Detecting Insider Threats Using Ben-ware: Beneficial Intelligent Software for Identifying Anomalous Human Behaviour

Andrew Stephen McGough[1]*, Budi Arief[2], Carl Gamble[2], David Wall[3], John Brennan[1],
John Fitzgerald[2], Aad van Moorsel[2], Sujeewa Alwis[4], Georgios Theodoropoulos[1], Ed Ruck-Keene[1]
[1]Durham University, Durham DH1 3LE, UK
{stephen.mcgough, j.d.brennan, georgios.theodoropoulos, e.a.ruck-keene}@durham.ac.uk
[2]Newcastle University, Newcastle upon Tyne NE1 7RU, UK
{budi.arief, carl.gamble, john.fitzgerald, aad.vanmoorsel}@newcastle.ac.uk
[3]University of Leeds, Leeds LS2 9JT, UK
d.s.wall@leeds.ac.uk
[4]Insighlytics Ltd, York, UK
sujeewa@insighlytics.com

Abstract

The insider threat problem is a significant and ever present issue faced by any organisation. While security mechanisms can be put in place to reduce the chances of external agents gaining access to a system, either to steal assets or alter records, the issue is more complex in tackling insider threat. If an employee already has legitimate access rights to a system, it is much more difficult to prevent them from carrying out inappropriate acts, as it is hard to determine whether the acts are part of their official work or indeed malicious. We present in this paper the concept of "Ben-ware": a beneficial software system that uses low-level data collection from employees' computers, along with Artificial Intelligence, to identify anomalous behaviour of an employee. By comparing each employee's activities against their own 'normal' profile, as well as against the organisational's norm, we can detect those that are significantly divergent, which might indicate malicious activities. Dealing with false positives is one of the main challenges here. Anomalous behaviour could indicate malicious activities (such as an employee trying to steal confidential information), but they could also be benign (for example, an employee is carrying out a workaround or taking a shortcut to complete their job). Therefore it is important to minimise the risk of false positives, and we do this by combining techniques from human factors, artificial intelligence, and risk analysis in our approach. Developed as a distributed system, Ben-ware has a three-tier architecture composed of (i) probes for data collection, (ii) intermediate nodes for data routing, and (iii) high level nodes for data analysis. The distributed nature of Ben-ware allows for near-real-time analysis of employees without the need for dedicated hardware or a significant impact on the existing infrastructure. This will enable Ben-ware to be deployed in situations where there are restrictions due to legacy and low-power resources, or in cases where the network connection may be intermittent or has a low bandwidth. We demonstrate the appropriateness of Ben-ware, both in its ability to detect potentially malicious acts and its low-impact on the resources of the organisation, through a proof-of-concept system and a scenario based on synthetically generated user data.

Keywords: Insider threats; detection; anomalous behaviour; human behaviour; artificial intelligence; assistive tool; ethics.

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, volume: 6, number: 4 (December), pp. 1-44

*Corresponding author: School of of Engineering and Computing Sciences, Durham University, Durham, DH1 3LE, UK, Tel: +44-191-33-41749

†This paper is an extended version of the work originally presented at the 7^{th} ACM CCS International Workshop on Managing Insider Security Threats (MIST'15) [19] – a full list of the additional work presented here can be found at the end of the introduction

1 Introduction

An insider threat can be defined as [6]: *the intent to inflict harm by one who has special knowledge or access to confidential information.* In this situation, we are dealing with an attacker who is an authorised user of the system, making the insider threat a difficult problem to resolve – especially in the case where the difference between legitimate acts of an 'insider' and those of a malicious 'insider' vary only marginally. Detection then becomes a process of identifying when a user of the system performs an act that is either not required by their job role, out of character for that user or suggests that the act they are performing will lead to harm against the organisation – for example data being stolen or corrupted. As none of these scenarios lend themselves to simple rules which can be evaluated by a computer, it is very difficult to programatically identify an insider threat. Instead, approaches tend to look for tell-tale signs that indicate such an attack is taking place, for example through identification of abnormal usage patterns by a user, or even by trying to catch the perpetrator red handed using honeypots [29] – specific fake data which no legitimate users should need to access.

The danger of insider threats for large closed organisations, in which the security of their information systems is a key asset, is all too apparent from stories such as the Edward Snowden and the National Security Agency (NSA) [11]. In such situations, it is advantageous to be able to detect quickly that a user (i.e. a legitimate employee) is acting in a manner which is anomalous to their normal behaviour. This is especially important when the behaviour appears to conflict with the prescribed practices or guidelines of the organisation. This could indicate that such an employee[1] has become an insider who acts maliciously against the organisation, or that their credentials have been compromised, i.e. someone else is masquerading as them [2]. Alternatively, this may just be the user performing a legitimate act. We categorise all of these as *anomalies* to a user's normal behaviour. Distinguishing between these cases would allow for a reduction in the number insider threats which are incorrectly identified as attacks on the system – the so-called false positives – thus allowing for more effort to be dedicated to the real attacks on the system.

The types of anomalies that we want to identify are those in which an employee is behaving outside of the prescribed rules of conduct set out by the organisation. For instance, such an employee may try to steal confidential information, seek to insert false information into the data store, or modify internal records for their own or someone else's benefit. Identification of malicious behaviour from benign behaviour is made complicated by the fact that a user may perform acts outside of those prescribed by the organisation's rule set without malicious intent [16, 35] – for example over-riding of security systems in order to expedite the completion of a legitimate task. In order to tackle this problem, we use human behaviour modelling along with Artificial Intelligence (AI) [24, 31, 34] to identify when a user deviates from their normal usage pattern. In this way, each user is compared with their own normal profile, which may or may not include acts considered against the organisation's rules of conduct, in order to determine when they are acting anomalously – whether this is due to malicious or benign intent. This can then lead to further investigation of the user, either to clear them or to continue with closer monitoring, which will allow for an earlier detection of the user moving from being a well behaved citizen to an insider threat. If, however, a user has always behaved with malicious intent from the start of their employment then this approach would be inappropriate as no change in behaviour would be detected. Under these circumstances, a comparison with a profile derived from others in similar occupational and organisational roles – the user's peers – would allow for the detection of this abnormal activity.

The work presented in this paper is a result of an interdisciplinary collaboration involving computer scientists, a criminologist and behavioural analysis experts. Although there exist systems that utilise AI

[1]Without loss of generality we use 'employee' here to refer to a legitimate member of an organisation whether they are paid or not. We also refer to an employee whilst using a computer as being a 'user'. However, the two terms can be considered as equivalent.

for identifying user anomalies, to the best of our knowledge we are the first in using human factors – especially analysis of human behaviour – to influence, inform and develop an improved AI detection techniques. The main underlying problem here is that human behaviours are not easily reduced to quantifiable values that can readilly be used within a computer system. They can, however, be reduced to a number of models that include, for example, the five states of wellbeing (happiness) discussed later in the paper.

Conventional AI approaches to classification problems are ill-suited for the detection of insider threats. The reason behind this is that such AI approaches are typically based on machine learning, in which a training phase is required. During the the training phase each test set is labelled – in our case either 'good behaviour' or 'malicious behaviour' – and the machine learning algorithm 'trained' such that it produces the same label for each test set. As such, this approach is well suited to situations where there are equal proportions of 'good' and 'malicious' data. However, this is not the case with insider threats. Thankfully the proportion of malicious behaviour against the organisation by an insider (i.e. the 'malicious' set) tends to be much smaller than that of legitimate activities (the 'good' set). Therefore obtaining a balanced training set is not possible which leads to poorly trained systems incapable of detecting the behaviour we desire. Instead, we make use here of an AI approach for categorising 'normal' behaviour and look for outliers from this behaviour. What constitutes 'normal' behaviour (a contested concept itself) requires a learning phase in order to understand the individual, their occupational role and their particular circumstances. This does introduce the additional problem of machine training time as we need to develop a sophisticated model for each individual user. The training time can be reduced by using a 'generalised' profile for particular job-roles, or after a short period of time we can 'match' new users with the most likely similar co-workers. As in all organisations, this learning process could, for example, follow existing standard competency and trust establishing activities including employee induction and skills development alongside occupational assessment and progression.

Organisational assets such as files can be tagged with varying levels of sensitivity or risk (e.g. High-Sensitivity $>$ Medium-Sensitivity $>$ Low-Sensitivity $>$ Unclassified). When a user interacts with these assets the interactions could be recorded. Collecting this information allows us to build up a profile of the 'normal' behaviour of an employee and determination of a boundary around this normal behaviour. Once we have a profile of normal behaviour, we can start to detect abnormal behaviour – situations where an employee's behaviour is significantly outside their boundary of normal behaviour. Our work is informed by various risk profiles, including those that have malicious intents, as well as negligent and well-meaning insiders as based on the work by Loch *et al.* [16] and Wall [35]. In order to identify the usage pattern of employees within an organisation, it is necessary to capture low-level information on how users are interacting with the system, for example, when they log in and when they copy a file to an external (USB) device. In doing so, we are able to apply machine learning to these datasets in order to develop a model of the user's normal profile.

Most existing approaches for the identification of anomalous user behaviour rely on a centralised data collection approach, where the data, collected on the activity of individual users, is stored on one dedicated central server – which is also used for analysis. In situations where the organisation has multiple sites, either all sites share one central server (which means there is the need for a provision of a large server and high-bandwidth networking) or each site has its own server (which means a user's activities on different sites are not guaranteed to be correlated, unless there is a rigorous synchronisation process to complement it). The effect of this centralised approach is more significant if sections of the organisation can become detached from the main network (e.g. remote offices with poor/intermittent networking) and / or employees may roam between sites (e.g. using a laptop). As an alternative to a centralised IT provision, the machine learning element could be moved to individual hosts. However, this would necessitate each host being capable of substantial computational effort (without impacting on normal operation), which is unlikely in many organisations as their IT provision comprises of networks

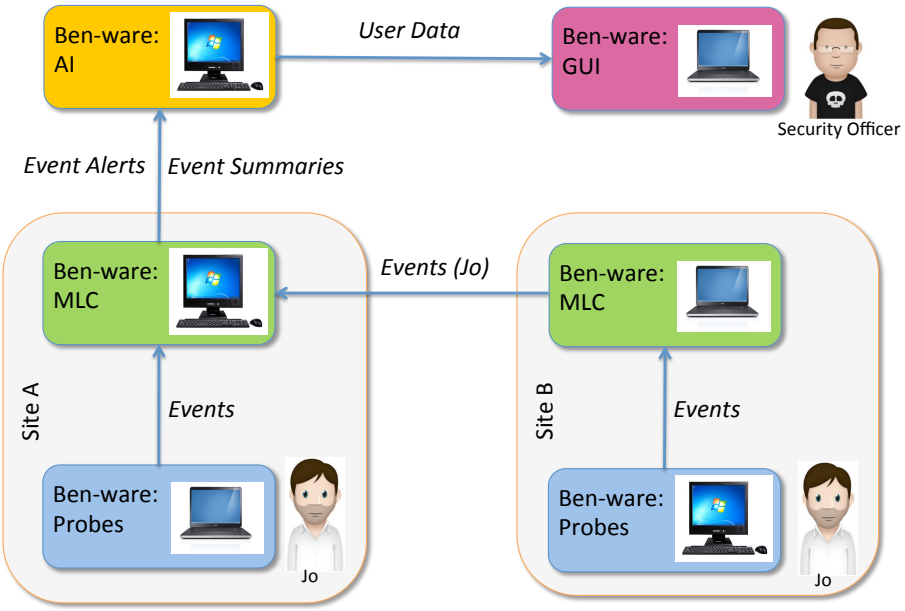


Figure 1: Overview of the Ben-ware approach

of heterogeneous legacy computers. Furthermore, this situation will very likely lead to the decrease (or even a complete loss) of the ability to detect anomalous behaviour across the whole organisation.

In order to address these problems, we propose a *distributed infrastructure* (shown diagrammatically as Figure 1) consisting of (i) lightweight probes hosted on each computer, (ii) intermediate nodes (Mid-Level Controllers – MLCs) which can cache and forward data, and (iii) high-end (AI) nodes, each of which can process the data from a small subset of users. This allows us to overcome the limitation of low-power computers within an organisation as the computationally heavy work is sent to computers within the organisation with enough spare computational power to be able to process a small number of users without effecting the computer's primary use. To illustrate our approach, consider the user Jo who works primarily at Site A (Figure 1). At some times Jo will visit Site B and perform some work there. In both sites, the computers that Jo logs into are instrumented with probes for capturing data about Jo's usage. While working at Site A, this data will be sent to a local AI for processing via a local Mid-Level Controller. When Jo visits Site B, the data collected by the probes there will be forwarded via a local MLC back to the MLC at Jo's normal site before forwarding that to the AI. In this way, all of Jo's activity will be passed to the AI. Note that the MLC at Site A keeps the history of Jo's activities and is thus not just a passive forwarder, unlike the MLC at Site B. It should also be noted that this is a highly simplified view of the architecture. In a more complex set up, users at site B would have local AIs and each site would have multiple MLC to balance the workload.

It is also necessary here to bring in the concept of coverage of the data sets being accessed. This provides an ability to observe how someone is accessing the dataset – are employees only accessing certain parts of the data sets or are they slowly covering the entire dataset? – is someone (either the genuine employee or someone with their credentials) accessing an unusual set of files from the data set? As outlined earlier, most organisations will likely have an existing data plan and structure for how files are organised. This can assist the machine learning process in identifying how 'normal' it is for a user to access a particular sub-branch of the filing system, for instance.

Finally, it is very important to note that it is impossible devise a solely-computer-based solution that

can detect criminal activity within a system. At best, such a solution might be able to detect breaches in prescribed practices – for example, identifying potential threats while reducing the false-positives through the use of AI and human-factors. In any organisation, there is a need to understand where and when cyber security issues (especially the threats and risks) might turn into cybercrime problems (i.e. the actual harm) for the organisation. One of the challenges associated with any attempt to identify potential risks and harms is in dealing with *false positives*. As such, one of the principal functions of Ben-ware is to help minimise false positives by focusing on individuals who are more likely to become a threat. This enables security personnel to allocate resources more efficiently and effectively.

This paper is an extended version of the work originally presented at the 7^{th} ACM CCS International Workshop on Managing Insider Security Threats (MIST'15) [19]. The key differences from the workshop version are:

- A more comprehensive overview of related work, which now has its own section
- An extension of the human factors consideration
- An expansion of the vision to include an example on user mobility between sites
- A detailed description of the message exchange protocol in Ben-ware
- An explanation of how the synthetic data (used to demonstrate the feasibility of the Ben-ware approach) is generated
- An additional discussion regarding ethical issues

The rest of this paper is organised as follows. Section 2 discusses related work and is followed by Sections 3 and 4, which explore the background to the Ben-ware approach and the vision, respectively. Section 5 discusses the current prototype implementation of the Ben-ware system along with a discussion on the synthetic generation of log data for testing the AI. The results – in terms of Ben-ware's ability to detect anomalies and the impact on the hardware – are presented in Section 6. A discussion of the implications of Ben-ware is presented in Section 7, before we provide conclusions and future directions in Section 8.

2 Related work

Few models have been developed for explaining insider attacks. Schultz [27] proposes an overarching model of Capability, Motive and Opportunity based on analysis of the work by Parker [23], Tuglular and Spafford [33], Suler [30] and Shaw *et al.* [28]. Schultz goes further to propose a set of traits that a human may exhibit which would indicate that an insider attack has or is about to take place. There are six traits: Deliberate Markers (purposefully leaving a trail to indicate the crime has happened), Meaningful Errors (non intentional trail of evidence), Preparatory Behaviour (as the perpetrator prepares for the attack they will investigate and collect information which can be observed), Correlated Usage Patterns (performing similar commands on different systems - such as searching for files), Verbal Behaviour (such as signs of anger towards the organisation) and Personality Traits (such as introvertion).

Colwill [7] provides a practitioners viewpoint on the problem of insider threats and identifies that the current preference for outsourcing increases the chances of insider attacks. He argues that users should only be given access to the minimal amount of material required to conduct their work and for the minimum amount of time – thus minimising the impact of potential attacks. Going further to argue that most situations can be identified by knowing and monitoring your employees. We exemplify this within

our system by assuming that line managers (or Human Resources) can provide invaluable input to the system about employees state of 'happiness'.

Maloof and Stephens [18] stress the importance of contextual information along with the tracking of how users access and interact with information in order to successfully identify insider threats. This is a viewpoint we concur with as part of our approach.

Current state of the art approaches to the detection and prevention of insider threats are: using complementary suites of monitoring and auditing approaches [3]; the combination of structural anomaly detection approaches along with the modelling of psychological factors in order to identify those employees most likely to be insiders [4]; distinguishing between malicious and benign behaviours by examination of behavioural characteristics of potential insiders [5]; developing decision support systems – for example a 10-step program to maximise the efficiency of an organisation's analytics [20]; along with a multi-disciplinary approach for better understanding of the underlying problem [22].

More practical approaches to the detection of insiders have focused on: identification of misuse of legitimate access to resources (e.g. documents) [1]; and the placement of fake resources, often referred to as honeypots, which would only be accessed by an insider [29] – though this approach only works when the insider is unfamiliar with resources in the organisation (this works well for cases of stolen credentials by poorly for genuine insider threats). Greitzer and Hohimer [12] propose the combination and analysis of multiple sources of data – including both computer monitoring data and human factors data [35] – we go further here to promote the use of Machine-Learning to analyse these multiple data sources for the identification of anomalous behaviour. Psychometric tests are used as part of the prediction model proposed by Kandias *et al.* [15] for identification of insider threats. In our approach we see psychometric as an additional data set which can be analysed by our Machine-Learning approach. Additionally other relevant human factors data – e.g. risk information produced by Human Resources (HR) – can also be included within our analysis. The use of Hidden Markov Models in order to detect divergence from normal user usage patterns was proposed by Thompson [32]. This approach looks at temporal patterns whilst our approach looks at point-wise patterns.

There has been substantial work conducted in determining if a user's credentials have been compromised. Davison and Hirsh [8] demonstrate patterns in users' commands on UNIX type systems and use these to predict anomalous activity. Although this does not directly map to the use of a Windows based GUI the same idea of patterns of activity could be applied as an additional approach to our own which looks at the set of activities, but not the relative ordering. Schonlau [26] observes the 'uniqueness' factor – that many commands are often used only by one user or rarely by more than one user – to infer when a different user is active in an account. Due to the nature of Windows where there are many routes to achieve the same objective (opening files from within an application or opening them from the file explorer) could again be used as a profiling approach. DuMouchel [9] uses Bayes factor statistics to determine if a set of commands matches the command transition matrix defined for a given user. Likewise this approach could supplement our approach. Shavlik *et al.* [21] monitor over two hundred Windows 2000 properties every second creating around fifteen hundred features. During training each feature is assigned a weight which then allows a score to be assigned to each 'feature' which votes on whether the system is being used by the correct person. Although this could be added to our approach we see it as having a significant impact on performance. Jha *et al.* [14] use Hidden Markov Models to identify patterns of usage and determine the person (either genuine employee or attacker) who performed the actions. Again this is computationally expensive.

Eberle *et al.* [10] demonstrate the use of a graph based approach for the detection of insider attacks. However, at present their approach can only be successfully applied to static situations and therefore would be difficult to apply in our context.

Mitigation techniques have been proposed by Pramanik *et al.* [25] which extends the access-control framework with a security policy defining subject, object, actions, rights, context and information flow as

applicable to the document control domain. Although as this requires extensions to the way the computer works we see this as too heavyweight for our system.

3 Background

Identifying a potential malicious insider among an organisation's employees is a complex task. The main challenges are that there are different types of insider threats, employees are not always malicious, and a trustworthy employee / malicious insider binary classification does not neatly apply. There are many human factors to be considered, ranging from issues that are internal to the subject concerned (such as personality, moral compass, contentment level, or change of ideology), to external factors (such as pressure at work or in the family, enticement of bribe, or blackmail and threats from a third party). People react differently when faced with the same situation or conditions – the same person might even react differently to the same situation at a different time. In other words, there is no solution that can provide a 100% certainty in identifying a potential insider.

Even though there are no silver bullets, we should still strive for ways to alleviate the threats of malicious insiders. Success is more likely to come from combining complementary techniques, including a better understanding of human behaviour. In this paper the focus is upon 'closed' organisations where contractual obligations are placed upon employees to follow strict information security rules. This reduces the complexity somewhat, as employees who are seeking to harm the organisation often achieve this through clear flaunting of the rules. However, this still does not necessarily mean that such employee is an insider, they might actually do this out of short-sightedness or as part of fulfilling their own ambitions – as we will see later.

Employees within an organisation can be classified within a two-dimensional space – of *actions* and *intent*, and both actions and intent may fall upon a continuous scale ranging from good to bad. Loch *et al.* [16] propose a binary classification for intent, being either accidental or intentional; however, we argue that this is a continuum, from 0% good (i.e. bad) to 100% good; likewise for actions. Nonetheless, it should be made clear that intentions are dependant on the observer. For example, a whistleblower may personally have good intentions, however, this would be seen as bad intentions by the organisation (and since the work presented in this paper is framed from detection of actions from the organisation's point of view, we adopt the standpoint of intentions as seen by the organisation).

It could be observed that employees motivated toward financial crime or espionage perform bad actions with bad intentions, whist well behaved individuals who abide by the rules perform good actions with good intentions and so reside in the opposite corner (see Figure 2 for a simple illustration of this classification). Unfortunately, employees can occupy any point within this space, which makes the task of identifying bad employees harder. It is also worth pointing out at this stage that there is an additional challenge whereby employees are likely to change their actions/intentions gradually during their time in the organisation, i.e. they are not static [2], even if they do not intend to cause any harm to the organisation. Therefore, there is a need to be able to adapt what is considered to be a normal pattern, for example by complementing the detection algorithm with some other means such as monitoring a wider coverage of the employee's actions (i.e. looking at historical data), and comparing their profile against a general profile for the role they are hired for.

Based on the action–intent classification and the work by Wall [35], we have identified six main categories of employees (as depicted in Figure 2):

1. **The well-behaved:** Follows the rules of the organisation and has positive intentions regarding

[2]In general, if someone is static then their actions will be learnt as their 'normal' pattern, and any departure from the normal pattern can then be marked as 'suspicious'.

their work, so both actions and intentions are >90%. This group is what any employer would wish to have.

2. **The negligent:** Breaks the rules without realising it with the intention of making life easier for either themselves or for others within the organisation. While they perform bad actions, their intentions are good. These employees may be susceptible to social engineering attacks because they lack an understanding of how their actions are detrimental to the organisation. Negligent employees tend to have <50% actions that are good, but their intentions are usually >50% good.

3. **The ambitious:** Intentionally breaks the rules in order to gain some advantage. As with the negligent, they have good intentions towards their work (>50%) but they perform bad actions (<50%). The distinguishing feature between the two groups is that the ambitious employee will actively and knowingly break the rules.

4. **The malicious insider:** Ignoring the specific motivation, this employee actively attempts to circumvent the organisation's rules. They perform bad actions and have bad intentions (<20%). It is this group of employees that we would like to be able to detect, in order to prevent damage to the organisation.

5. **The whistle-blower:** This is subtly different to the malicious insider, in that while this employee's intentions are bad (<20%) from the organisation's point of view, from their own point of view their intentions are good (>90%).

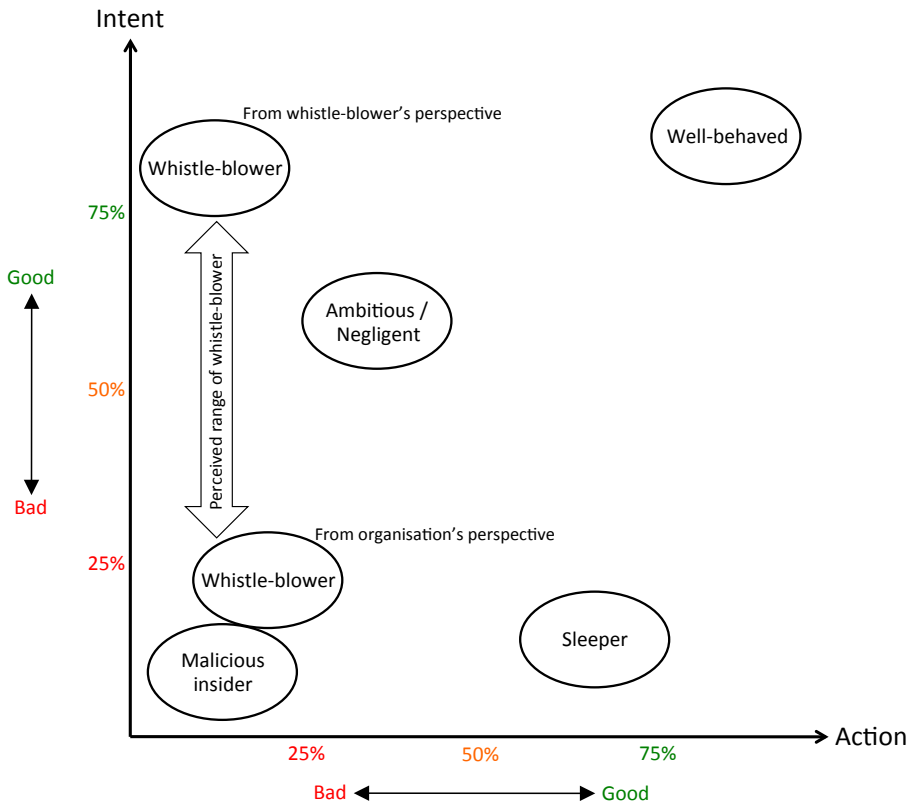


Figure 2: Categorisation of employees based on intent and action

6. **The sleeper:** This employee has bad intentions (<20%) although they are currently not performing any bad actions (>80%). They might be reluctant to carry out the bad actions (for example, due to the risk of being found out or because their desire to cause harm to the organisation is not yet strong enough).

This is not an exhaustive list, and there are bound to be other categories of employees within any organisation. Some types of organisation might have more than their fair share of certain categories, for example companies that promote ruthless working practices might harbour more ambitious or negligent employees than those that uphold more ethical approach. Nonetheless, the six categories outlined above provide a useful starting point for digging deeper in our effort to understand the dynamics of employees' actions and intents.

3.1 What Does An Insider Mean?

The complexity of insider threat detection when false positives are to be avoided is illustrated in three hypothetical cases covering a diverse range of threats:

- **Theft of data (whistleblowing/revenge)** – passed over for promotion multiple times by her employer, a prestigious clothing manufacturer, Joanne Ben-Nevis wishes to leave the organisation and go freelance. She is also strongly motivated to embarrass her employer by exposing the organisation's poor practices to the public. This requires the exfiltration of the many documents detailing the outsourcing of manufacturing to organisations in third world locations with poor working conditions. These practices are legal, but would seriously damage the commercial reputation of the organisation. Before resigning she spends three months slowly and deliberately downloading the files that she wants.
  Characteristics: Large data acquisition but over a (reasonably) long period of time.

- **Espionage (Trade Secret theft)** – Paul Penyghent intends to move to an organisation that is a direct competitor to his current employer. In preparation for the move he wishes to copy the contact details for all the existing clients and also take the design of a new device the company is planning to manufacture. He must extract all the data he needs before resigning as he will have no further access to the required systems after that point.
  Characteristics: Large data acquisition over a very short period of time.

- **Corruption of data (falsifying accounts)** – Ben Lawers has in the past made improper comments to a female co-worker and he believes this is a reason for his career stagnation. He wants to remove these events from his HR records and is waiting for his line manager, whom he sits next to, to leave her computer unlocked.
  Characteristics: The use of someone else's account to perform a malicious act.

In comparison, let us consider how some actions performed by an employee might be considered bad, but they should not be considered as insider threats:

- **Circumvention of procedure (Well-meaning)** – Jane Galtymore (CEO) finds the security policies at her company frustrating as it only allows her to work at her terminal in the office. She regularly makes copies of any files she needs so she may work on them while commuting to and from the office.
  Characteristics: Although breeching security, the acts are not malicious.

- **Circumvention of procedure (Negligent)** – Ben Lomond works as a secretary at the head office of a multi-national advertising company, and he handles calls from executives based at branch offices all around the world. One day he receives a call from a woman claiming to be the Head of the Hong Kong office asking for the detailed plan of a car advertising campaign that the company has been hired to do. Ben has never spoken to her before, but she sounds very convincing, so he sends the latest promotional portfolio to her. It turns out that she is actually someone from a rival company.
  Characteristics: In an attempt to be helpful and fulfil their job, an employee could become a victim of social engineering that damages the organisation – through negligence – even if they do not intend to cause this harm.

The scenarios presented above contain a mix of both intentionally malicious activity and non-malicious acts, but it is still likely that the companies involved will want to address the bad behaviours to prevent damage. Where the employees have positive intentions the company may provide training and guidance to prevent further infractions while employees with negative intentions will need different approaches to halt their bad actions.

The scenarios illustrate one of the key problems associated with the identification of an insider threat, that some acts are malicious while others are not, but in either case they are potentially harmful to the organisation. Falsely identifying and accusing an employee as being an insider threat could be very damaging and so it is important to include sources of data that are not purely technical such as human factors information including historical HR data, these may help to evaluate the threat posed by individual employees.

Creating a 'job profile' for each role that exists in the organisation might provide a quick way to spot if an employee behaves anomalously from their job specification. However, these profiles do not take into account the difference between different workers' patterns – for example one employee may be very willing to stay late and complete work while another will leave at the same time each day. Therefore an employee who leaves at the same time every day who suddenly accesses files in the evening has greater potential for 'badness' than someone who regularly works late, which is not necessarily true. As such, an approach that can 'learn' the usage patterns of individual employees is more likely to spot abnormal behaviour. Nevertheless, a weakness of an AI that learns behaviour is that it does not help in the short term, also if an employee decides that from the start of their employment they will exfiltrate a small number of files every day then the AI will eventually see this as their 'normal' work pattern. Likewise, relying only on usage patterns of each individual employee might lead to false-positives, for example when an employee changes roles (say, if Ben Lomond is promoted to a line-manager role, he will have a new privilege to access personnel records, which could be taken as an anomalous behaviour based on his historical usage patterns).

Ben-ware attempts to address these issues by allowing the AI to utilise human factors information to help it assess when threats reach potentially harmful levels. Organisations often already hold such information consisting of hard facts about individuals:

- *Human Resources data* may contain valuable information to give some indication regarding an employee's feeling towards the organisation. For example, someone who has been denied a promotion or a has disciplinary action noted against them is more likely to pose an insider threat than others who have been progressing well in their job.

- *Verifiable intelligence sources*, relevant personal knowledge held by a line manager can also provide an indication of an employee's likelihood to behave well. It is also possible that an employee might share their grievances with their colleagues (note that we need to be careful here that such

information is not created erroneously as part of someone's attempt to oust an employee that they do not like).

- *Personal characteristic information*, e.g. psychometric indicators (such as Belbin or Myers-Briggs test) may shed some light into how an employee might react in a specific scenario or under certain pressure. This can also be used to gauge the likelihood of an employee to follow rules or to behave in a way that would avoid harming themselves or their organisation.

- Determination of a employee's IT skills [17] to understand *individual's profile*. By understanding an employee's IT skills level, an organisation can gauge the risk of allowing this employee to explore various information systems owned by the organisation, or even how likely they will be able to cover their tracks.

- *Creating a general profile* of the individual's occupation. There are usually many employees holding the same role within an organisation, so it is useful to create a general profile for each role to act as boundaries within which such employees should operate within.

- *Suggesting an "exfiltrator profile"*, which can be used to determine whether someone has started to exhibit some worrying behaviour towards becoming an insider. These include low frequency access of many files (as compared with other employees' profile with similar role) or changes in working patterns (as compared to their normal individual profile).

The main difficulty lies in converting these pieces of information into something quantifiable that can be easily processed by the decision making system and the AI. Therefore we need to start with a simplified version, for example through defining 'five states of happiness' that each employee will belong to at a particular point in time. This ranges from very unhappy to very happy, with gradual changes between one end to the other. The choice of five states is arbitrary but reflects the fact that quantifying this value exactly is not possible. It is highly unlikely that someone will jump from one extreme to the other in a very short time, so a method of detecting gradual changes could be useful towards identifying a potential threat.

4 Our Vision: Ben-ware

The vision for Ben-ware is to provide a novel architecture that is able to detect anomalous behaviour with minimal impact on the existing computing and network resources in place – which may be comprised of legacy and low-performance computers – in terms of requiring no extra facilities, minimising the CPU, memory and storage footprint, supporting older operating system versions, resilience to unreliable and low bandwidth communications and, importantly, transparency to the users. Portable devices such as laptops and tablets are particularly prone to periods of isolation due to travel, therefore the architecture must support the storage of events for later transmission, while at the same time it should respect that the purpose of the network connections is to support the goals of the organisation by yielding network bandwidth to other applications unless significant security issues are detected.

The hierarchy of services in the Ben-ware architecture (Figure 3) consists of four types of component, these layers take detailed records of user activity generated by the lower levels and collate it into a profile of each user's activity so that that upper levels may compare it to the normal profile of each user. At the lowest level we find the Ben-ware Probes, this component resides on all hosts and contains the probes that monitor user activity on each host. Above the Ben-ware Probes we find the Mid-level Controller (MLC), this component runs on hosts containing sufficient resources (processing power and storage) and it receives activity data from one or more Ben-ware Probes instances. Its primary roles are to filter and

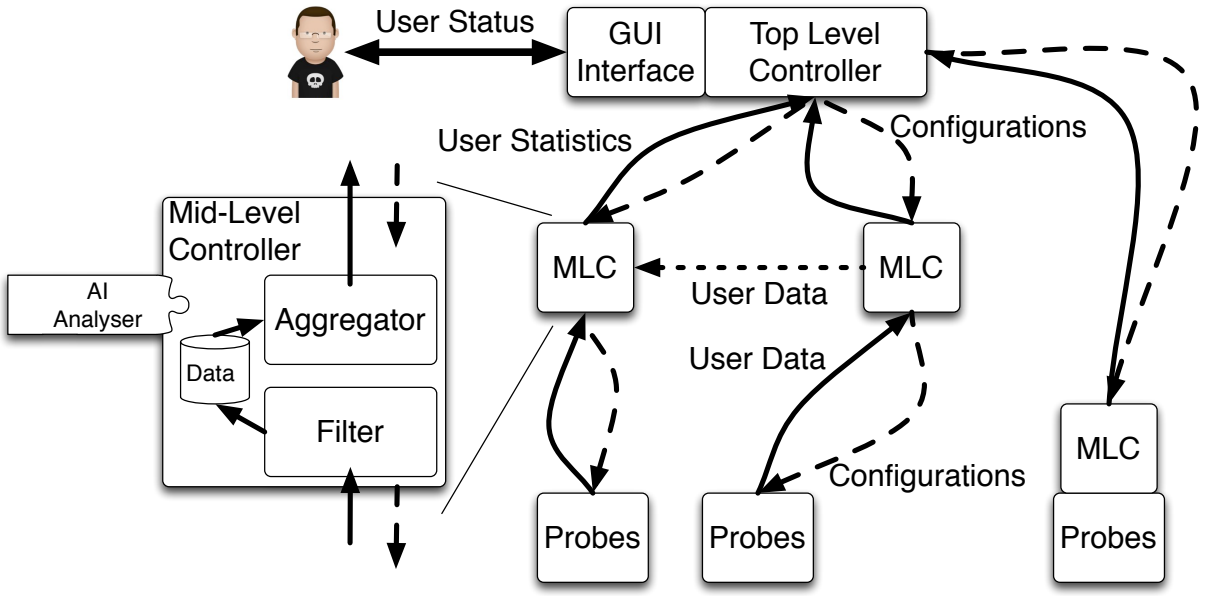


Figure 3: Overall Architecture

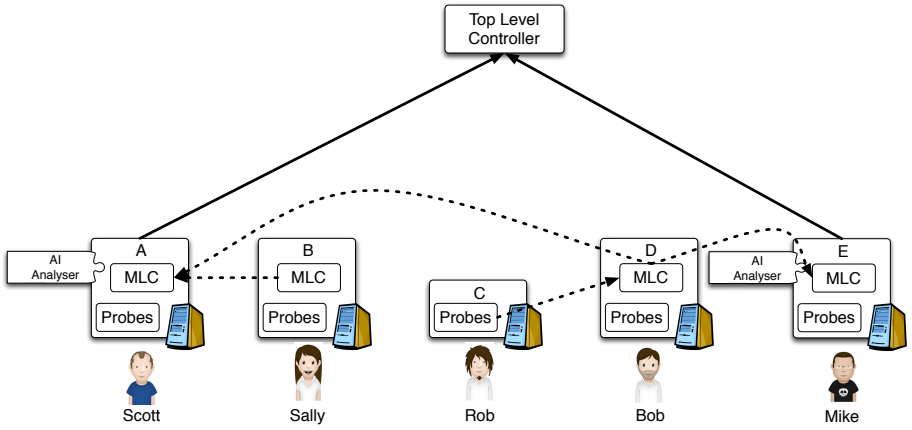


Figure 4: Deployment and communication chains

aggregate a user's activity data or to forward the data to another MLC responsible for that user. The AI can sit within a MLC and on top is the Top Level Controller (TLC). These components perform the roles of detecting anomalies in user activity and presentation of user activity summaries to a security officer respectively. While a single TLC is represented in Figure 3, there may be many of them permitting managers at different levels or locations to monitor their employees. If a host has enough resources then it may contain both an MLC and an AI instance. Figure 4 shows the paths followed by user event data in a small network of hosts. Here each host contains an instance of Ben-ware probes, host C does not have the resources required to host an MLC, so data it generates is sent to Host D. Similarly Host B can not maintain an AI instance and so the aggregates it generates are sent to Host A for analysis.

Ben-ware probes can capture information such as user logins/logouts, USB attachments, web access and file transfers, and this information is transferred up to an MLC for further processing. The Ben-ware probes consist of two parts, the probes themselves that actually monitor aspects of user behaviour and a probe manager that is responsible for instantiating the probes and handling communications, Figure 6.

This separation of concerns and the definition of a minimal interface between the probes and the probe manager means that new probes may be developed and added to Ben-ware as new threats or monitoring techniques are identified. The MLC contains filters and rules that determine when to aggregate data, when to forward it to another MLC and when and how long to store the data for potential future forensic use. The purpose of the filtering and aggregation is to reduce communications overhead and allow prioritisation of what data should be sent to the AI/TLC immediately, making Ben-ware adaptable to low bandwidth, intermittent or partitioned networks. Messages sent direct to a TLC will alert the human manager to a potential security threat.

We identify three deployment scenarios (Figure 4):

- **Machine with Probes only (C):** This may be a low-power/legacy computer incapable of running the full service
- **Machine with Probes and MLC, without AI (B and D):** A more powerful computer that is capable of running the MLC. In this case the computer may be powerful enough to run an AI though there may not currently be a need for running one
- **Machine with Probes, MLC and AI (A and E):** This is a powerful computer running all three services.

We have previously highlighted the issue that with imperfect networks and portable devices, experiencing a loss of connectivity to the organisation's network is a situation that Ben-ware must treat as common place. In addition to the planned and unplanned network disconnections there is also the effect of employee mobility between organisation sites or between location that are on different and disconnected network segments. These situations require different mechanisms to ensure the continuity of the Ben-ware functionality. The case of the an employee moving between sites is illustrated in Figure 1, here an employee, Jo, normally works at site A, and so his activity is normally processed by an MLC at site A. On occasion, Jo works at site B and while there it is still required that Ben-ware tracks and analyses his activities in a timely manner. Ben-ware does not know how long Jo will remain at site B and so it can not store the event messages until he returns to A, so the strategy Ben-ware employs is to use peer-to-peer communications to send event data laterally across the Ben-ware hierarchy. In the example, when Jo works at site B his event data is propagated from the workstation he is using up to the site B MLC. As Jo is not designated as a user the site B MLC acts as a communication intermediary, using a DNS type lookup to determine which MLC on which network Jo's data should be forwarded to. The site B MLC then proceeds to forward the events generated by Jo's actions to the MLC at site A so that a global view of Jo's activity may be obtained. The global view of a user's activity is necessary if Ben-ware is to detect suspicious activity of a user that is attempting to avoid detection by spreading their activities geographically.

There is also the scenario of an external computer, not owned by the organisation, coming in through a VPN connection (Figure 5) or a Bring Your Own Device (BYOD) policy. In this case, there will be no Ben-ware probes on the external devices and so other monitoring opportunities need to be taken. Here Ben-ware would need to be extended in two ways. In the first instance Ben-ware would need to reside on network hardware such that it may report upon external users opening and closing VPN connections along with any potentially suspicious activities such as scanning the network. Secondly the Ben-ware that resides on individual workstations and file servers would require new probes developed that are able to detect the opening of a share drive and opening or copying of files. Such probes would report these external user evens back to the local MLC which would then forward them to the appropriate MLC for aggregation to produce the global view of the external user's activity. This scenario is not considered further here.

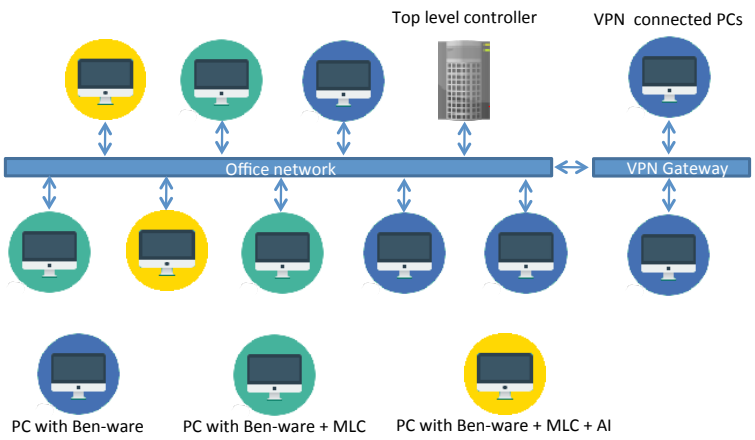


Figure 5: A network with a VPN connection

The AI has the ability to make modifications to any filter and aggregation rules for the users it is responsible for and it may send these updates to the relevant MLCs. The rules determine, for each type of user activity, when the information should be propagated up and the parameters for the aggregation algorithm. Thus if an AI determines that a particular user never logs in until after 8am, then a login event detected before this time should be immediately forwarded up the Ben-ware chain, but login events during what the AI has determined is their normal hours, should be aggregated and the aggregate sent once per day. This permits information prioritisation and reduces network impact.

A user's AI will be placed close to their normal working location, where close is function of the speed of the network between the AI/MLC and the user's workstation and also takes into account the processing power required to run the AI[3]. If the network becomes partitioned such that the MLC/AI for the user is not reachable from the user's workstation, then the event message generated by the Ben-ware probes will be stored until the network connection is restored. If however, a significant time elapses, where significant is considered on the scale of the period of aggregation for the event data, then a new, reachable MLC/AI will be selected for that user and the event data will be processed there. The result of this is that there can be more than one MLC/AI allocated to a user when the network becomes contiguous and in this case one of the MLC/AIs will become the master and receive all data from the other for aggregation as a whole.

The MLC stores the event data generated by its associated Ben-ware probes in a database. The data is retained in the database for a period of time and is destroyed after this time elapses. Retained data may be examined in detail to support a forensic analysis of any security breach or when an alert is raised regarding a user to determine in detail what activities the user has been performing.

The Ben-ware AI (described further in Section 5.4) is responsible for detection of anomalous user behavior which may be an indicator of an insider attack. As previously discussed, a user could attempt to train the Ben-ware system by performing 'bad' actions from the outset of their employment, thereby making the bad behaviour appear normal for them. Nonetheless, such an attempt may be detected by other means including monitoring by a supervisor or comparison of detected behaviour with general role based activity profiles.

[3]The selection of the computer on which to run the AI is a much more complex problem, compounded by locations of other AIs and a user having multiple 'normal' computers.

5 Implementation

In order to test our conceptual model we have implemented a prototype of Ben-ware written as a set of interacting Java programs which run within and monitor a Windows based operating system – supporting Windows XP, 7 and 8. In this section we discuss the events which are sent between the different components before discussing each of the components in more detail – those of probes, Mid-Level Controller (MLC), the Artificial Intelligence (AI) and the Top Level Controller (TLC). The section concludes with a discussion of the process used to generate synthetic user logs used for testing.

5.1 Events

Data is transmitted between components using events, these events follow a common pattern between each layer of the system or between components at the same level. We first define here the event types as these are common between all levels. Following this up with a discussion of the different event message formats used between components. Table 1 lists the event types that we have defined as part of our prototype. Each event type is coded through an ID which comprises of a major id, minor id and context. The major id indicates the broad category of event type – such as the computer changing state (1) or a USB device action (6). The minor id indicates the type of action within this category. While the context indicates the context under which the action has been performed – for example 4.1.1 would indicate a file copy while 4.1.2 would indicate the copy was to a USB device and 4.1.3 would be a copy from a USB device. By providing a categorisation we not only provide a mechanism for future expansion but we also allow quick identification of action type – Logon / Logout events are closer in relevance to each other than they are to file manipulation operations. While an internal (to the computer) file copy would be less significant than a copy to a USB device.

ID	*Name*	*Description*
1.1.x	Boot up	The host boots up
1.2.x	Shut down	The host is shut down
2.1.x	Heartbeat	The system is still active
3.1.x	Logon	User logs into system
3.2.x	Logout	User logs out of system
4.1.x	Copy	File copy (internal, USB)
4.2.x	Create	File creation (internal, USB)
4.3.x	Move	File move (internal USB)
4.4.x	Delete	File deletion (internal, USB)
4.5.x	Modify	File modify (internal, USB)
4.6.x	Open	File open (internal, USB)
4.7.x	Close	File close (internal, USB)
4.8.x	Rename	File rename (internal, USB)
5.1.x	HTTP	HTTP request performed
6.1.x	USBInsert	USB Insertion
6.2.x	USBRem	Removal of USB device

Table 1: Event types

Field	*Name*	*Description*	*Format*
1	Event ID major	Description of the event class	integer
2	Event ID minor	Description of the event sub-class	integer
2	Priority	Indication of message priority, 0=low, 9=high	integer
3	Time Stamp	Time of the event using the 64bit epoch time	UNIX epoch
4	Host IP	The IP address of the originating host	IP string
5	User ID	Logged in user ID	string
6	Host OS	Operating system name	string

Table 2: Ben-ware message common header

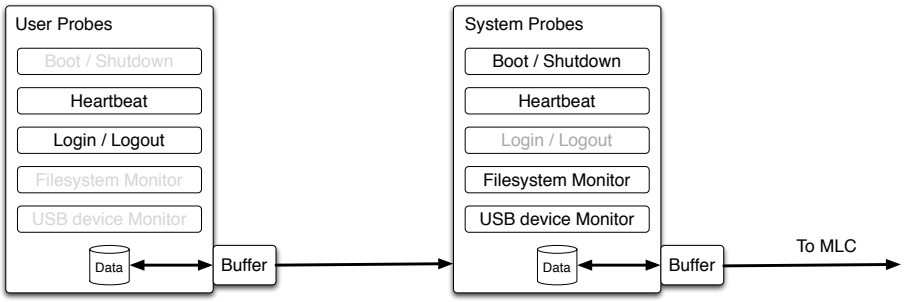


Figure 6: User and System Ben-ware probes

5.1.1 Event format from Probes

Here we discuss the format of event messages generated by probes. Each event message is constructed as a comma separated list of elements on a line. The line contains two elements an initial common header – as described in Table 2 – followed by an optional set of comma separated values which depends on the particular event type. For example, *Rename* will contain a copy of the original file name and the new name whilst *HTTP* will contain the URL. Priority is used by the components to determine how important a particular event is – for example when dealing with a low bandwidth network connection high priority events will be communicated before low priority events. Time stamps are represented as the number of seconds since the 1st January 1970 (commonly referred to as the UNIX epoch), while the Host IP is stored as a string comprising of four numbers (in the range 0 to 255) separated by full stops. User id's are the standard login names used on the system (it is assumed that there is a common naming system used across all computers) and Host OS is a short string indicating the operating system and major revisions – e.g. 'WinXP SP2'.

It should be noted that probes generate single line event records which are sent to higher components. However, these event records may pass through other probes and / or multiple MLCs before reaching the MLC responsible for the user. In these cases multiple event records can be concatenated together. The event type specific data is appended to the end of the common header as comma separated values. The extra data fields for those events which require them are listed in Table 3. In this table x can be one of 1=internal to internal, 2=internal to external, 3=external to internal and 4=external to external, where internal is part of the systems internal file system and external is an external file system (currently USB). Likewise y indicates the web request protocol 1=HTTP and 2=HTTPS. Note that for the events Boot up, Shut down and Heartbeat as these are system level events the user field is set to null.

Field	*Name*	*Extra fields*
4.1.x	Copy	Path to source (string), path to destination (string)
4.2.x	Create	Path to new file (string)
4.3.x	Move	Path to source (string), path to destination (string)
4.4.x	Delete	Path to old file (string)
4.5.x	Modify	Path to file (string)
4.6.x	Open	Path to file (string)
4.7.x	Close	Path to file (string)
4.8.x	Rename	Path to source (string), path to destination (string)
5.1.y	HTTP	URL (string)
6.1.0	USBInsert	Device type (string), name (string), unique id (string)
6.2.0	USBRem	Device type (string), name (string), unique id (string)

Table 3: Event specific data fields

5.2 Ben-ware Probe

In order to identify the activity performed by users upon resources we have implemented the concept of Ben-ware probes. Probes are provided through a generic framework in which pluggable probe instances – used for monitoring specific activity – can be installed. Due to the fact that certain information can only be obtained from specific running contexts – a consequence of the Windows security model – multiple probes may be active on a machine at the same time. In which case there is one 'master' probe which collates all of the different probes data together before sending it to the Mid-Level Controller. Figure 6 depicts the case used for our prototype in which two instances are running. In both cases only a sub-set of the probe instances are active.

The master probe, which runs at all times, handles the system-wide events such as power up/down and USB insert/removal, whilst the second probe is responsible for the user and is only activated as a user logs in – and terminates as the user logs out. As part of the boot up process Windows will initiate the master probe – referred to as the 'System' probe in Figure 6. The system probe is responsible for monitoring the underlying operating system. Probes have been developed to indicate when a Windows instance boots up and when it is correctly shut down. As an attacker could try to mask their trail by forcing the computer to crash or by physically powering the computer down a heartbeat is sent to the MLC at regular intervals – say every five minutes. Alternatively they could kill off (or suspend) the Ben-ware software whilst performing their illicit activity. By having these regular heartbeats the system can be aware of the fact that something happened to the normal running of the Ben-ware probes.

The system probe also provides file monitoring and detection of USB insert and removal events. In terms of file monitoring the probe is able to listen out for file change events produced by the operating system – such as open, close or move. For optimisation purposes this is restricted to a set of sub-trees from the entire file structure – we assume here that the user only has write access to these particular sub-trees. This allows us to determine file creation, move, copy or deletion events. It should be noted that high-level programs such as Microsoft Word generate a number of file events when performing actions such as file save. However, the pattern of these file events along with the file names produced follow a fairly regular pattern. This allows for the identification of such high-level application actions. On insertion of USB filing systems a USB probe will inspect the contents of the USB drive and add this to the list of sub-trees for monitoring. The USB probe will then regularly check that the USB device has not been forcefully removed (pulling out without a dismount) whilst also listening out for dismount events.

The restrictions of the Windows security model prevents the system probe from being aware of when a user logs into the computer or obtaining any data on the logged in user. In order to overcome this

problem a second probe instance is initiated as part of the normal user login process. This is the 'User' probe in Figure 6. This probe is able to capture information about which user has logged in and the time of the login. This data can be forwarded to the Master probe for sending up to the MLC. As a system crash, physical power-down of a computer or termination of the user probe would prevent the user probe from completing its function it also generates heartbeat messages which are forwarded to the master. As sending these messages up to the MLC would just duplicate the heartbeat messages coming from the Master these messages are discarded by the master probe. However, if these messages are not received then this will trigger a new event from the master probe to indicate that a secondary probe had failed.

The probes are configured using a configuration script allowing definition of such factors as how often heartbeat messages will be performed and which sub-trees of the filing system should be monitored. If the master probe is unable to send messages to the MLC these are cached within a flat file for transmission at the earliest time of re-connection. If the probes are re-started any unsent messages will be transmitted to the MLC. In so doing the probes can recover from crashes and cope with network failures.

5.3 Mid-Level Controller (MLC)

The Mid-Level Controller performs a number of data movement and aggregation operations. There is one MLC within the system for each computer. This will be placed on the computer 'closest' to the computer being monitored – ideally on the same computer – subject to computational and storage requirements. All events captured about a particular user will be stored on a single MLC responsible for that user. The MLC selected for a particular user will be chosen as the 'closest' MLC to the users normal computer of use. Events relating to a given user will be forwarded on to the MLC responsible for that user. If the network becomes segregated for a period of time (normally longer than an aggregation cycle) then a new MLC will be allocated for that user on the part of the network unable to contact the primary MLC. Once the network comes back together then the new MLC will forward all data onto the primary MLC. It should be noted here that as our primary interest is in the identification of anomalous activity by users, the system events – namely power up, power down and heartbeat – are not considered further unless they show evidence indicating inappropriate activity occurring on a computer which may indicate a user trying to subvert the Ben-ware probes.

Three components make up the MLC, those of: server, database and aggregator as depicted in Figure 7. A fourth component – The Artificial Intelligence – is an optional part of the MLC and is only present in the case where the processing power of the computer hosting the MLC (which would fit into the jigsaw space) is high enough to support it. The server component will listen on a specific TCP port for data arriving from the system probes[4]. The incoming data will be checked for consistency – receiving data which is incorrectly formatted or showing events in impossible orders (e.g. a file being opened on a computer where no user is logged on) would suggest that a probe is malfunctioning or being compromised. In either case this warrants further investigation. Events which relate to a user which is linked to this MLC will now be added to the local database else a DNS-like service can be interrogated to identify which MLC is responsible for this user and the data forwarded. We have used here the open-source MySQL database server which allows long-term storage of the data received from the probes along with the ability to perform complex queries over this data[5].

The aggregates of the data collected over the last n minutes (by default $n = 1440$) is produced by the aggregator through queries against the database. At present our aggregates are just the total of each type of event over the last n minutes. However, other aggregates could be produced such as averages, max or min. For those event messages which contain extra fields these need to be translated into values that can

[4]Note that for efficiency this could be done as an in-memory data transfer when the probes and MLC are on the same computer.
[5]For performance of a real deployment the MySQL database could be replaced with a more compact and less performant service.

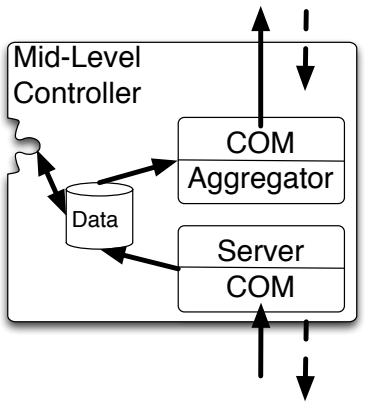


Figure 7: MLC architecture

Field	*Name*	*Description*
1	Event ID	A numerical description of the event
2	Date/Time	Date and time of the final data point
3	User ID	Logged in user ID
4	Host IP	The IP address of the originating host
5	Level	Level of security
5	Aggregate	Aggregate of metric
6	Time Span	Time duration covered by the measurement

Table 4: Mid-Level Controller to AI message structure

be used for summation. File names need to be converted into a notation about how significant the file is to the organisation – High-Sensitivity, Medium-Sensitivity, Low-Sensitivity, Unclassified. We assume here that the organisation has some mechanism for labelling their files either through location, naming conventions or a separate lookup table. Likewise URLs need to be classified in terms of their risk to the organisation – Approved, Benign, Inappropriate or Banned. Similarly, it is assumed that the organisation has a mechanism to classify sites into these categories.

It should be noted that although aggregates are only sent to the Artificial Intelligence once every n minutes the aggregator will normally run every m minutes ($n > m$ and normally $m = 60$). This allows for detection of specific thresholds being exceeded. For example a user may have a limit of no more than five USB inserts per day. If this threshold is exceeded it is flagged up to the Top Level Controller (TLC) no more than m minutes after the threshold is exceeded. This allows for faster responses to unusual activity.

Communication with the AI is performed via a TCP connection[6]. In all cases if a connection with the AI is not currently possible then the messages will be buffered until a communication is possible. Aggregates to the AI are sent as a set of comma separated lines as defined in Table 4. Where Event ID is as defined in Table 1, Date/Time is a UNIX epoch, User ID is the login ID (assumed to be consistent across the organisation), Level is the level of security, if appropriate – for example High-Sensitivity, Medium-Sensitivity, Low-Sensitivity, Unclassified – Aggregate is a count value and Time Span is the number of seconds since the last event Date/Time for this Event ID and Level combination.

As well as receiving events from the probes the MLC also receives updates from the AI for re-

[6]This is due to the fact that the AI need not be local to the MLC. However, if the AI is local then in-memory communications could be performed.

Field	*Name*	*Description*
1	Event ID	A numerical description of the event (see Table 1)
2	User Set	Set of user id's
3	Security Level	Security level this applies to
4	Operator	Operator to apply
5	Value	Value to apply

Table 5: AI rules

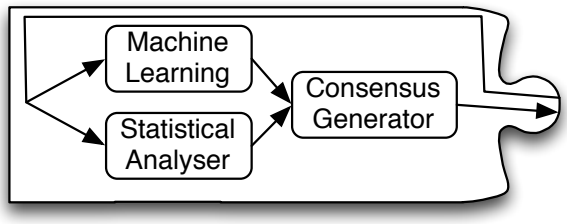


Figure 8: Overview of the intelligent agent

configuring the threshold rules. These AI rule-sets define thresholds for a particular event (and potentially security level) which can happen in a given time interval (n) before the breech should be reported up to the TLC – based on the aggregation performed every m minutes. The structure of a rule is a tuple as defined in Table 5. For example a tuple of $\{4.3.2, \{Ben, Jo\}, Medium\text{-}Sensitivity, >, 10\}$ indicates that if more than ten Medium-Sensitivity files are copied from the system to a USB device then this should be immediately flagged up. This rule should be applied to both users Ben and Jo. Note that if any threshold rule is breached then they should (all) be flagged up.

When initiated the MLC reads a configuration file which contains the required data to access other services such as the TLC, AI and database along with an initial set of threshold rules.

5.4 Artificial Intelligence

Anomaly detection is provided by the AI component. This component produces a model of the normal behaviour of a given user which can then be used to determine if that user is acting against their normal user profile. This can be achieved through a combination of statistical algorithms and machine learning – thus producing an intelligent agent. The two approaches are first applied independently to the data received from the MLC before being combined together to form a consensus – see Figure 8. This consensus is then mapped to a risk score.

The inputs to the AI are those as defined in Section 5.3. We assume here that the Human Resources score for a member of staff is provided by some other means and for this initial prototype is one of the five levels of happiness which is provided on a fortnightly basis. All of this input data can be used to generate the features required by the anomaly detection algorithms. The feature set includes those listed in Table 6.

By analysing these metrics we can develop a profile for a particular user's normal activity pattern, it is then possible to detect abnormalities as these will be be significant divergences from this normal profile. This set of features is by no means exhaustive nor is the set of probes that can be developed into Ben-ware. Additional probes and associated feature sets can be developed into Ben-ware to allow for the monitoring of other types of user activity.

In order to eliminate distortions due to features of different degrees of variability all features were

Feature set	*Identified by number of:*
Logon	individual logons, different hosts logged on to, out of hours logons (based on user's normal profile), logons to user's normal computer and logons on shared/other computers
External Storage Devices	USB device accesses, out of hour device accesses, different hosts used for device accesses, device accesses on own computer and device accesses on other computers
File Use	file accesses, file writes, file writes to external devices, sensitive file writes, sensitive file writes to external devices, out of hours file accesses, out of hours file writes, out of hours file writes to external devices, out of hours sensitive file writes, out of hours sensitive file writes to external devices, computers used for file accesses, file accesses on own computer and file accesses on other computers
HTTP	http requests, out of hours http requests, http requests on black-listed / categorised sites.

Table 6: AI Feature sets

normalised using the 95th percentile. We chose the 95th percentile as this helped avoid normalisation by outliers. Other normalisation approaches – including maximum value, median, mean and 90th percentile – were evaluated, however, the 95th percentile produced the most consistent results.

A number of different approaches to to machine-learning can be deployed here, with the choice being influenced by the type(s) of anomalous behaviour we seek to identify and the feature set which is available. As the number of actual insider acts which have been observed in organisations is extremely small, the number of observed potential threats is also very small, the fact that there exists examples of potentially suspicious-looking activity within most organisations which need not indicate actual internal miss-behaviour and the fact that there exists very little (if any) labelled examples of internal miss-behaviour makes the use of conventional (labelled) categorisation approaches unsuitable.

Instead we make use of a one-class classifier: the Support Vector Data Description (SVDD) [24, 31] as our machine learning algorithm. SVDD is similar to the Support Vector Machine (SVM) [34] approach; whereby a minimum bounding hyper-sphere (containing most of the observed values) is determined from the training set. In Figure 9 we show two examples for these boundaries. Case (a) is where a compact bounding can be achieved – in which case all sample points can be fitted within the hyper-sphere, whilst for case (b) a number of outliers exist where if we wished to include these within the hyper-sphere we would need to make this cover almost all of the volume. One of the advantages of this machine-learning approach is that it only requires the fine-tuning of one parameter during training – that of the maximum fraction of samples which can be outside of the hyper-sphere. The simplest boundary which can be obtained using this method is a hyper-sphere, however, kernel methods can be created for constructing more complex boundary forms – which may be more closely bounded to the data. For example, a kernel method based on the radial basis function. Though this requires adjustment of another parameter – the kernel width. However, in situations where we are using relatively small datasets this can lead to over-optimisation of the configuration leading to over-fitting. We can avoid this over-fitting by applying a cross-validation method.

In general a user's activity can be bounded within some region – consistent users will have small bounded regions whilst other users who are less consistent in their activity will have larger bounds. In general though a potentially malicious user can be identified by a number of new samples which exist

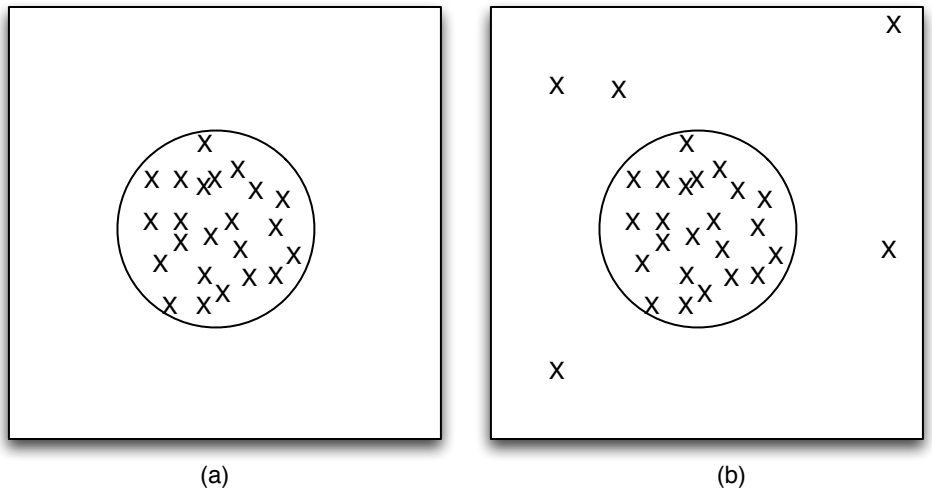


Figure 9: Bounding circle for a set of points

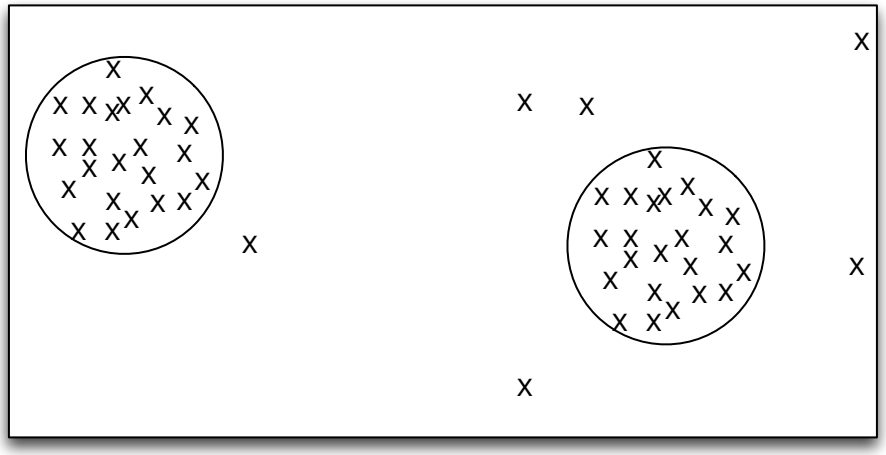


Figure 10: Clusters of user behaviour

outside of their bounding. A balance here is needed between the number of outliers and the severity of the outlying points. A significant outlier could indicate a significant miss-behaviour such as stealing a large number of files, whilst a large number of minor outliers could indicate slow exfiltration of files. It should be noted that the profiles of users will change over time. This could be as a natural consequence of becoming more efficient at their work or through promotions or job-reallocation. It is therefore necessary to re-train the AI at regular intervals.

One of the key observations made as part of this work was that the normal behaviour patterns of a user need not be consistently the same over a working day. We assume this to be a consequence of performing multiple roles simultaneously within an organisation. Therefore it may be more effective to construct multiple boundaries to enclose different 'profiles' (Figure 10). A clustering approach, such as K-means clustering [13], can be used to obtain clusters of different behaviour and then SVDD can be applied to train a separate classifier for each cluster. A significant improvement was obtained using this modification and to the best of our knowledge this approach is a novel extension for the problem of anomaly detection. However, a full treatment of this approach is not possible within the space of this

paper.

Our algorithm can be used with different combinations of features to identify different insider threat scenarios. At present we have trained our algorithm to capture scenarios which relate to the 'stealing' of files, though our approach could easily be re-appropriated for other forms of threat. A cross-validation method was used during feature selection phase. This incrementally adds features, retaining those delivering performance improvements leading to the optimal set of features to use. The retained features were those of: the number of file writes to external devices, total number of sensitive file writes, number of sensitive file writes to external devices, number of out of hours file writes to external devices, number of out of hours sensitive file writes and number of out of hours sensitive file writes to external devices.

The statistical analyser was implemented to use the same set of features as indicators. The main difference being that all features were used within a one-dimensional space as opposed to the multi-diminutional space used by the machine-learning algorithms. Each feature value was compared to the percentile value, calculated from the training data set, allowing a score to be assigned based on the amplitude of the indicator. Both the accumulated amplitude values and the duration of continuous behaviour were considered, thus allowing us to capture weak signals (for example if someone was stealing just a small number of files each day over a prolonged period of time).

The anticipated threat level for each user can then be passed on to the TLC along with any indication that the user has performed a misbehaviour. It should be noted here that although the AI may indicate that a user has breeched some rule of the organisation this cannot be used on its own as confirmatory evidence. Further conformation should be sought by a human operator who can inspect the evidence for act and intent.

5.5 Top Level Controller / User Interface

A user interface for Ben-ware has been developed in the form of a Top Level Controller (TLC). We have developed a prototype user interface specification as part of our development work. However, at present we have not implemented the TLC. The TLC can provide a broad overview of the system to an operator – who is assumed to be a line manager or computer administrator. Although we present here the notion of just one TLC we assume that in any large organisation there would be multiple (potentially overlapping) TLCs and potentially one super-TLC which was aware of all users. The operator is presented with a broad overview of those users that they are responsible for, with each user being marked with a traffic-light coding of their perceived level of threat to the organisation (green representing low threat whilst red representing a high level of threat). Figure 11 illustrates how the different interfaces can be organised for the TLC, where two main interface areas exist – those of system-wide configuration and monitoring of staff.

The TLC will hold the current state of each staff member within the organisation (subject to the time delays incurred from only processing normal data at regular points in time). This allows an operator to, at any time, investigate the current status of a user. This may be to follow-up on a concern or just to monitor the activities currently being performed by the user. As this could provide significant intrusion into the employees' personal space the set of operators who can perform this should be kept to a minimum and logging of the reasons for observations should be recorded – potentially through a pop-up dialog box when the request is made. Details of a particular (identified) threat can be investigated as well as viewing of the collected aggregates for the user which prompted the system to flag up a potential threat. If appropriate the operator can mine further into the raw data which has been collected about the specific user. This would give the operator access to the complete log traces for that user and additionally access to a coverage map of those files within the system that the user has been accessing. For ease of use all of this information can be presented as tables or rendered graphically. A configuration interface is also built into the TLC. This allows the operator to reconfigure how the Ben-ware system operates, such as the

time between aggregates being generated and the thresholds for moving between different traffic-light states.

We present here a number of user-interface mock-ups in order to exemplify our proposed user interface. Figure 12 shows the initial screen for the TLC, which allows rapid identification of potential threats within an organisation. Each user is labelled with a traffic-light indication of their current status with users being ranked from highest (potential) threat downwards. This is listed along with an actual threat score and thread delta – how the user's threat has changed since the last period (day).

The details of a user can be obtained by clicking the user's name in the main display. Figure 13 illustrates the threat scores for a particular user. This interface would allow the operator to identify the user and their role along with delving into the aggregates used to produce the threat score or even potentially examine the individual events captured from the users activity. There is also an additional tab which allows the operator to observe the coverage of the dataset that the user has accessed. By providing a quick graphical view of the files within a system, and which ones the user has accessed, it is possible to get an idea of whether the user is systematically collecting files from the organisation or only accessing a small subset of files relevant to that users work. Figure 14 illustrates how the individual events, captured through the Ben-ware system could be presented to the operator. This collected data – along with the aggregated data – can aid the operator in their initial investigations into a particular user's activity. Either allowing the operator to determine that the actions are benign or identifying a potential attack.

5.6 Synthetic Employee Generation

Due to the problems inherent in deploying Ben-ware within a real working environment for the collection of data: problems with having a robust and reliable software base; requirements to monitor staff for many months; ethical considerations of such data capture; along with factoring in insiders, it was decided to generate synthetic logs of user events for testing our prototype of Ben-ware. In order to simulate realistic

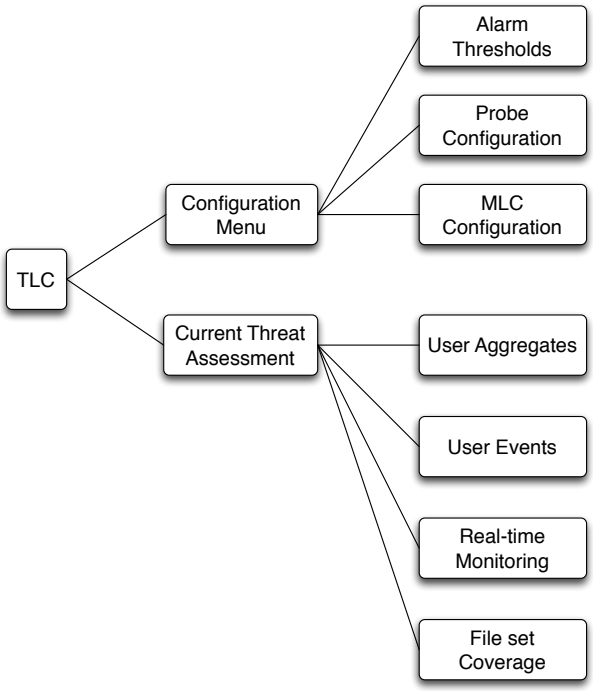


Figure 11: Top Controller organisation

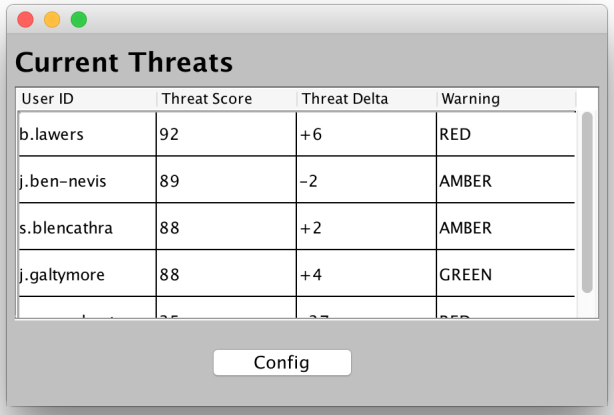


Figure 12: Initial screen showing current threats

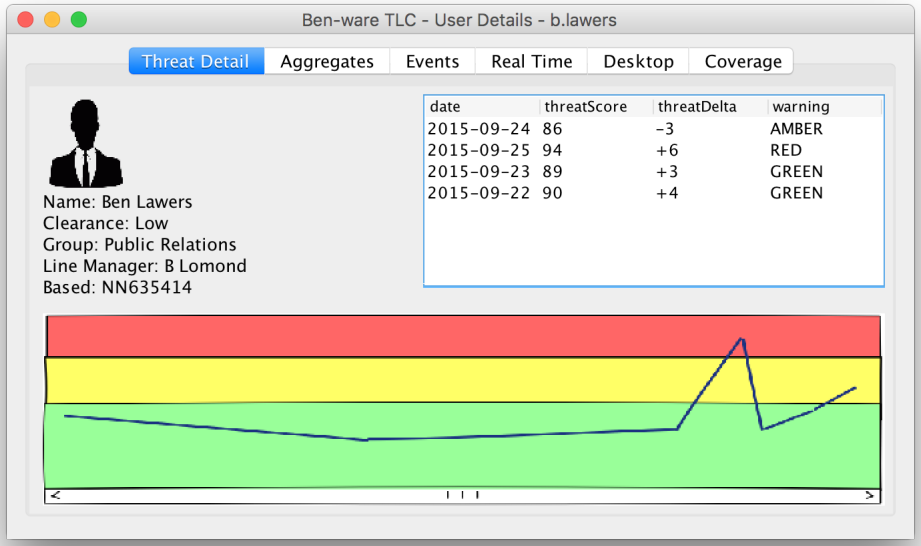


Figure 13: Details of a user over recent days

user interaction within the system we assumed that each user could cary out at any given time one of a set of activity types with the probability of moving from one activity type to another being defined through a Markov process. Below we will first develop a model for the environment and the users within this environment before presenting the development of a Markov model. Finally we will define our fictitious users who will be used as our test subjects.

Our fictitious environment can be defined based on the following assumptions. The organisation is modelled as a typical office environment in which the majority of employees will be present at their desks

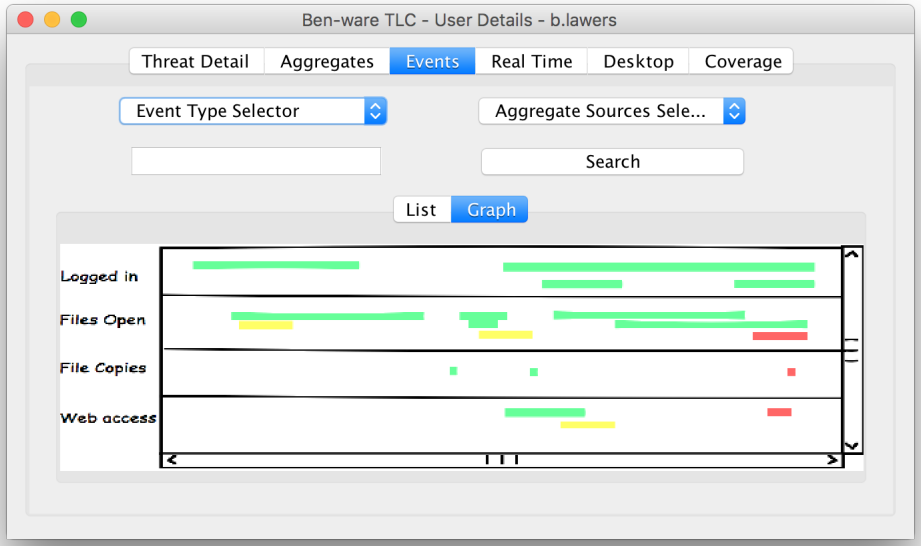


Figure 14: Individual events on a time-line

each day. Most employees will follow a normal working day (i.e. 9am till 5pm) however, employees may choose to come in early or leave late. The working week is Monday to Friday with no work taking place at the weekend. At present we do not model employees taking holidays. Employees are entitled to a lunch break which they will take between 12:30pm and 1:30pm. They are also entitled to a small number of short breaks which they can take at any point during the day. At the end of the day an employee may choose to leave the current task to continue the following day, stay until the task is completed or take the task home with them. This will be determined by a set of probabilities unique to that employee. If an employee takes a task home with them they will load up a memory stick with the required files just before leaving work and will copy these files back onto the system at the start of the next day.

Employees will attend meetings during the course of the day with each employee having their own probability for attending meetings. Meetings can only happen between 9am and 5pm, will always last one hour and always start on the hour. It is assumed that if an employee has to arrive early, leave late or change their lunch break they will do this in order to attend the meeting. In meetings employees will not make use of the computer systems.

It is assumed that there will be employees with different job roles within the organisation. One of the employees is a 'chief executive' type of role (e.g. may work unusual hours and choose not to follow some guidelines) while another will be a Personal Assistant to the Chief Executive.

The computers are running various versions of Microsoft Windows operating systems – a mixture of Windows XP and Windows 7 computers will be present. The security of the machines is assumed to be reasonably open – for example USB ports and CD/DVD writers will not be disabled. Networking between computers within the organisation are assumed to be always available and running at 100Mb. Internet access will be fully open. However, the organisation is assumed to have a strict policy on which sites may be accessed. Sites are therefore categorised as being one of the following: Approved, Benign, Inappropriate, or Banned. Each document within the system will have a name which ends with one of the approved security levels – those of High-Sensitivity, Medium-Sensitivity, Low-Sensitivity and

Unclassified – for example `myFile_Medium-Sensitivity.txt`.

As our intention here is to capture the handling of file interaction with the system the actions of employees will be based around the computer-based tasks they perform. These can be categorised as:

- **Administration:** Normal administration tasks performed within an organisation. Actions are related to other employees. E.g. approving holidays, approving time sheets.
- **Personal Administration:** Administration done by an individual for themselves. E.g. filling in expense claims.
- **Reading:** The reading of existing documents. May require the opening of other files but will rarely require the writing to or the production of new files.
- **Writing:** Essentially producing a new document. May require the opening of other files and surfing the internet for information.
- **Converting:** Taking an existing document and converting it in some manner. This may be summarising an existing document or 'declassifying' a higher security document. Security levels can go up or down for the new file. The user will open at least one other file and may open others and surf the internet.
- **Surfing:** The user is surfing the internet. File access will not normally be present.

Figure 15 illustrates the Markov transitions which are possible. It should be noted that re-entering the same action is not allowed. Figure 16 gives an example Markov transition matrix. It should be noted that these matrices are specific to each employee.

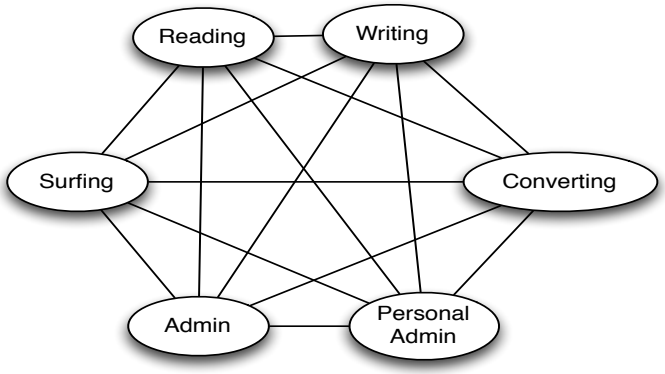


Figure 15: Interconnected actions

The construction of a given employees day builds up from first determining the preferred start and end times, allocating the times of meetings and lunch breaks[7], selecting the actions to be performed along with there durations, and then finally determining the events which will happen during each of the actions and the timings of these events. An overview of this process can be seen in Figure 17.

The start and end times for the day are determined via a probability distribution model, currently supporting normal or uniformly distributed probabilities. Lunch time is modelled by selecting a start time which is normally distributed around the time t and a duration normally distributed around d minutes where t and d are parameters unique to each employee (each of which have associated standard

[7]Note that the introduction of meetings may pull the start of the working day forward or move the end of day back.

	Administration	*Personal Administration*	*Reading*	*Writing*	*Converting*	*Surfing*
Administration	0	0.2	0.2	0.2	0.2	0.2
Personal Administration	0.3	0	0.175	0.175	0.175	0.175
Reading	0.3	0.175	0	0.175	0.175	0.175
Writing	0.3	0.175	0.175	0	0.175	0.175
Converting	0.3	0.175	0.175	0.175	0	0.175
Surfing	0.3	0.175	0.175	0.175	0.175	0

Figure 16: Markov Transition probabilities

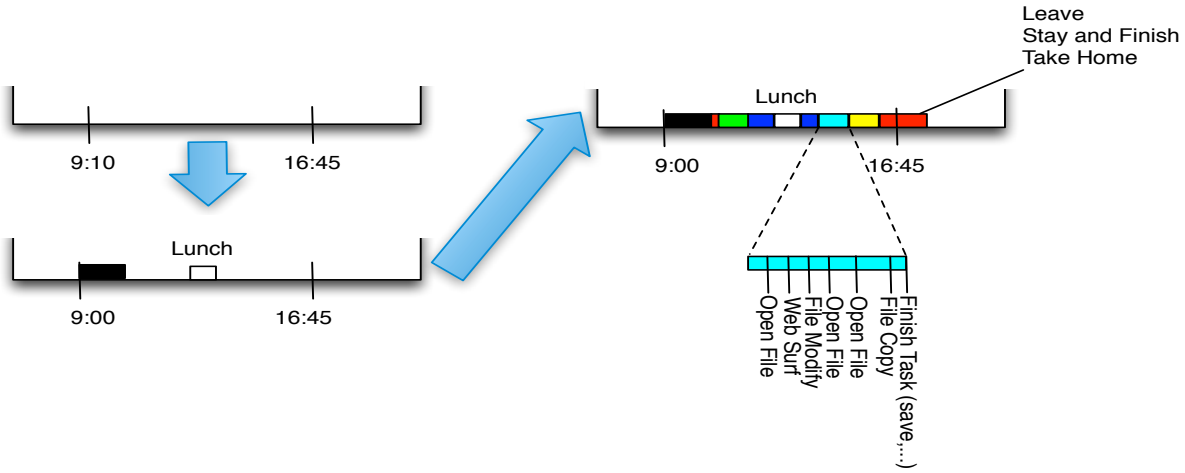


Figure 17: Individual events on a time-line

deviations). Then for each hour of the day the attendance of a meeting is determined based on a uniform probability distribution where each employee has their own probability of attending meetings. If the first meeting of the day starts before the employee's desired start time then the start time is moved forward to the start of the meeting. Likewise, the time for finishing work is moved back to the end of the last meeting if required. If meetings overlap with lunch breaks then either the portion of the lunch break which lies outside of the meeting remains or if the lunch break is completely within a meeting then the lunch break is lost (and it is assumed that lunch is provided in the meeting).

The rest of the working day is divided into activities where the selection of the next activity to perform is determined using the Markov transition matrix and the duration of activities is computed based on a poisson distribution. As it is assumed that employees will not start the day with a Surfing action a fake initial action of Surfing is assumed with zero duration. If an action over-runs into either a meting or the lunch break it is assumed that the remaining portion of that action will be completed when the member of staff returns to their desk.

In the case where the current action runs beyond the desired time for end of day then the employee will perform one of three possible actions: stay late to finish action, leave action to finish the following

Action	*Primary Event*
Administration	File open either read or write
Personal Administration	File open for write
Reading	File open for reading
Writing	File open for writing
Converting	Two files opened one read one write
Surfing	URL opened

Table 7: Automatic events at the start of an action

day or take the action home to finish. Each employee will have their own personal probabilities for each of these options. In the case of taking the work home it is assumed that all files that are required for the action are copied to a USB stick. On returning to work the employee will perform a short task of copying any files which were written to back onto the system.

At the start of a new activity the employee may change computer. This depends on the previous and new activity. Here again a Makov transition matrix (unique to each employee) is used to determine of a computer change takes place. The log out, movement between computers and log in to the new computer are assumed to be instantaneous.

During an action individual events will occur representing: file {write, read, copy, move, delete}, insert or removal of USB storage and web surfing. For each action type an initial set of events will be generated these are depicted in Table 7. For each of these actions a security level is associated with the action. A combination of action and security level will determine the the probabilities of other specific events happening during the action. The subsequent security levels of all other events will be affected by this initial security level. For example if an employee first opens an Unclassified document for writing then they should have a very low probability of opening a High-Sensitivity file for reading. This again is handled through a set of probabilities specific to the action and security level of the action. The time between each event is formed from a poisson distribution. At the end of each action a new file will be created with probability p where p depends on the action being performed and the security level of the action.

There are many reasons why an employee may be unhappy with the organisation in which they work, they may have: discovered that their contract is not going to be renewed, been rejected for promotion, been disciplined within the organisation, been bullied or humiliated by other employees, had a poor annual performance review or moral / philosophical objections to something the organisation is doing. These are a complex and difficult to pin down set of subjective criteria. At present we simply assume that a member of the Human Resources department can provide a simple metric value indicating an employee's happiness with the organisation. A value is provided on a fortnightly basis indicating one of five levels of happiness – zero indicating absolute hate for the organisation whilst a value of five indicates that they are very happy. As transitions between different levels of happiness are not independent – you are unlikely to move from hate to extremely happy in one jump – a Markov transition matrix is used to determine the next state of happiness. This matrix favours making small changes in happiness by being diagonally dominant.

We have developed a set of fictitious employees (Jo, Petra, Scott, Sally, Ben and Mel) each of whom have their own set of probabilities as defined above. These probabilities are derived from the employee profiles:

- **Jo:** ambitious person – is willing to do what it takes to get an advantage and impress those people above him, most nights he will take work home, likes to attend meetings to get noticed, has normal access rights.

- **Petra:** journey-person – is the chief executive, has not had a promotion in many years and does not see this changing in the future, works a strict 9-5 day with minimal effort, anything which isn't finished by 5pm is left for the following day, likes meetings as they fill up time without effort, has normal access rights.

- **Scott:** Frontline secretary – paid a minimal wage but experiences significant aggravation from other employees, works from 9-5 and has no need or desire to stay late or work from home, is rarely seen in meetings and has a low level of security access.

- **Sally:** all round good employee – likes her job and is keen to help out others, acts as IT support when no-one else is available, although she works 9-5 but there is significant variation in this, sees meetings as pointless so avoids when she can, does not work from home but will remain at work to finish the current task, due to her IT skills is often found on other people's computers fixing problems.

- **Ben:** tech support – works to support the computers within the organisation, is normally in work between 8am and 6pm but due to the nature of his work there is significant variance, rarely gets invited to meetings, often stays late to fix problems, does not take files with him if he works from home, and works on a large number of computers in the organisation.

- **Mel:** PA to Chief Executive – arrives at 8:30am and leaves at 5:30pm, will attend many meetings with her boss, is often found working late or working from home.

Each of our employees is simulated for an eighteen month period. For the first twelve months the exhibit only 'good' behaviour – where no employee performed data theft. However, each employee is set to go 'bad' at some point in the last six months. We define here the four different ways that an employee may go bad:

- **Bad1:** a large number of files were stolen on a single day.

- **Bad2:** from a given date the employee will start stealing a small number (normally 2) of files until the end of the simulated period.

- **Bad2_gaps:** a modification of the Bad2 case in which the employee would not steal files every day but would instead have 'gaps' in their theft pattern when they did not steal files on a given day or days. For example after five days of continuous file stealing, the employee does not steal files for two days. Note that this does not take the weekends into account where the employee would not be expected to be accessing the system.

- **Bad3:** from a given date the employee will start stealing files, however, unlike Bad2 the number of files stolen each day will increase day by day.

It should be noted here that the simulated data takes weekends into account adjusting the theft patterns accordingly. We randomly select the date on which an employee will go 'bad'. We acknowledge that there is an implicit bias in our approach by generating our own synthetic data to evaluate the AI approach we have developed. However, to minimise the effect of this bias we had two independent teams: one team working on AI development whilst the other team worked on generation of synthetic employees. Purposefully minimising the communication between these two groups.

6 Results

In this section we provide an evaluation of Ben-ware in terms of detection accuracy for insider threats but also in terms of the impact the Ben-ware system has on the legacy hardware on which it runs. In this way we can evaluate both the effectiveness of the Ben-ware approach, but also the impact that it has on the resources it is developed to protect.

6.1 Probes

Here we evaluate the impact of running probes on legacy hardware which is assumed to be significantly less powerful than current offerings. In terms of this work we have evaluated Ben-ware on laptops which are over eight years old. We evaluate the impact here on the memory footprint, disk footprint, network footprint and CPU load on such an old laptop.

6.1.1 CPU impact

To find out how the probes might affect the performance of computers within an organisation we evaluated the impact of running just the Ben-ware probes on a number of legacy laptops. During the tests there was no identifiable impact to the performance of the laptops due to running the probes. We present here the empirical evidence derived from our tests on the lowest specification (oldest) laptop that we had available – a twelve year old Sony VAIO PCG-FR which was powered by an AMD mobile Athlon XP 2000+ processor (1.67GHz) with 256GB of RAM and a IDE hard disk. All other laptops had greater specifications and thus exhibited less impact from the running probes. The laptop ran Windows XP Home patched to Service Pack 3. The operating system had been installed in 2008 and had been used on a daily to weekly basis. Numerous software packages had been installed and removed from the system.

We conducted five different sets of experiments on the laptop to determine the impact of the probes, in each case the experiment was conducted both with the probes active and with all of the probes deactivated. A small Java program was run during the test which recorded the proportion of each second when the CPU was active. For the idle case the system was left to run for a period of ten minutes, whilst in all other cases a script of actions to perform and the times those actions should be performed was followed – thus maximising the consistency between experiments between five runs to reduce experimental errors. The conducted experiments were:

- **System idle:** no activity for ten minutes
- **File copy internal:** 100 files within the laptop
- **File copy USB:** copying 100 files to a USB stick
- **USB insert and removal:** this experiment consisted of inserting a USB stick, which contained a hundred Microsoft Word documents, into the laptop and then removing the USB stick. To demonstrate the impact of different removal approaches the first two removals were proceeded with a dismount of the stick, whilst for the remaining eight the memory stick was just pulled out.
- **Document edit:** For this experiment a Microsoft Word document was opened, the contents modified, the document saved and then re-saved before the document was closed. In this case the script was followed five times.

In Figure 18 we present the average increase in CPU load on the system due to running Ben-ware probes. The average increase was calculated as:

$$\frac{\sum_{i=0}^{N} b_i - \sum_{i=0}^{N} n_i}{N}$$

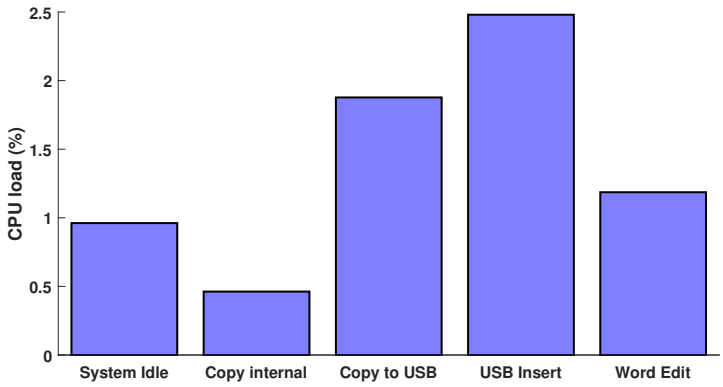


Figure 18: Average CPU increase with Ben-ware

where the number of sample points is N, b_i is the CPU load recorded on the system whilst running the probes and n_i is the CPU load when Ben-ware probes were disabled. In all cases running the Ben-ware probes added no more than an average of 2.5% extra load to the CPU. With the worst results being recorded for the experiment where the USB device was inserted – this extra load being a consequence of the probe needing to scan the whole USB file structure as the device is inserted. When copying files to the USB device this increases the CPU load by 1.9% as again the probe needs to re-scan the file structure on the USB disk. When editing a Word document this causes file open / write / close operations to occur which are detected by the probes. As document editing happened on the main hard disk of the laptop the impact was only 1.2% as file system scanning was much faster on this device. The idle stage, in which the probes sent heart-beat messages and monitored for file changes, increased the CPU load by 0.96%. It may seem surprising that the file copy experiment increased the CPU load by just 0.46% – less than the idle load. However, this can be explained by the fact that the file copy took less time than a heartbeat cycle of the probes thus the only impact in this case is the monitoring of the files. If a longer file copy had been performed the background monitoring and heart beat would have become apparent here.

Although we cannot directly compare individual CPU values between the active and inactive logs we can compare the median CPU load values – this indicates the average shift in CPU load. For the idle experiment the median CPU load was increased from 3.8% to 5%, whilst for the copy to USB experiment the median CPU load was increased from 27% to 30%. Thus there is an impact on performance caused by running the probes but in general these values are negligible.

In Figure 19 we depict the CPU trace for USB inserts and removals with both the Ben-ware probes active and inactive. The peaks present in the graph mark the point as the USB stick is being inserted – in most cases the peaks when Ben-ware is active are slightly higher. The bulk of the CPU spike can be attributed to the operating system which opened a new Explorer window each time the USB device was inserted. The variability in the load in the operating system is likely to account for why the peaks for active Ben-ware probes is sometimes lower than that for inactive probes. It should be noted that we cannot directly compare values between the two traces as they were conducted at different times – though under very similar conditions. The slight increase in peak height for active probes is likely to be a consequence of the file system traversal.

Figure 20 presents the CPU load trace for the experiment of document editing. Here the peaks correspond to the opening and saving of files. Irrespective of whether Ben-ware probes were running or not the CPU load maxes out at 100% during file operations. Though as these CPU maximisations only happen for one second a user would in general not be aware of them. In addition when the probes are active the length of these max CPU load intervals does not increase. The Ben-ware probes introduce a

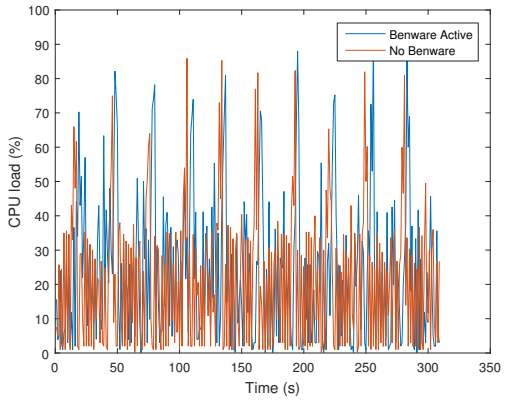


Figure 19: CPU load - 10 USB insert / remove

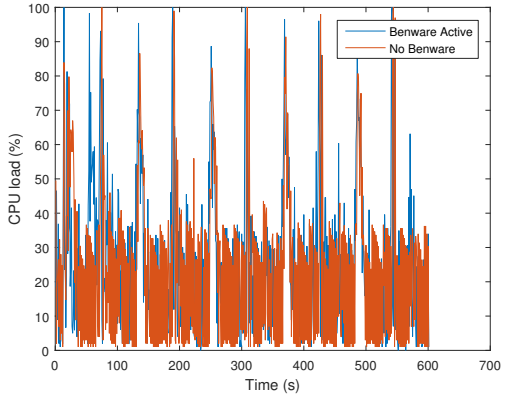


Figure 20: CPU load - Document edit

number of small peaks between the main file-action peaks, most likely a consequence of the traversal of the filing structure – whilst attempting to detect new file writes.

6.1.2 Memory Impact

Both the System and User instances of the Ben-ware probes are instantiated as separate java programs each running within its own Java Virtual Machine (VM). The memory consumed by the System instance, along with its VM was 13,268K and the memory consumed for the User instance was 10,792K – requiring less memory due to having less active probes and acting as a slave to the System instance. Thus the total memory impact from running Ben-ware probes is 24,060K (∼23.5MB). On our twelve year old legacy laptop this would take up approximately 10% of the 256MB memory. This is seen as moderately significant for such a low-memory laptop. However, it can be noted that this is just a prototype of the Ben-ware probes. If these probes were to be deployed into a real environment then they would be most likely re-coded into a much more memory efficient language without the need for a significant virtual machine.

6.1.3 Disc Space and Networking

Although the memory impact for the Ben-ware probes is moderately significant the actual byte code is much smaller – only 53KB including configuration files. However, further disc space may be required to store messages created by the probes when networking is not available. We can approximate this if we know the average message size and the expected number of messages per unit time. To compute this we ran a test session lasting 249 seconds which consisted of 24 heartbeat messages, 172 messages relating to file activity and two USB removal events. This generated a log of 19,018 bytes – thus the average data generation rate is 76 bytes per second. Thus if the laptop were disconnected from the network for the duration of a normal working day (eight hours) this would equate to 2188800 bytes ($\sim$2.1MB) of data. If we remove the user activity events from our test session this would leave just the power events, login and heartbeat events – roughly equivalent to a computer sitting idle overnight. In this case a total of 1,541 bytes would be created for our log – at a rate of 6.2 bytes per second. Thus over the 16 hours of idle time remaining in a day this would lead to 357,120 bytes ($\sim$0.35 MB). Hence each 24-hour period could be stored in $\sim$2.5MB. Given that there was 50GB free on the hard disk of the Sony laptop (total size 80GB) then this would take approximately 56 years to exhaust the free space with probe data if no network connectivity were to be available.

As the intention here is to transmit data from the probes to the MLC at the earliest opportunity we need to consider the impact that this would have on the available network link. Given that we know the data generation rate is between 6.2 and 76 bytes per second we can use this in conjunction with the network bandwidth to determine the impact on the network connectivity. Remembering here that the Ben-ware system should not override the functional requirements of the organisation. In Table 8 we present the impact on overall network capacity when using Ben-ware over different network bandwidths. Here the time is for how long it would take to transmit an eight hour active log in one go, and $\bar{u}$ indicates the proportion of the network bandwidth which would be used given that the messages are transmitted at the time of generation. We assume here that the available bandwidth is 80% of the stated maximum after we remove the capacity lost due to payload headers and gaps. It can be seen from the table that only twenty active users could saturate a 14.4Kbit modem link, whilst if the bandwidth is measured in Mbits then many users can be active simultaneously. Thus if multiple users are connected via a 14.4Kbit line then throttling would be required within Ben-ware.

Network bandwidth	**Lower generation rate** (6.2 bytes/s)		**Upper generation rate** (76 bytes/s)	
	Time	$\bar{u}$	**Time**	$\bar{u}$
14.4Kbit/s	124s	0.43%	1520s	5.3%
28.8Kbit/s	112s	0.22%	760s	2.6%
33.6Kbit/s	53s	0.18%	651s	2.3%
48 Kbit/s	37s	0.13%	456s	1.6%
1Mbit/s	1.7s	0.0059%	21s	0.074%
10Mbit/s	0.17s	0.00059%	0.21s	0.0074%

Table 8: Transmission times & network utilisations

6.2 MLC

Here we test each of the components within the MLC in terms of the impact that they have on a legacy computer. As it is assumed that the MLC will only run on a computer with enough power to meet its needs without adversely affecting the user of that computer we run our tests on a laptop with an Intel

Pentium M processor running at 1.73GHz and 512 MB of RAM. The laptop was running the Microsoft Windows XP Professional operating system which had been installed some years previously.

Performance results are presented for each of the different services which comprise the MLC. Each is tested with the system under load and in an idle state (i.e. no requests being placed on the MLC) over a test period of one hour. We used this to generate statistics on CPU load and memory use. In all cases the CPU load in idle state was too small to observe – and was effectively considered to be zero – however, when the system was under load this lead to a 1-2% extra load on the CPU – which is considered negligible. The only exception to this was the aggregator which peaked at 15% CPU load, however, as the aggregator only runs for a few seconds each hour for each user this was not considered an excessive impact on the user of the system. If this were considered to be too excessive for the user then the base level specification for a computer which runs the MLC could be increased until the impact were considered tolerable.

In the rest of this sub-section we will discuss other metrics relevant to the MLC:

	Idle	**Load**
CPU Load (%)	0%	2%
Memory Used (kb)	16944	4300
Message Throughput (msg/sec)	4054	
Maximum Concurrent Connections	>320	

Table 9: Server's message throughput

	Idle	**Load**
CPU Load (%)	0%	15%
Memory Used (kb)	10224	25000
Messages Generated	2544	
Average Message Size (bytes)	63	

Table 10: Aggregator's messages generated

	Idle	**Load**
CPU Load (%)	0%	1%
Memory Used (kb)	108	3192
Message Write Rate (msg/sec)	36	

Table 11: Database's message write rate

Message throughput (c.f. Table 9): – the rate at which the server can process messages. In this experiment a connection was made with the MLC from a piece of code acting as a Ben-ware probe. Through this connection 9,244 messages were streamed without any time gaps between them. This data was the synthetic data generated for testing Ben-ware. The time taken for the MLC to consume all messages was recorded. Testing the maximum number of concurrent connections to the MLC was conducted using a simple telnet client connecting to the MLC. In this case the resources on the computer were exhausted before the MLC was itself unable to accept further connections – therefore this should be seen as a lower bound on the maximum number of connections.

Messages generated (c.f. Table 10): – the number of aggregate messages that would be produced in a one year period – for egress to the AI. In this case we tested this using the synthetic data discussed

previously. It took the laptop just 124 seconds to generate all aggregate messages. It is worth noting here that the processing of aggregates would normally happen once per hour thus each aggregate processing would take only 0.067 seconds to perform – this is assuming that the user worked 46 weeks per year with each week being five days long and each day being eight hours of computer activity. The average message size observed was 63 bytes long.

Message write rate (c.f. Table 11): – the rate at which messages could be written to the MySQL database once the server has accepted the data and performed an SQL `INSERT` command. In this case the write rate to the MySQL database is much slower than the MLC message throughput rate. However, this is not considered a problem as an MLC receiving more than 36 messages per second could delegate some of its users to MLCs running on different computers.

6.3 Intelligent Agent – AI

The Intelligent agent has been developed as a Java application which have been evaluated on both Linux and Windows (versions 7 and 8). In order to use the Intelligent agent on an employee or if deploying within a new organisation the AI components and statistical analysis components need to be trained for the employee(s). This allows the system to build a model for the normal behaviour of an employee based on a historical trace of their activity. The time required to train the system for an employee depends on the diversity exhibited by the employees behaviour – with a highly consistent employee requiring less training data than an employee who has significant variation in their activity. We estimate that an employee who is fairly consistent in their behaviour would require approximately three months of training data ($\sim$ 60 data points). However, if an employee exhibits a high degree of variability in their (computer use) behaviour, then it is advisable to train the AI with more data points – say 200-250 points.

If historical data is not available for an employee it is possible to train the AI based on data obtained from other employees performing the same or similar job roles. Although this will not work as well as personal training it will allow shorter term analysis to be performed. In either case the AI should be re-trained on a regular basis – at least once every six months. This will help the system cope with the changing work-patterns of employees as they become more experienced in their role. This can be performed automatically without the need for an administrator to provide parameters for the SVDD or the number of clusters to use. This is possible due to the fact that the Intelligent Agent contains built-in cross-validation functions that can divide the training data into different ratios of training data to test data – allowing auto-tuning of the parameters.

Once trained, to the normal behaviour of an employee, the Intelligent Agent can be used to identify anomalous behaviour exhibited by the employee by comparing their current activity against the model held in memory. Our experiments also show that it takes around 8 milliseconds to make a decision against a data vector (aggregated daily activities of a user). Training takes substantially longer – around two minutes per user when processing eighteen months of data on a 2.4GHz laptop with 8GB of RAM, however, this could be done offline – for example at night when computers are not in use. Each user will normally be evaluated once per day – unless a trigger event has occurred, in which case an evaluation will be made immediately. As the Intelligent Agent for each employee will be located on an MLC close to the normal computer used by the employee this allows for the workload to be distributed around the organisation. Once the software is activated on a computer, it takes around 100MB of memory and disk space requirement for the model file that is around 4KB per user. However, this can normally be run when the user is not active on the computer.

In Tables 12–14 we present the analysis of our synthetic employee data when processed through the Intelligent Agent under different risk thresholds. In these tables α indicates the number of true positives – the number of days when Jo acted in a bad manner, β is the number of true positives identified by the Intelligent Agent – the number of days when the AI correctly identified that Jo behaved badly, and Ω

Test Dataset	α	β	Ω
Jo_Bad1	1	1	0
Jo_Bad2	85	82	1
Jo_Bad2_gaps	76	72	0
Jo_Bad3	69	66	1

Table 12: User: Jo – under a VERY HIGH RISK threshold

Test Dataset	α	β	Ω
Jo_Bad1	1	1	7
Jo_Bad2	85	82	2
Jo_Bad2_gaps	76	72	2
Jo_Bad3	69	66	2

Table 13: User: Jo –under a HIGH RISK threshold

is the number of false positives – the number of days when the AI incorrectly identified Jo acting in a bad manner. The risk threshold (NO RISK, MODERATE RISK, HIGH RISK or VERY HIGH RISK) is used as a modifier to the AI indicating how much of a risk the organisation is willing to take with that employee. Thus a MODERATE RISK threshold will be more likely to throw an anomaly detection than a VERY HIGH RISK threshold. As there is only one start date for a bad action in each data set the difference between α and β can be seen as the number of days between the bad action commencing and the Intelligent Agent identifying it.

Table 12 depicts the results where the organisation is willing to take a VERY HIGH RISK with Jo. Each row of the table represents a six month period in which the employee could have gone bad and the entire data set represents eighteen months – 391 days as it is assumed that employees don't work at the weekend. For both the cases of Jo_Bad2 and Jo_Bad3 it can be seen that the Intelligent Agent took three days to identify that Jo had started stealing files. However, for these two cases the number of files stolen per day is just two – thus a total theft of just six files. Hence we claim our system is sensitive to small variations in use patterns. For both bad cases the number of false positives was just one. Although both these and the true positive cases would require analysis by an operator these values are relatively small thus minimising the amount of wasted work by the operator. By introducing gaps into the constant theft model the Intelligent Agent requires an extra day to identify the theft – though there are now no false positives.

Synthetic logs were produced for each of the six employees identified in Section 5.6 with each employee being allowed to go bad in each of the four different ways. Thus generating 24 different synthetic logs, each of which was processed by the Intelligent Agent. Figure 21 summarises the effectiveness of the Intelligent Agent at identifying anomalous behaviour – indicating the average number of days taken to identify that an employee had gone bad along with the standard deviation (marked as whiskers on the bars). For all cases Bad1 was identified on the day that it happened. This is irrespective of whether the employee would normally take work home with them or not. Similarly for Bad2 – stealing a constant number of files – this always took exactly three days to identify. If we allow gaps (days when files aren't stolen) then the Intelligent Agent averages just less than three days to identify the theft. This is brought down by Sally for whom the Intelligent Agent only takes two days to identify. Though alongside Jo this brings in some variation. When the number of files stolen per day increases (Bad3) the Intelligent Agent takes the longest time to identify theft ($4\frac{1}{3}$ days). The variation here is a consequence of Sally and Bob who's normal working pattern does not include taking work home.

We present the results for the average number (and standard deviations as whiskers) of false positives

Test Dataset	α	β	Ω
Jo_Bad1	1	1	24
Jo_Bad2	85	82	10
Jo_Bad2_gaps	76	72	10
Jo_Bad3	69	66	14

Table 14: User: Jo – under a MODERATE RISK threshold

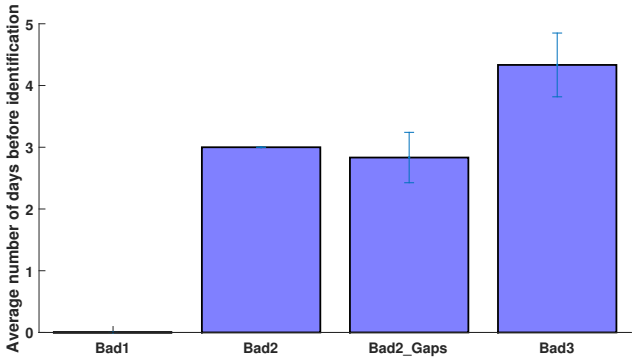


Figure 21: Average time to identify bad action

in Figure 22 – those cases where the Intelligent Agent incorrectly identifies an anomalous act. In the cases of Bad1 and Bad3 no false positives were observed. For the cases of Bad2 and Bad2_gaps false positives were observed – due to Jo and Ben. Jo is ambitions and often takes work home while Ben is a technician who has significant variation in his usage pattern. Both of these scenarios could lead to the Intelligent Agent miss-identifying their activity as bad. Again, the number of these false positives is small which helps operators as they don't waste too much time following up on incorrect leads.

We investigate the effect of reducing the level of risk that the organisation is willing to take with members of staff in Tables 13 and 14 – reducing the risk from VERY-HIGH RISK (Table 12) down to HIGH RISK and MODERATE RISK. Although we may expect reducing the tolerable risk would reduce the time between bad activity starting and the Intelligent Agent identifying the act this is not observed as the values of α and β remain the same. Though it does increase the number of false positives. Slightly for the case of HIGH RISK and significantly for the case of MODERATE RISK. Although the results here are not conclusive this may be a situation in which real employee data, where the employee does not instantaneously change overnight, might be better analysed by varying the risk threshold.

7 Discussion

The Ben-ware system combines artificial intelligence with machine learning and real 'human factors' information – the types of information and risk data that the organisation should already know about the individual it works with or employs – in order to identify anomalous behaviours that could threaten the work of closed organisations, or the parts of organisational systems that are particularly sensitive to the organisation's core mission.

While this paper has focused upon the technological 'proof of concept' issue, it will be necessary to address more fully a range of ethical conflicts in order to take the idea of Ben-ware forward, especially if the Ben-ware is to be used outside a closed organisation. In this pilot project, which does not use live data, the practical question of ethics did not arise and if live data had been used, then the issue of

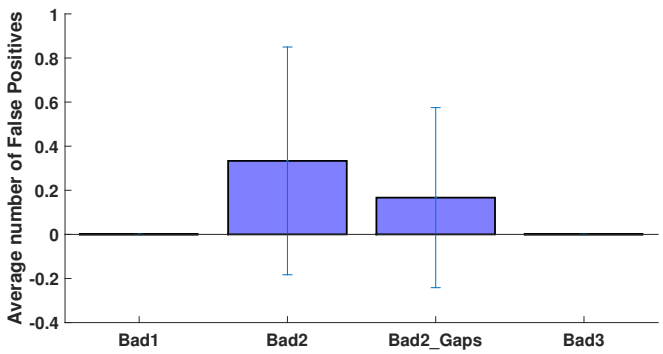


Figure 22: Average occurrence of false positives

ethics would have largely been contained as the closed nature of the organisations we seek to protect and their specific information security policies linked to employment contracts would have allowed for such data gathering. Linkage of the organisational goals, information policies and employment contracts is important so that we can align the expectations and understanding of organisations and their employees. In such linkage, however, it is also very important to maintain a balance between the need for security and providing employees with a large degree of occupational freedom and flexibility to enable them to effectively achieve the organisation's goals without stifling the creativity often required to best meet those goals; and also develop personally within its framework. We believe that Ben-ware will be most effective when such a balance exists.

It is our belief that as we develop Ben-ware we can apply it to the closed (from public view) parts of information systems which contain confidential or restricted information: information and data which have to be protected from being leaked. Information, which if leaked, could not only compromise the running of the organisation, but is entrusted to it in good faith by providers and for which it has a moral, legal and commercial responsibility to protect the information and its owners from abuse and misuse. Moral, in terms of being entrusted by the subjects or owners of the particular sets of information held; legal in terms of complying with the Data Protection Laws that most jurisdictions possess, and commercial in terms of organisational reputation within the market place which is essential to maintain both consumer and shareholder confidence. When taking Ben-ware forward it will also be important to address the concerns raised in the wider public debates over mass human surveillance. Not least, because one of the principal functions of Ben-ware is to reduce false-positives so that security resources can be focused upon resolving the main threats to the information systems in which it is used.

8 Conclusion and Future Work

The identification of insider threats within an organisation is a complex process due to the fact that system administrators are dealing with people who have legitimate access rights to that system. However, by capturing user activity events and processing these through an AI engine whose machine learning informed by human factors information we are able to identify abnormalities in individual employees profiles. Although abnormal behaviour need not imply malicious activity, the Ben-ware system does help reduce the search space significantly, thus allowing human operators to focus their security efforts where most needed.

As part of this work, we have developed a successful prototype of our Ben-ware vision for evaluating insider attack threats against a closed organisation where the attacker intends to 'steal' files and data.

The Ben-ware system is capable of identifying through an AI engine when the activity of a particular user becomes anomalous with their normal behaviour patterns. This can be achieved immediately when the user steals a significant number of files on the same day or within three or four days in the case where exfiltration happens at a much reduced rate, say two or more files per day, thus raising suspicions that the employee is an inside threat to the organisation. Although three to four days may seem a long time, it is anticipated that future versions of Ben-ware will reduce the time frame, especially as more bespoke information is inserted into the mid-level controller to inform the machine learning aspect of the AI component.

The detection rates achieved through Ben-ware were found to be very good and the system was able to identify all occurrences of bad activity, the true positives, whilst at the same time exhibiting very low rates of false-positives (an average of less than 0.4 out of 182). This level was achieved whilst keeping the CPU overheads of Ben-ware to an acceptably low level which is distributed over the computers within the organisation. The CPU load exerted by the Ben-ware probes was less than 2.5% whilst the CPU load from the MLC was between 1% and 2%. While the aggregator can exert a 15% CPU load, this only runs once per hour and only for less than a second each time, so this temporary load is not regarded as significant. Likewise, the memory footprint of Ben-ware is small at less than 25MB and can easily be catered for on a legacy computer with only 256MB of RAM – as found in many older systems of eight or more years. This load is even more impressive when the fact that the code has not been optimised in any way for memory or CPU impact. Therefore we claim that Ben-ware is scalable solution to the detection of insider threats and holds great promise for future applicability.

At present we are only assessing employees based on anomaly detection provided by an SVDD approach, however, the information collected could be processed in many different ways. For example we could give each user-event a risk value and each user a risk budget for a day, week or month which would normally satisfy their work needs. Exhaustion of this budget would indicate potential misbehaviour. This could be further extended by combinations of actions having a greater cost than the individual actions, so, inserting a USB device and reading a sensitive file could incur greater cost than performing each action alone. Alternatively, the event logs could be analysed to identify normal usage patterns which could, in turn, be used to identify if another person is mis-using the legitimate user's account, or that an employee is performing an unusual sequence of actions. By exploiting different approaches for analysing the wealth of data we are collecting about users a more robust and reliable means can be established to detect insider threats.

Thus far we have only demonstrated Ben-ware with small numbers of synthetic employees in order to demonstrate the feasibility of the approach. However, as our approach becomes more distributed then the impact on the computers in the system should remain constant as the size of the organisation grows. We acknowledge that there is a potential bias in the synthetic data used, however we have attempted to minimise this effect by developing Ben-ware and the data independently of each other. Though we now seek to perform a real-world deployment of the Ben-ware system which could also include regular information collected about employees, such as occupational personality test profiles as well as human resources information, but also advanced risk based information and even disciplinary records. As stated in the previous section, this latter development will increase the predictability of Ben-ware, but it will also introduce new ethical and interdisciplinary considerations into the equation that will have to be balanced by clear information security policies and employment contracts, but also instructions as to what employees can and cannot do on particular systems within the course of their duties.

Acknowledgments

We would like to thank UK Defence Science and Technology Laboratory (DSTLX1000088892) for funding this work.

References

[1] B. Aleman-Meza, P. Burns, M. Eavenson, D. Palaniswami, and A. Sheth. An ontological approach to the document access problem of insider threat. In P. Kantor, G. Muresan, F. Roberts, D. Zeng, F.-Y. Wang, H. Chen, and R. Merkle, editors, *Intelligence and Security Informatics*, volume 3495 of *LNCS*, pages 486–491. Springer-Verlag, 2005.

[2] M. Bishop and C. Gates. Defining the insider threat. In *Proceedings of the 4th Annual Workshop on Cyber Security and Information Intelligence Research: Developing Strategies to Meet the Cyber Security and Information Intelligence Challenges Ahead*, CSIIRW '08, pages 15:1–15:3, New York, NY, USA, 2008.

[3] B. Bowen, M. Ben Salem, S. Hershkop, A. Keromytis, and S. Stolfo. Designing host and network sensors to mitigate the insider threat. *Security & Privacy, IEEE*, 7(6):22–29, Nov 2009.

[4] O. Brdiczka, J. Liu, B. Price, J. Shen, A. Patil, R. Chow, E. Bart, and N. Ducheneaut. Proactive insider threat detection through graph learning and psychological context. In *IEEE Symposium on Security and Privacy Workshops'12*, pages 142–149, 2012.

[5] D. Caputo, M. Maloof, and G. Stephens. Detecting insider theft of trade secrets. *Security & Privacy, IEEE*, 7(6):14–21, Nov 2009.

[6] E. Cole and S. Ring. *Insider Threat: Protecting the Enterprise from Sabotage, Spying, and Theft: Protecting the Enterprise from Sabotage, Spying, and Theft*. Syngress, 2005.

[7] C. Colwill. Human factors in information security: The insider threat–who can you trust these days? *Information security technical report*, 14(4):186–196, 2009.

[8] B. D. Davison and H. Hirsh. Predicting sequences of user actions. In *Notes of the AAAI/ICML 1998 Workshop on Predicting the Future: AI Approaches to Time-Series Analysis*, pages 5–12, 1998.

[9] W. DuMouchel. Computer intrusion detection based on bayes factors for comparing command transition probabilities. *National Institute of Statistical Sciences Tech. Report*, 91, 1999.

[10] W. Eberle and L. Holder. Insider threat detection using graph-based approaches. In *Conference For Homeland Security, 2009. CATCH '09. Cybersecurity Applications Technology*, pages 237–241, March 2009.

[11] G. Greenwald, E. MacAskill, and L. Poitras. Edward snowden: the whistleblower behind the nsa surveillance revelations. *The Guardian*, 9:2013, 2013.

[12] F. L. Greitzer and R. E. Hohimer. Modeling Human Behavior to Anticipate Insider Attacks. *Journal of Strategic Security*, 4(2):25–48, June 2011.

[13] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Applied statistics*, pages 100–108, 1979.

[14] S. Jha, L. Kruger, T. Kurtx, Y. Lee, and A. Smith. A filtering approach to anomaly and masquerade detection. Technical report, Citeseer, 2004.

[15] M. Kandias, A. Mylonas, N. Virvilis, M. Theoharidou, and D. Gritzalis. An insider threat prediction model. In S. Katsikas, J. Lopez, and M. Soriano, editors, *Trust, Privacy and Security in Digital Business*, volume 6264 of *Lecture Notes in Computer Science*, pages 26–37. Springer Berlin Heidelberg, 2010.

[16] K. D. Loch, H. H. Carr, and M. E. Warkentin. Threats to information systems: Today's reality, yesterday's understanding. *MIS Quarterly*, 16(2):173–186, 1992.

[17] G. Magklaras and S. Furnell. A preliminary model of end user sophistication for insider threat prediction in IT systems. *Computers & Security*, 24(5):371–380, 2005.

[18] M. A. Maloof and G. D. Stephens. Elicit: A system for detecting insiders who violate need-to-know. In *Proceedings of the 10th International Conference on Recent Advances in Intrusion Detection*, RAID'07, pages 146–166, Berlin, Heidelberg, 2007. Springer-Verlag.

[19] A. S. McGough, D. Wall, J. Brennan, G. Theodoropoulos, E. Ruck-Keene, B. Arief, C. Gamble, J. Fitzgerald, A. van Moorsel, and S. Alwis. Insider Threats: Identifying Anomalous Human Behaviour in Heterogeneous Systems Using Beneficial Intelligent Software (Ben-ware). In *Proceedings of the 7th ACM CCS International Workshop on Managing Insider Security Threats*, MIST '15, pages 1–12, New York, NY, USA, 2015. ACM.

[20] J. Murphy, V. Berk, and I. Gregorio-de Souza. Decision support procedure in the insider threat domain. In *Security and Privacy Workshops (SPW), 2012 IEEE Symposium on*, pages 159–163, May 2012.

[21] N. Nguyen, P. Reiher, and G. Kuenning. Detecting insider threats by monitoring system call activity. In *Information Assurance Workshop, 2003. IEEE Systems, Man and Cybernetics Society*, pages 45–52, June 2003.

[22] J. Nurse, P. Legg, O. Buckley, I. Agrafiotis, G. Wright, M. Whitty, D. Upton, M. Goldsmith, and S. Creese. A critical reflection on the threat from human insiders – its nature, industry perceptions, and detection approaches. In T. Tryfonas and I. Askoxylakis, editors, *Human Aspects of Information Security, Privacy, and Trust*, volume 8533 of *LNCS*, pages 270–281. Springer-Verlag, 2014.

[23] D. B. Parker. *Fighting computer crime: A new framework for protecting information*. John Wiley & Sons, Inc., 1998.

[24] E. Pauwels and O. Ambekar. One class classification for anomaly detection: Support vector data description revisited. In P. Perner, editor, *Advances in Data Mining. Applications and Theoretical Aspects*, volume 6870 of *Lecture Notes in Computer Science*, pages 25–39. Springer Berlin Heidelberg, 2011.

[25] S. Pramanik, V. Sankaranarayanan, and S. Upadhyaya. Security policies to mitigate insider threat in the document control domain. In *Computer Security Applications Conference, 2004. 20th Annual*, pages 304–313, Dec 2004.

[26] M. Schonlau, W. DuMouchel, W.-H. Ju, A. F. Karr, M. Theus, and Y. Vardi. Computer intrusion: Detecting masquerades. *Statistical science*, pages 58–74, 2001.

[27] E. Schultz. A framework for understanding and predicting insider attacks. *Comput. Secur.*, 21(6):526–531, Oct. 2002.

[28] E. Shaw, K. Ruby, and J. Post. The insider threat to information systems: The psychology of the dangerous insider. *Security Awareness Bulletin*, 2(98):1–10, 1998.

[29] L. Spitzner. Honeypots: catching the insider threat. In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pages 170–179, Dec 2003.

[30] J. Suler and W. Phillips. The bad boys of cyberspace: Deviant behavior in online multimedia communities and strategies for managing it. *J. Suler, The psychology of cyberspace. Retrieved January*, 21:1999, 1997.

[31] D. Tax and R. Duin. Support vector data description. *Machine Learning*, 54(1):45–66, 2004.

[32] P. Thompson. Weak models for insider threat detection. In *Proceedings of Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense III*, volume 5403, pages 40–48, 2004.

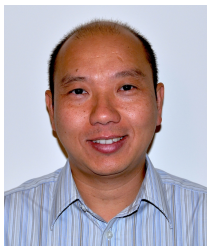
[33] T. Tuglular and E. Spafford. A framework for characterization of insider computer misuse. *Unpublished paper, Purdue University*, 1997.

[34] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, NY, USA, 1995.

[35] D. Wall. Enemies within: redefining the insider threat in organizational security policy. *Security journal.*, 26(2):107–124, April 2013.

Author Biography

Andrew Stephen McGough received a B.S. in Mathematics from the University of Durham in 1993 followed by an MSc and PhD in Computing from Newcastle University in 1996 and 2000. Currently he is a Lecturer in Computing at Durham University. His research interests include Modelling and Simulation of Systems, Machine Learning for analysis of systems, Big Data Analytics and the analysis of rare events within complex computing systems.



Budi Arief is currently a Senior Research Associate at the School of Computing Science at Newcastle University, UK. He is about to take up a Lectureship in Computing post at the University of Kent, UK, starting from February 2016. His main research interests are human aspects of computer security, the dependability of computer-based systems, and mobile/ubiquitous computing. He obtained his B.Sc. in Computing Science (First Class) and Ph.D. in Computing Science in 1997 and 2001 respectively, both from Newcastle University.

Carl Gamble is a Research Associate in Computing at Newcastle University, UK. He worked as a manufacturing engineer before completing a PhD in service-oriented architecture. He has worked on metadata-driven reconfiguration, representation and abstraction of provenance data and is now working on the INTO-CPS project developing methods for design space exploration, and provenance and traceability in CPS engineering.

David S. Wall is Professor of Criminology in the Centre for Criminal Justice studies, School of Law, University of Leeds, UK where he researches and teaches cybercrime, identity crime, organised crime, policing and intellectual property crime. He has published a wide range of books and articles on these subjects and he also has a sustained track record of interdisciplinary funded research in these areas from a variety of Research Councils and Government Agencies.

John Brennan is currently a PhD candidate at the School of Engineering and Computing Sciences at Durham University. His main research interests are Big Data analytics, complex network analytics, energy efficiency in large systems and computational resource scheduling. He received a BEng in Electronic Engineering and Computer Systems in 2012, followed by an MSc in Computer Science in 2014, from the University of Huddersfield.

John Fitzgerald is a full Professor in Computing Science at Newcastle University, UK. A researcher and practitioner in formal verification (PhD, 1991), avionics and embedded systems, he directed the EC COMPASS project on model-based engineering for systems of systems and now leads methods work in INTO-CPS. He is academic lead in the University's $90m centre for digitally enabled urban sustainability. He is a member of INCOSE and a Fellow of the BCS.

Aad van Moorsel is a full Professor in Distributed Systems and Head of School at the School of Computing Science in Newcastle University. His group researches security, privacy and trust containing elements of quantification through system measurement and predictive modelling. Aad worked in industry 1996 to 2003 at Bell Labs/Lucent Technologies and at Hewlett-Packard Labs, both in the United States. He has a PhD in computer science from Universiteit Twente, a Masters in mathematics from Universiteit Leiden and was a postdoc at the University of Illinois at Urbana-Champaign.

Sujeewa Alwis is CEO of Live Foresight and before this was Technical Manager at Cybula Ltd, Senior Research Scientist at Pattern Computing Plc and a Research Associate at the University of York. He specialises in machine learning with a track record in solving hard practical problems. He has developed solutions for global organisations including Government Defence & Security Organisations, BAE Systems, Rolls Royce, Sun Microsystems, Vodafone, Reckitt Benckiser and Shimadzu Corporation.

Georgios Theodoropoulos is the Executive Director of the Institute of Advanced Research Computing at Durham University, UK where he holds a Chair in the School of Engineering and Computing Sciences. Prior to his current position, he was a Senior Research Scientist with IBM Research. His current research focuses on prescriptive analytics for complex socio-technical systems and co-design for workload optimised systems.

Ed Ruck-Keene is Impact and Business Development Manager at the Institute of Avanced Research Computing, Durham University, UK. Prior to this he was a Director at Tactic Solutions Ltd.