – 652

Buford JF, Lewis L, Jakobson G (2008) Insider threat detection using situation-aware MAS. In: 11th International Conference on Information Fusion, IEEE Xplore, Cologne, Germany, pp 1–8

Chebrolua S, Abrahama A, Thomas JP (2004) Feature deduction and ensemble design of intrusion detection systems. Computers and Security 24(4):295–307

Chivers H, Nobles P, Shaikh SA, Clark JA, Chen H (2009) Accumulating evidence of insider attacks. In: The 1st International Workshop on Managing Insider Security Threats (MIST 2009) (In Conjunction with IFIPTM 2009), CEUR Workshop Proceedings

Colombe JB, Stephens G (2004) Statistical profiling and visualization for detection of malicious insider attacks on computer networks. In: The 2004 ACM Workshop on Visualization and Data Mining for Computer Security, ACM Press, pp 138–142

Eberle W, Holder L (2009) Insider threat detection using graph-based approaches. In: Cybersecurity Applications & Technology Conference For Homeland Security (CATCH), IEEE Computer Society, pp 237–241

Goodin D (2007) TJX breach was twice as big as admitted, banks say. The Register

Heberlein T (2002) Tactical operations and strategic intelligence: Sensor purpose and placement. Tech. Rep. TR-2002-04.02, Net Squared, Inc.

Nguyen N, Reiher P, Kuenning GH (2003) Detecting insider threats by monitoring system call activity. In: 2003 IEEE Workshop on Information Assurance, IEEE Computer Society, United States Military Academy, West Point, pp 18–20

Randazzo MR, Cappelli D, Keeney M, Moore A, Kowalski E (2004) U.S. secret service and CERT coordination center/SEI insider threat study: Illicit cyber activity in the banking and finance sector. Tech. rep., Software Engineering Institute, Carnegie Mellon University

Spitzner L (2003) Honeypots: Catching the insider threat. In: 19th Annual Computer Security Applications Conference (ACSAC '03), IEE Computer Society, pp 170–179

Staniford S, Hoagland JA, McAlerney JM (2002) Practical automated detection of stealthy portscans. Journal of Computer Security 10(1/2):105–136