Edith Cowan University
Research Online

Australian Information Security Management Conference

Security Research Institute Conferences

2010

Anomaly Detection over User Profiles for Intrusion Detection

Grant Pannell
University of South Australia

Helen Ashman
University of South Australia

Originally published in the Proceedings of the 8th Australian Information Security Mangement Conference, Edith Cowan University, Perth Western Australia, 30th November 2010
This Conference Proceeding is posted at Research Online.
http://ro.ecu.edu.au/ism/94

Anomaly Detection over User Profiles for Intrusion Detection

Grant Pannell and Helen Ashman
University of South Australia
Adelaide, Australia
grant.pannell@unisa.edu.au
helen.ashman@unisa.edu.au

Abstract

Intrusion detection systems (IDS) have often been used to analyse network traffic to help network administrators quickly identify and respond to intrusions. These detection systems generally operate over the entire network, identifying "anomalies" atypical of the network's normal collective user activities. We show that anomaly detection could also be host-based so that the normal usage patterns of an individual user could be profiled. This enables the detection of masquerading intruders by comparing a learned user profile against the current session's profile. A prototype behavioural IDS applies the concept of anomaly detection to user behaviour and compares the effects of using multiple characteristics to profile users. Behaviour captured within the system consists of application usage, application performance (CPU and memory), the websites a user visits, the number of windows a user has open, and their typing habits. The results show that such a system is entirely feasible, that characteristics physically related to the user are more relevant to profiling behaviour and that the combination of characteristics can significantly decrease the time taken to detect an intruder.

Keywords

Anomaly detection, behavioural intrusion detection, behavioural biometrics, user model

INTRODUCTION

Intrusion detection systems are often employed in networks, to watch the flow of traffic for malicious behaviour, and for some time have been found reliable in production environments. A concept that is still in early development is the idea of a behavioural intrusion detection system that is able to profile a user's behaviour so that it can be used to determine unauthorised users or malicious use of a machine. Instead of being a network-based system, the IDS becomes host-based, running on a machine watching various types of behaviour. Behaviour can either be focused on the system, where the IDS inspects an application's behaviour for malicious and exploitive behaviour, the network, where the IDS analyses the outgoing network traffic for patterns and malicious signatures, or the user, where the IDS looks at user behaviour to determine whether the user is who they say they are or if they're performing malicious tasks. This paper focuses on the profiling users for behavioural intrusion detection with a range of different characteristics.

Like a typical intrusion detection system, a behavioural IDS can also be classified in the way it can detect an anomaly. In terms of rule-based systems, the behavioural IDS would follow a set of rules, either disallowed or allowed, in relation to a user's behaviour. For example, a user may specify that their default text editor is "vi", and when "emacs" is used an alert is issued. However, a behavioural IDS is most beneficial from a statistical anomaly-based system that learns behaviour over time and determines activity that is significantly outside the norm. So the system would learn that the user predominantly uses "vi" and since "emacs" is not frequently used it is a statistical anomaly and therefore publishes an alert. Thus the behavioural IDS determines user habits implicitly, instead of having them explicitly supplied by the user.

Many characteristics can be stored in a user model to capture a user's behaviour, e.g. keystrokes typed, the websites viewed or favourite applications. Capturing data from a range of sources allows the behavioural IDS to accurately profile individual users by modelling a range of behaviour. The user model can also be applied in different ways: A *role-based model* allows the IDS to map the user's activity against a typical behaviour model. For example, using a role-based model would allow the system to determine the difference between a user using their machine for recreational or office use; A *personal model* allows the system to characterise individual users, looking for specific behaviour traits such as common typographical errors or patterns of application use. We propose a host-based behavioural IDS that uses anomaly detection on user models to characterise individual behaviour by combining the results of multiple characteristics to improve detection performance.

LITERATURE REVIEW

Behaviour, when related to IDSs, can be modelled in terms of the system, user or the network. Even with this wide reach it is found that using behaviour to identify intruders of a computer system is unusual.

For profiling system behaviour, system logs are often analysed to determine execution paths of running applications (Forrest et al., 1996, Hofmeyr et al., 1998, Cabrera et al., 2001). This allows the detection of exploitation attacks, in particular privilege escalation. The Unix application 'sendmail' is often used as a benchmark for testing different analysis algorithms and grouping of execution paths (Forrest at al. 1996, Cabrera et al., 2001). A significant project related to system profiling is NIDES (later named EMERALD and eXpert-BSM). NIDES aimed to capture application-related characteristics, such as CPU usage and memory usage, to individually profile processes running on the machine (Anderson et al., 1995) but has not been reported in literature since 2002 (Lindqvist; Porras, 2001).
Network behaviour often involves sniffing network traffic in order to identify the normal behaviour of a host which can then be used to identify anomalies such as spam (Stolfo et al., 2006) or worm propagation (Ellis et al., 2004). This is slightly different to the workings of most network-based IDS as it aims to identify behaviour from multiple packets rather than finding malicious anomalies or signatures in individual packets.

User profiling systems are the main focus of this paper. User behaviour tracking has dated back to the 1970s (Boies, 1974) with a user study of 375 mainframe users over 4 months, although the data collected is not used to identify individuals but instead to determine how the mainframe is used. Anderson (Anderson, 1980) however, collects data from users in order to identify individual behaviour. It is often said that Anderson is the first to introduce the idea of an automated behavioural IDS where audit logs, containing file, resource and command usage, are automatically collected to determine if a user is not performing malicious tasks (Eugene, 2008). Denning (1987) is also a pioneer of the idea of behavioural IDSs as she formalises a model that is used in several prototype systems.

Behavioural biometrics is a new area that captures characteristics needed to profile a user. As opposed to biometrics, where a static unique object (i.e. a fingerprint) is used for authentication, behavioural biometrics intends to use the behaviour of a user for both authentication and re-authentication, so that a system is continuously checking if the user is who they say they are. Since behaviour becomes even harder replicate, it can reduce the possibilities of masquerading users and hijacked sessions (Yampolskiy; Govindaraju, 2008). Yampolskiy and Govindaraju (2008) have defined behavioural biometrics as covering all characteristics of a user for identification, involving their skills, style, preference, knowledge, motor-skills or strategy, in any area. It becomes important to focus on characteristics that do not require the user to perform specific tasks to train the system so that a behavioural IDS is able to seamlessly learn behaviour without any user interruption. Such characteristics range from keystroke (Umphress; Williams, 1985, Leggett et al., 1991, Monrose; Rubin, 1997, Bergadano et al., 2003, Revett, 2009) and mouse usage (Pusara; Brodley, 2004, Ahmed; Traore, 2005, Revett et al., 2008) to application-specific characteristics such as the websites viewed in a browser (Tauscher; Greenberg, 1997, Cheung et al., 1998, Cockburn; McKenzie, 2001).

Many characteristics in behavioural IDSs focus on Unix command line usage (Tan, 1995, Lane; Brodley, 1997, Balajinath; Raghavan, 2001), such as combinations and frequency of commands, and audit data (Shavlik; Shavlik, 2004, Li et al., 2006, McKinney; Reeves, 2009), such as CPU usage, memory usage, session times and resources access. One of the first implementations was by Smaha (1998), a DARPA funded project named Haystack that has not been reported in literature since 1988. Often, different algorithms are used to analyse the incoming data, however, the characteristics are generally the same.

Physical behavioural attributes of the user have also been shown to be beneficial. Keystroke dynamics looks at keystroke patterns, delays and frequency and have been shown to have excellent detection rates (Revett, 2009), and have even been adapted to identify whether a user is cognitively or physically stressed (Vizer et al., 2009). Mouse dynamics is a less perfected characteristic being a recent introduction (Pusara; Brodley, 2004). It analyses mouse accelerations, curves and time between clicks to identify unique behaviour. However, one drawback of physical characteristics is that they are less reliable with remote users.

Characteristics on the Windows operating system have been rare. Shavlik (2004) uses resource usage data to profile user and achieves a 95% detection rate and false-positive rate of "less than one alert per day", however, fails to characterise GUI behaviour. Imsand and Hamilton (Imsand; Hamilton, 2007) conducted a user study and achieved a detection rate of up to 95% by analysing the manipulation of windows, icons, menus and pointers, however, their results vary significantly depending on the attack simulation. Using data from the browser to characterise behaviour is also rare. While browser data is often used to identify user interface flaws or to group users by similar interests, it is rarely used to identify individuals (Pannell; Ashman, 2010).

The combination of characteristics within a behavioural IDS also seems to be an underexplored area. Very few systems focus on using several characteristic engines to improve performance. Behavioural IDS systems that do implement multiple characteristics often implement several metrics from a single characteristic, for example, typing delays and typographical errors of keystroke dynamics, rather than taking metrics from multiple characteristics, for example using both keystroke dynamics and mouse dynamics. In contrast, the network IDS field employs warehousing, a concept that uses several detection engines to analyse incoming data, and has been proved to decrease false-positives (Julisch; Dacier, 2002, Gu et al., 2008). Applying this concept to behavioural IDS to profile multiple characteristics is an obvious direction to increase performance.

IMPLEMENTED CHARACTERISTICS

The behavioural IDS captures data from multiple sources in order to accurately determine the behaviour of a user and therefore intruders of the system. By introducing multiple characteristics, it is possible to decrease detection times, reduce false-positive rates and capture behaviour of all different user roles. The implemented system captures data using 5 different characteristic engines in an attempt to create a comprehensive user model:

1. Applications running
2. Number of windows
3. Application performance
4. Websites viewed
5. Keystroke analysis

The running applications characteristic is used to determine the number of applications that are concurrently running and the number of new applications that are opened by the user. These two metrics determine the typical workload of the user as well as a usual set of applications they interact with. The number of windows characteristic aims to capture data from the graphical user interface layer to determine the user's style of use. The application performance characteristic captures CPU and memory usage data for each application running on the machine to determine if applications are both being used and acting in a similar manner. The websites viewed characteristic captures web history data from the browsers installed on the user's machine, specifically looking for the number of new sites visited by the user. Finally, the keystroke analysis characteristic aims to capture delays between typed keys to determine user behaviour.

CHARACTERISITC ALGORITHMS

Characteristics included in the prototype used simplistic statistical algorithms to determine whether or not they were viable to profile user behaviour. The sets of data that are chosen aimed to collect as many variations of user activity as possible and the algorithms chosen attempted to be similar in order to enable comparisons between the characteristics.

The running applications characteristic uses the processes running on the machine to determine the number of processes currently running as well as whether the process has been previously run on the machine. The first metric, variance in number of processes aimed to determine if the user runs the normal amount of applications on their machine. This involves collecting the number of processes every 30 seconds and comparing it to the last hour of collected data (i.e. 120 30-second collections). The new data would be determined anomalistic if it were 2.5 standard deviations away from the mean of the past hour's data. This same standard deviation algorithm is used for the second metric, number of new processes, where instead of using the number of processes the number of new processes is tallied over the current hour and compared to the last 6 hours of data to determine if the user is using the applications they often interact with. The number of windows characteristic uses the same standard deviation algorithm but instead uses the variance in number of windows to determine if the user is interacting with the user interface as normal. Due to the lack of standardization in browser history, not all browsers are able to provide the number of times a user has revisited a page. Instead, the websites viewed characteristic determined the number of new sites a user visits. Like the number of new processes algorithm, the websites viewed characteristic also worked by tallying the number of new sites visited per hour and comparing it to previous data. This allowed the system to capture behaviour for those that revisit the same set of sites and those that often browse to different sites.

The application performance characteristic focused on CPU and memory usage from running applications. This used the same standard deviation algorithm used by the number of processes metric but also includes a rolling average, upper and lower limits, and a weighting system to determine anomalistic behaviour. The rolling average algorithm captured a cumulative average by taking the new value and averaging it with the previous cumulative average, providing a lifetime view of behaviour for the application rather than the past hour. The upper and lower limits recorded the lifetime high and low, for CPU usage and memory usage, for each application which also provides a lifetime view. Capturing these peaks

over time also provides accurate data of anomalistic behaviour. All three algorithms are then combined with a weighing system with a 3-point limit. Each algorithm is assigned the following point values depending on their ability to accurately determine anomalies: standard deviation – 0.5, rolling average – 1, limits – 2. Points would accumulate over 3 data collections and if the threshold of 3-points was breached, a system would trigger anomalous behaviour.

Keystroke analysis was performed using digraph delays, as introduced by Bergadano et al. (2003), where if a user were to type "test" the system would record the delays between the user typing 't' - 'e', 'e' - 's' and 's' - 't'. Once a user retypes a digraph, the system is able to compare with previous delays of the digraph to determine if it is slower or faster than usual. For each digraph, the system recorded the past 100 delays and using the same standard deviation algorithm, determined if the new delay was 2.5 deviations away from the mean of past delays. The system would then check if there had been more than 5 abnormal delays in a row, over all digraphs, or 3 abnormal delays in a row, for the typed digraph, to determine if there is an anomaly. Unlike other characteristics in the system, the keystroke analysis data collection occurred in real-time but was analysed during the 30-second collections.

After each analysis was performed, characteristics were combined using a weighing algorithm by placing characteristics into two separate categories: user-related and application-related. User-related characteristics are defined by a user performing an activity directly affects the data collected by the characteristic. This includes keystroke analysis, websites viewed and number of windows. Application-related characteristics are those that are affected by the execution of the application and include application performance and number of processes. Like the application performance characteristic, each category is weighted. User-related characteristics are assigned 2-points and application-related characteristics assigned 1-point. If after 3 data collections, the point threshold exceeds 5-points the system flags the user as an intruder. This favours both decreased false-positives and faster detection times. False-positives are decreased as it requires single characteristics to trigger at least twice to flag intruders, while intruders with significantly different behaviour will be flagged almost instantly due to triggering multiple characteristics at once.

SYSTEM ARCHITECTURE

The prototype implemented a design that is often used in both host and network-based IDS. This is a generic design that gathers input data, analyses it through a set of engines and outputs a result. Figure 1 shows the architecture of the implemented system. The engine that gathers the input is known as the data collection engine. This engine gathers data for all of the implemented characteristics and sends the data to the relevant characteristic engine for analysis. In this prototype, data was collected from keystroke inputs (keystroke analysis), browser history files (websites viewed), the Windows performance monitor (CPU usage, memory usage) and the Win32 API (applications running, number of windows).

Each characteristic in the system has a separate analysis and profiling engine. Each engine employs both a misuse and anomaly detection algorithm to determine intrusions, however the focus of the prototype is on anomaly detection. The detection algorithms check incoming data against prior data stored in the user's profile to determine if it is significantly different. The results of this analysis are sent to the profiling engine so that authorised data is included in the user profile.

Results of each characteristic are then sent to the data mining engine where characteristics are combined and weighted to determine if there has been a significant change in behaviour, again checking against prior data from the user profile. If the data mining engine determines that the new data is anomalistic, the alert/action engine responds by locking the user out of the system. After the analysis of data has been completed, the system schedules itself to recollect data after a 30 second interval. Characteristic data was collected per interval or on event. In the case of the prototype behavioural IDS, the keystroke analysis characteristic collected data on event (per keystroke) and all other characteristics were gathered at a 30 second interval. If a characteristic was collected on event, data would be collected and analysed within its own characteristic engine but not combined and weighted in the data mining engine until the next collection.

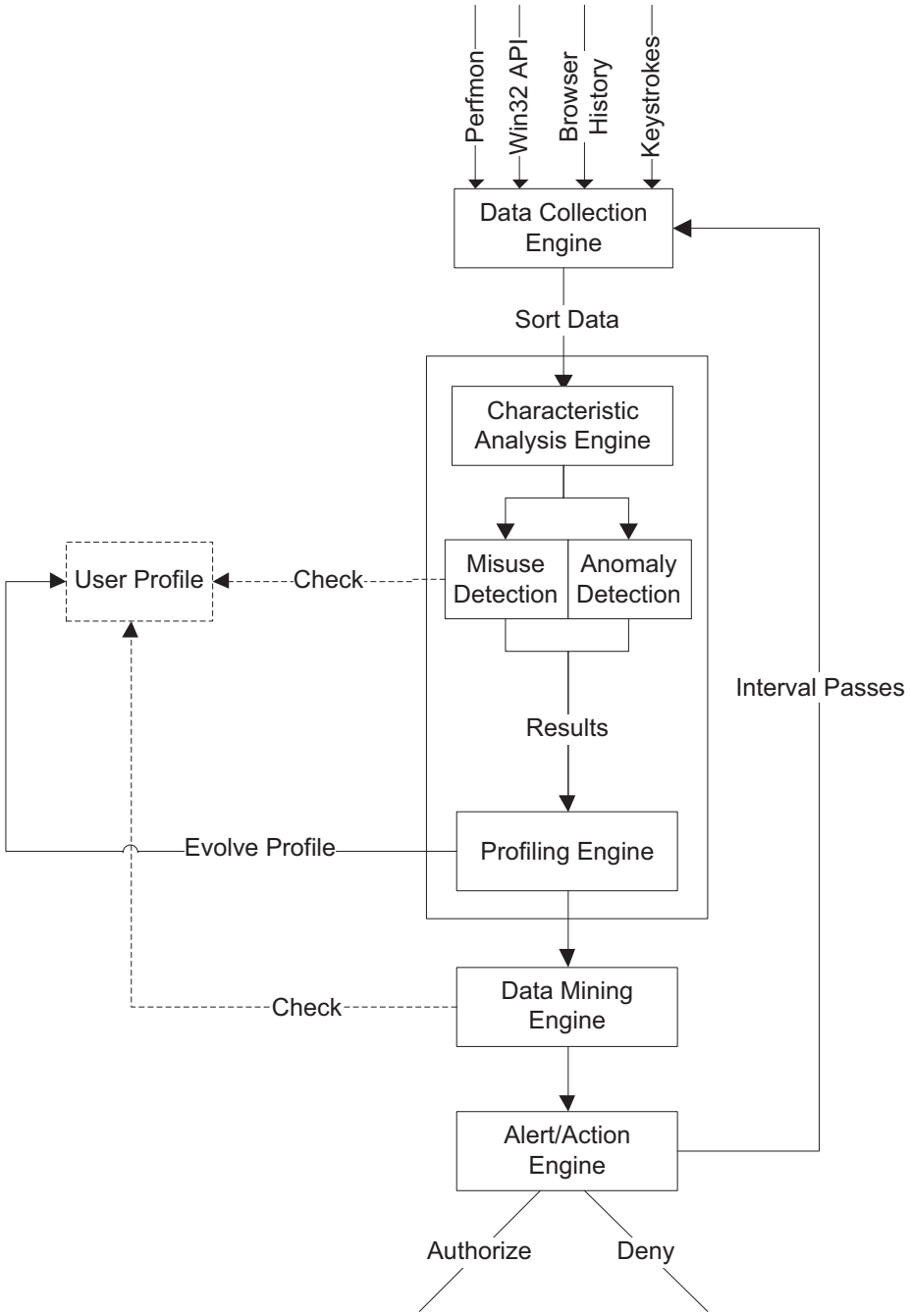

Figure 1: The Architecture Of The Implemented Behavioural IDS

USER STUDY

A user study was conducted to determine the learning ability of each characteristic and to find the time taken to detect an intruder, both for each characteristic and when characteristics are combined. The user study consisted of 11 users who installed and ran the behavioural IDS in the background for 10 days. This gives a total of 28880 30-second data collections for learning the user's initial profile. Each user was asked to define the primary role of their machine to determine if characteristics were best suited to different user roles. It was found that users fit into 4 main categories. The spread of user roles is shown in table 1. In addition to a spread of user roles, the behavioural IDS was tested on various versions of the Microsoft Windows operating system ranging from XP to 7, both 32 and 64-bit.

Number of users	Primary machine role
3	Web browsing Using a web browser to surf the Internet.
1	Gaming Playing 3D games.
3	Office / University work Using office based tools such as word processors, spread sheets, presentations, etc.
4	Power user A combination of all of the above.

Table 1: The Spread Of Different User Roles Used In The User Study

The experiment was broken into two separate stages: learning and detection. These stages are required by a behavioural IDS to function properly and output accurate results. Firstly the learning stage where the system aggregates and analyses sufficient data to model the user with reasonable accuracy, and secondly, the detection stage where the IDS applies its detection algorithms over the learned user model(s). Thus the experiment evaluates two things: the system's ability to learn an adequate user model, and subsequently the system's ability to use anomaly detection methods over the user model to accurately detect intrusions.

The learning phase shows how each characteristic is able to learn a user's behaviour by showing a decrease of learning anomalies over time. During the beginning of a learning phase, a characteristic is expected to output a high number of learning anomalies as the profile contains no previously learned data. As the profile is populated with data, learning anomalies decrease as there is now a sufficiently large profile of user behaviour to compare incoming data. When combining characteristics, learning anomalies should be decreased over the entire learning period due to the number of anomalies needed to flag an intrusion. The learning phase assumes a no-intrusions policy meaning that all observed behaviour is not malicious and is that of the authorised user of the machine.

The learning ability of the system is critical for the system to be able to determine intrusions. Without a properly learned user model the system will perform poorly with slow detection times or high false-positive rates. If characteristic exhibits no learnt behaviour curve, it may become impossible to accurately detect a user. For example, a characteristic showing a constantly high amount of learning anomalies may result in a false-positives that quickly trigger the system and a characteristic showing a constantly low amount of learning anomalies may result in lack of or slow detection times.

The detection phase attempts to introduce realistic intrusions into the system and to stress test each of the individual characteristics, and then finally tests its ability to detect a wrong user. This shows how quickly the system is able to react to both realistic and simulated intrusions. Stress tests on individual characteristics involved different tasks for each characteristic as outlined in table 2. Since introducing a new user often triggers multiple characteristics, the stress tests are designed to show the performance of each characteristic in a best case scenario while the new user test aims to test the combination of characteristics. The new user test involves placing a user in front of a logged in session of a learned user and asking them to use the machine to perform any task they wish to perform, simulating a hijacked login.

Characteristic	Stress test
Applications running	Introduce an abnormal amount of processes used by the user. Introduce an abnormal amount of new processes.
Number of windows	Open an abnormal amount of windows.
Application performance	Stress processes to high CPU and memory usage that normally have low usage, or vice versa.
Websites viewed	Visit a high number of new sites. Visit no sites but leave detection system running.
Keystroke analysis	Use abnormally high keystroke delays to trigger the system as much as possible.

Table 2: Stress Test Tasks For Each Characteristic

RESULTS

During the learning phase, each characteristic's performance is shown by a decrease of learning anomalies over time to show how the system learns the user's typical behaviour. These learning anomalies can also be seen as a type of false-positive, where an anomaly is detected when the authorised user of the profile is using the machine, however, the user model is incomplete as it is still being trained and therefore the decrease of the false-positives over time shows learnt behaviour as the profile populates. The results show that all characteristics were able to successfully characterise user behaviour by showing a decreasing learning curve however, some showed a steeper decrease in learning anomalies than others, showing that the learning time needed to profile a user could be decreased and that the characteristic is very suited towards that user role.

The CPU usage characteristic produced the highest number of false-positives compared to other characteristics but still showed a decreasing learning curve that converges to fewer than 50 false-positives per two days near the end of the learning period. Power users in this case were less successful with significantly high learning anomalies showing that the 10-day learning period may not have been sufficient to model the variability of such users' activity. The high number of false-positives may also relate to the characteristic being more application-related than user-related, where the characteristic is influenced more by the application than the user's behaviour and therefore having more seemingly random behaviour. False-positive rates for the CPU characteristic ranged from 6.73% for a power user machine to 0.6% from a web browsing machine. Figure 2 shows the spread of learning anomalies for the CPU usage characteristic.

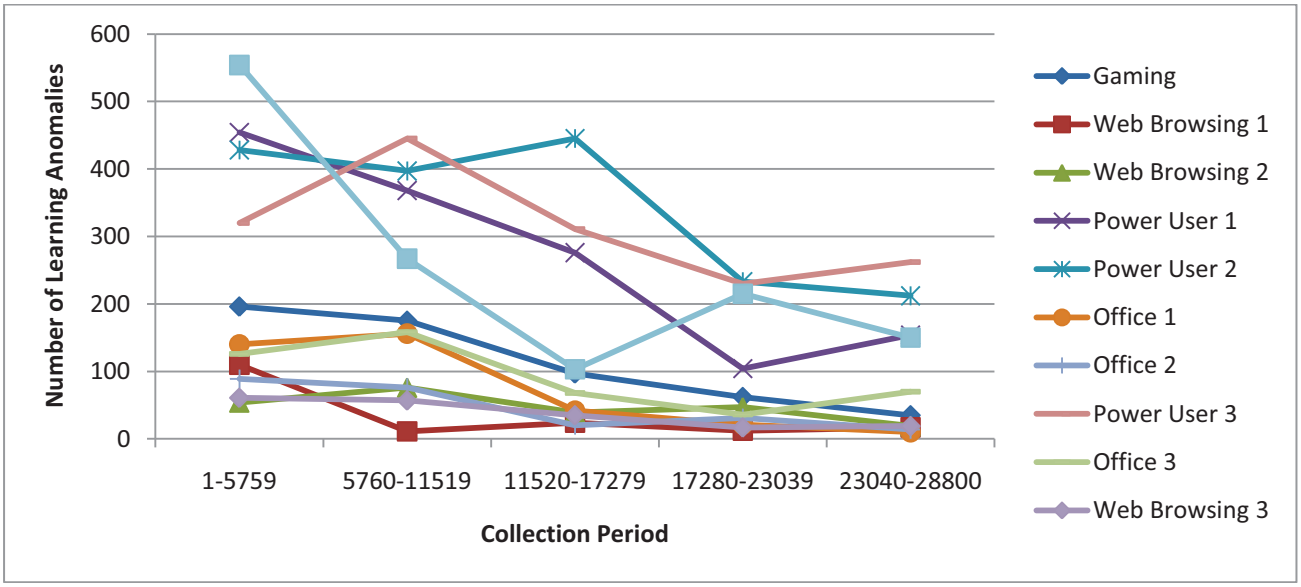

Figure 2: Decreasing Learning Anomalies Over The Learning Period For The CPU Usage Characteristic

The memory usage characteristic is one of the least successful, perhaps because of the high number of learning anomalies in the CPU usage characteristic. The memory usage characteristic shows nearly no learning curve, meaning that it was unable to profile any behaviour. This may be due to processes being able to jump from low usage to high usage by opening a particular file. False-positive rates for web browsing machines appear to be higher than power users, ranging from 0.58-1.18% vs. 0.88-1.03%. This is likely due to web browsers significantly varying their memory usage depending on the number of windows or tabs open. Figure 3 shows the lack of decreasing anomalies for the memory usage characteristic, an example of a characteristic that is unable to capture behaviour.

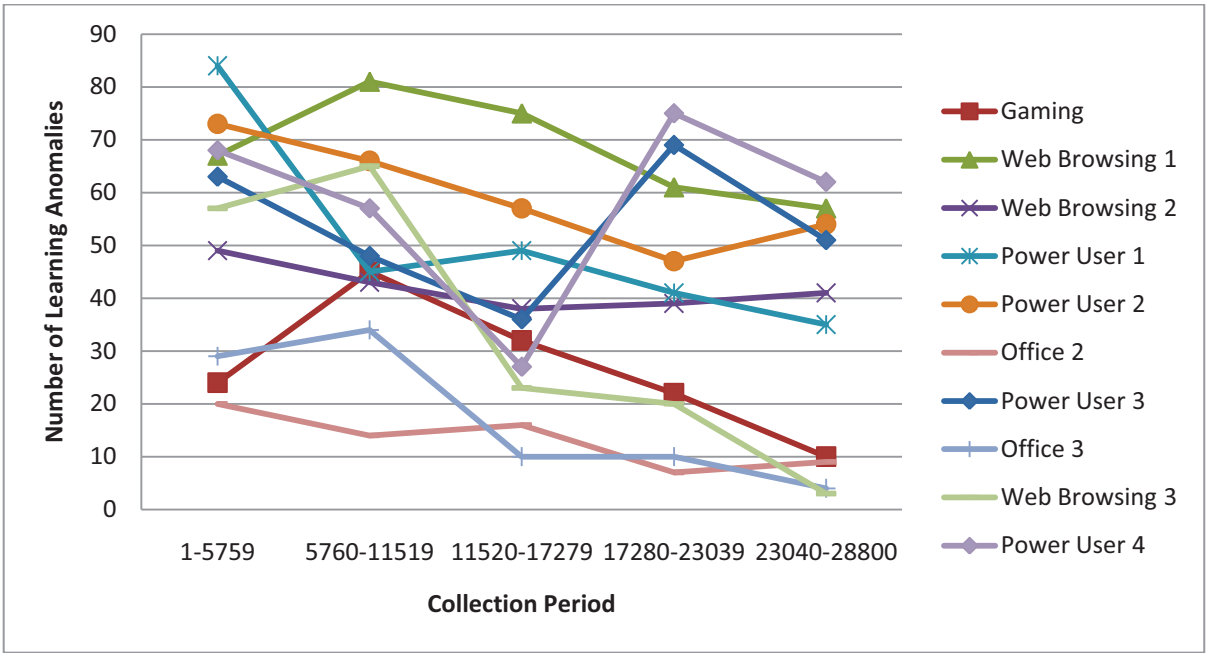

Figure 3: Stable Amount Of Learning Anomalies Over The Learning Period For The Memory Usage Characteristic

The number of processes, number of windows and websites viewed characteristics provided the fewest learning anomalies but all provided decreasing learning curves and near zero learning anomalies by the end of the learning period. In particular, the websites viewed and number of windows characteristic showed a peak in learning anomalies part way through the learning period. This is explained by a user exhibiting a particular behaviour, for example browsing the same sites, then exhibiting another behaviour, for example browsing new sites. Until the user exhibits both behaviours, the system is unable to learn both behaviours however in the case of these characteristics, learning anomalies would once again decrease after the mid-period peak occurs. False-positive rates for each of the three characteristics were less than 0.10%. For each characteristic, power users again have the highest false-positive rates except in the case of the websites viewed characteristic where web browsing machines equalled false-positive rates of power users. Figure 4 shows the websites viewed characteristic and an example of increased learning anomalies part way through the learning period.

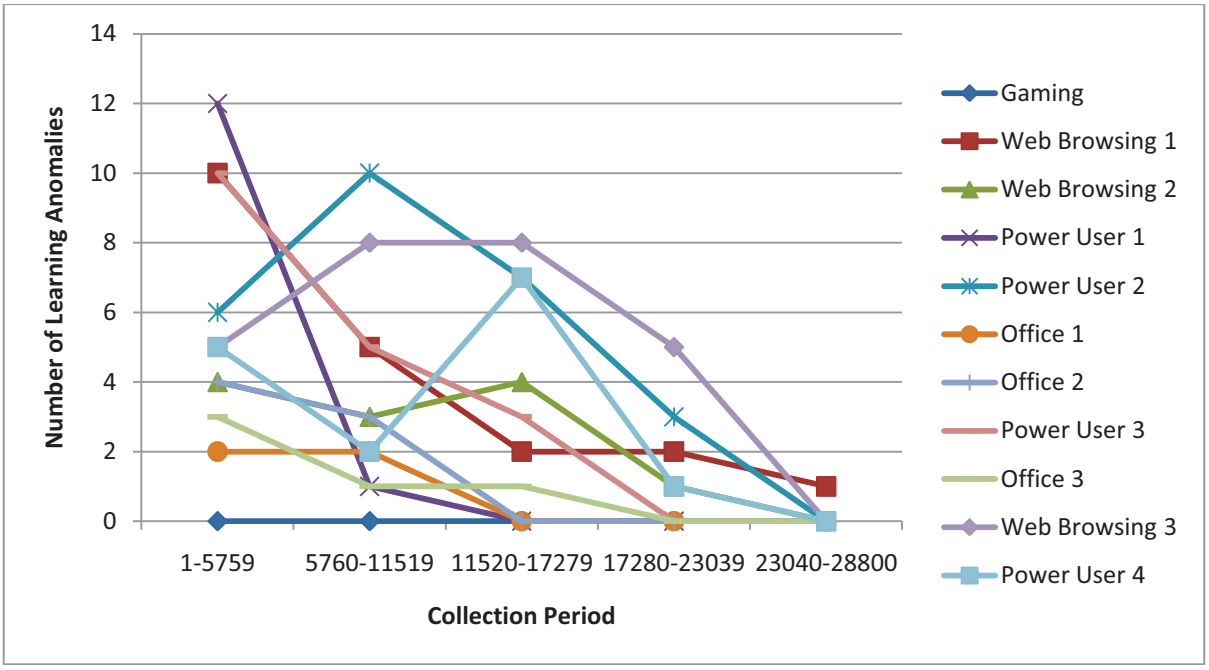

Figure 4: Spread Of Learning Anomalies During The Learning Period For The Websites Viewed Characteristic

The most successful characteristic in terms of both learning time and ability to profile behaviour is the keystroke analysis. This is likely due to the characteristic being directly influenced by the user. The steep learning curve shows that

the characteristic shows rapid learning of behaviour which may allow a reduced learning period. While the characteristic still output 50 alerts per two days at the end of the period, in a worst case scenario, further optimisations may decrease false-positives. Power users had the highest false-positive rates of up to 2.36% but other machines had an average false-positive rate of 0.44%. One anomalistic result stems from the gaming machine where the false-positive rate is 3.84% and there is no apparent learning curve. This can be explained by a gaming user reacting to what they see on the screen rather than performing a particular task. Instead, the game is influencing the user's behaviour due to its interactive nature making behaviour more random than usual. Figure 5 shows the steep learning curve of the keystroke analysis characteristic.

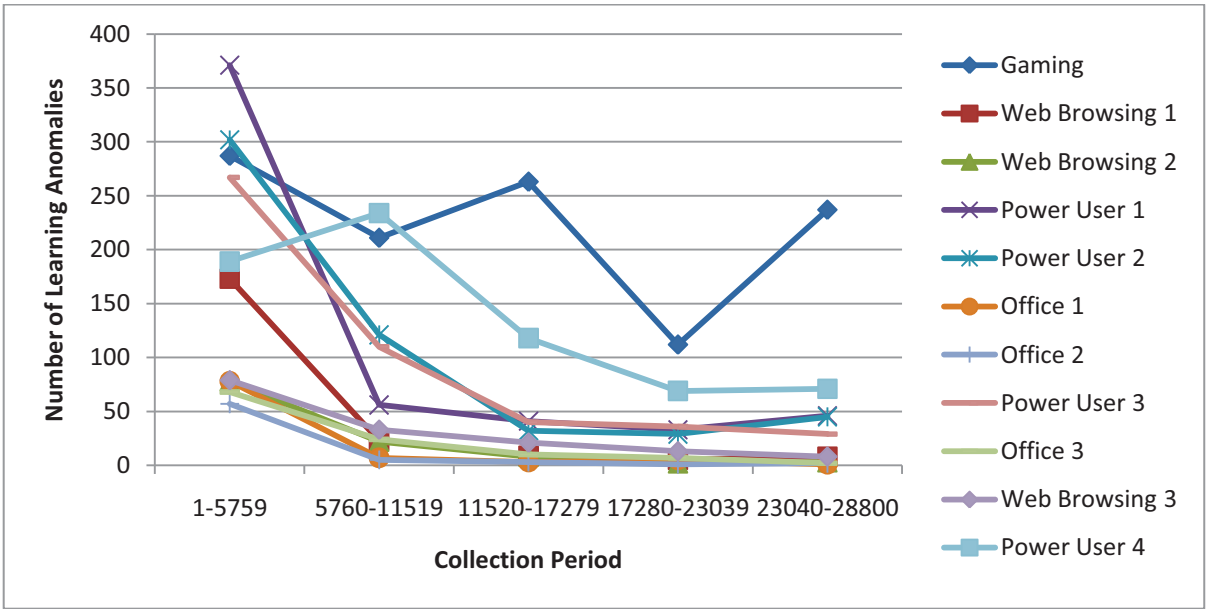

Figure 5: Decreasing Learning Anomalies Over The Learning Period For The Keystroke Analysis Characteristic

When combining characteristics within the data mining engine, the learning curve starts to flatten and the system begins to output the same number of false-positives per day. The flattened curve does not necessarily mean a successful result, in fact, at first glance the system is showing no sign of learning behaviour. The false-positives output by the system are likely biased towards the less successful, application-related characteristics that output significantly more learning anomalies than user-related characteristics. If these less successful characteristics are removed, the system will converge to zero false-positives per two days, showing that the system has learnt typical user behaviour in the allocated learning time. The flattened curve also means that it is possible to implement a better combination algorithm to consistently decrease false-positives. The false-positive rates for the combination of characteristics are lower than most of rates for individual characteristics. Power users showed the highest rate of 0.64% followed by the gaming user at 0.58%. Web browsing and office machines had false-positive rates as low as 0.03%. Figure 6 shows the spread of false-positives from the system when combining characteristics.

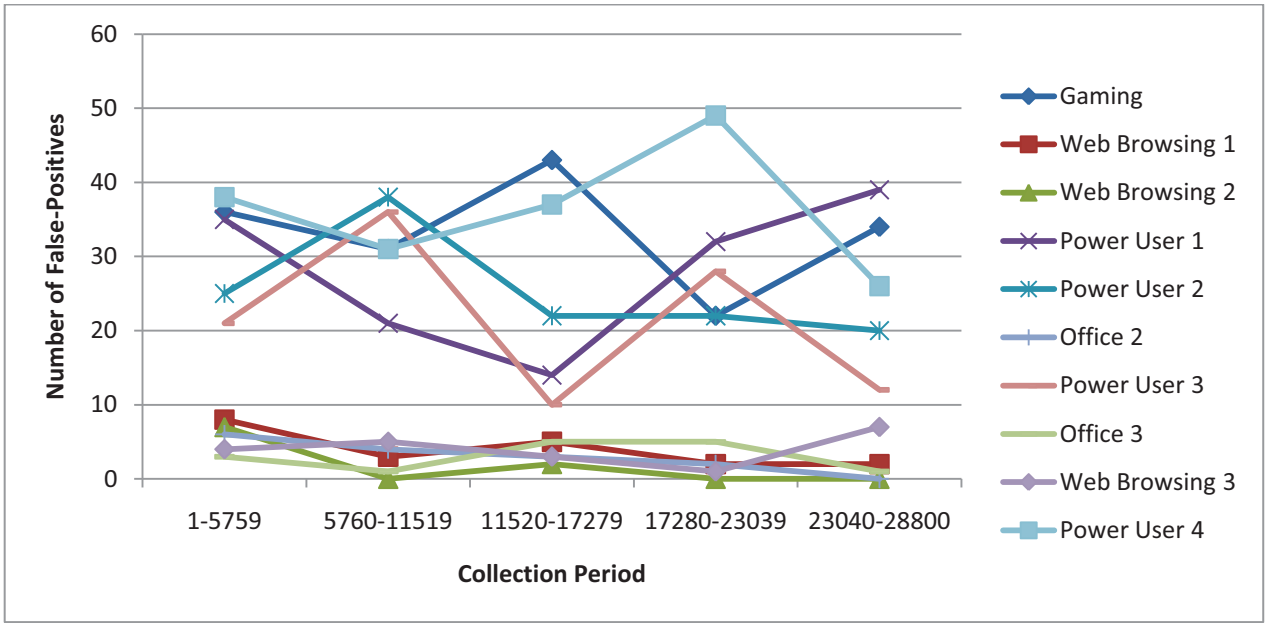

Figure 6: The Spread Of False-Positives During The Learning Period When Combining Characteristics

During the learning phase, on average, the memory usage characteristic had the highest false-positive rate of 3.05%, followed by CPU usage at 2.64%. Although keystroke usage showed a promising learning curve, its average false-positive rate was 1.29% biased by the unlearned gaming machine behaviour. Other characteristics, websites viewed, number of windows, number of processes had false-positive rates of less than 1 percent: 0.04%, 0.01% and 0.02% respectively. The combination of characteristics had the slightly higher average false-positive rate of 0.40%.

The detection tests allowed the testing of individual characteristics, by using abnormal stress tests, and combined characteristics, by using new user behaviour. The most successful individual characteristic was keystroke analysis which was able to detect anomalies within an average of 120 seconds (4 data collections). Other user related characteristics had slightly higher detection times of approximately 150 seconds (5 data collections). Users that specialised in a specific characteristic, for example web browsing machines and the websites viewed characteristic, often had higher, but expected, detection times due to the range of behaviours and more extreme usage than other user roles. For example, power users often took longer to trigger the number of processes and windows characteristic and web browsing users took longer to trigger the websites viewed characteristic. Application related characteristics took approximately 180-210 seconds (6-7 data collections), on average longer than user related characteristic. When placing a new user in front of the machine, the combined characteristics decreased detection time to just 90 seconds (or 3 data collections) for nearly all user roles with unoptimised learning and detection algorithms. Figure 7 shows the detection times for all intrusion tests. The intrusion tests show that not only does the combination of characteristics increase detection performance but user related characteristics are more relevant than application related characteristics when profiling user behaviour.

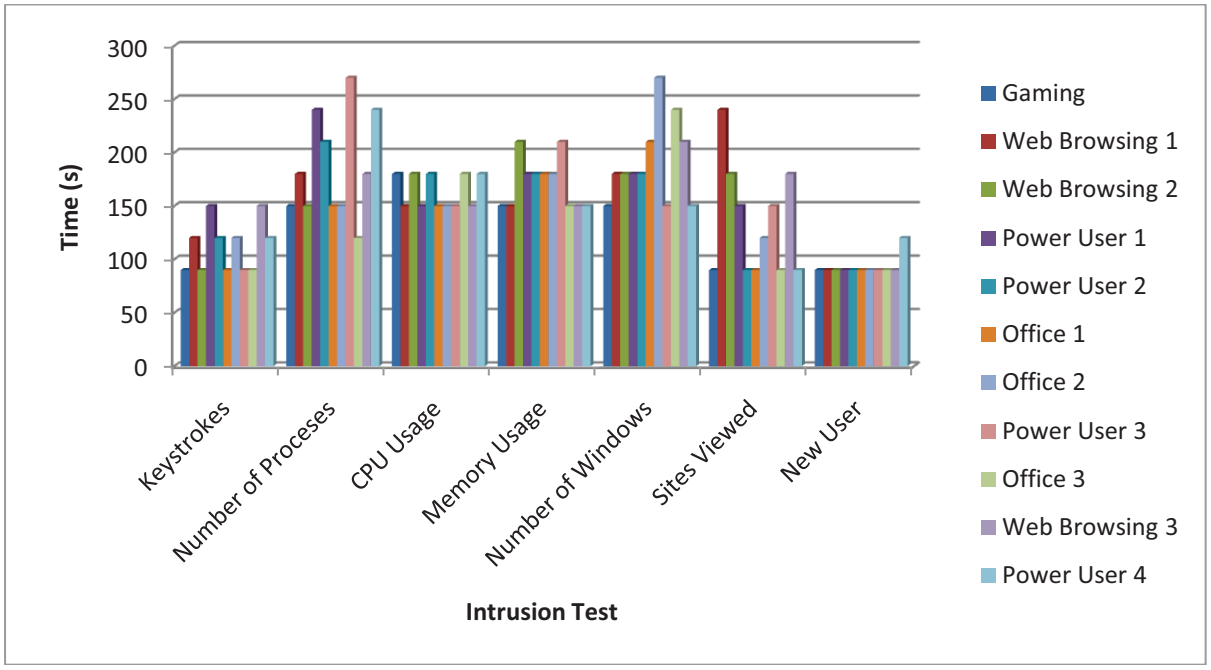

Figure 7: The Time Taken To Detect An Anomaly During Intrusion Tests For All Users

Figure 8 clearly shows that the characteristics that are more related to applications produce higher detection times while characteristics that preference user activity produces quick detection. The new user test, or combination of characteristics, produces the lowest average time of 90 seconds with the other user related characteristics, keystroke usage and websites viewed producing the second lowest average time of 120 seconds. Interestingly, the GUI characteristic, number of windows, which had hoped to be mostly user related, had a high average detection time of 180 seconds showing that the characteristic may require further optimization. All application related characteristics, CPU usage, memory usage and number of processes had the worst performance with CPU usage performing the best of them at an average of 150 seconds. Memory usage and number of processes had average detection times of 180 seconds.

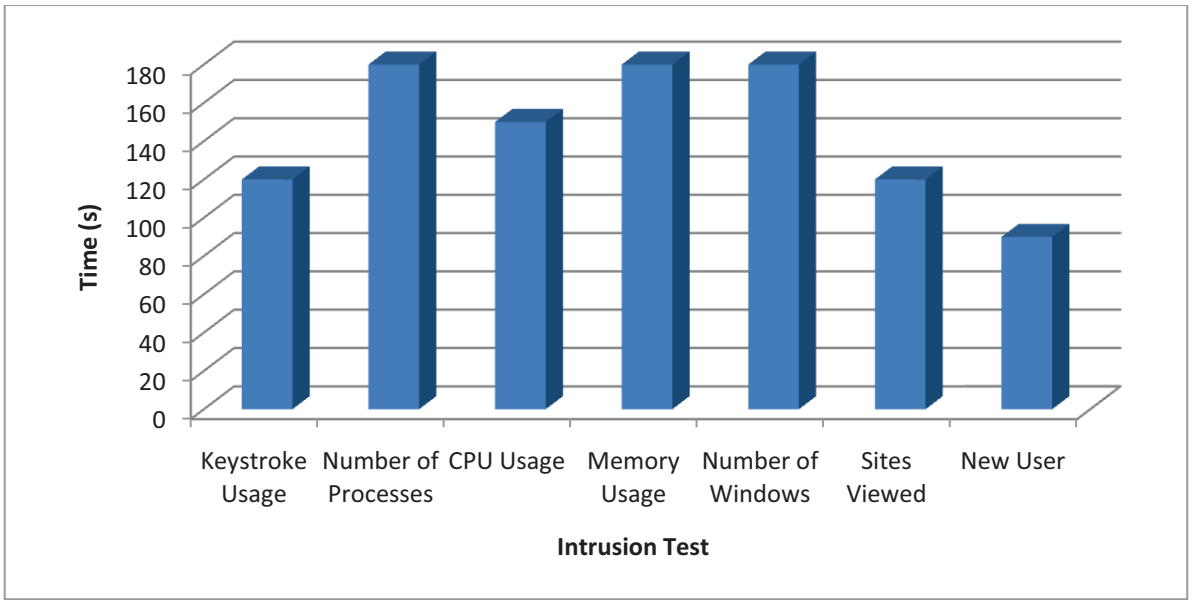

Figure 8: Average Time Taken To Determine Intrusion Per Characteristic

CONCLUSION AND FUTURE WORK

This paper reports on a prototype behavioural intrusion detection system to determine anomalies in user behaviour. It is shown that the combination of characteristics is able to improve detection times when compared to singular characteristics and that user-related characteristics are often better at learning user behaviour than application-related characteristics. Overall, we have shown that the prototype system has potential to accurately profile users. In future work we aim to optimise the algorithms used for learning and detection to improve system performance in terms of both detection rates and resource usage. Further characteristics will also be introduced unobtrusively so that the system is able to continuously re-authenticate users, including mouse dynamics and GUI interaction. It is also planned to introduce further analysis to current characteristics. For example, keystroke analysis may involve looking at typed keystrokes to determine a user's typographical errors. It is also possible to group actions from multiple characteristics to determine patterns in behaviour, or to group applications by category to determine if a user switches their default text editor or web browser.

REFERENCES

Ahmed, A. A. E. & Traore, I. 2005, 'Anomaly intrusion detection based on biometrics,' in Information Assurance Workshop, 2005. IAW '05. Proceedings from the Sixth Annual IEEE SMC, pp. 452-453.

Anderson, D., Frivold, T. & Valdes, A. 1995, 'Next-generation Intrusion Detection Expert (NIDES) A Summary,' SRI INTERNATIONAL Computer Science Laboratory.

Anderson, J. P. 1980, 'Computer Security Threat Monitoring and Surveillance,' James P. Anderson Company.

Balajinath, B. & Raghavan, S. V. 2001, 'Intrusion detection through learning behaviour model,' Computer Communications, vol. 24, no. 12, pp. 1202-1212.

Bergadano, F., Gunetti, D. & Picardi, C. 2003, 'Identity verification through dynamic keystroke analysis,' Intelligent Data Analysis, vol. 7, no. 5, pp. 469-496.

Boies, S. J. 1974, 'User Behaviour on an Interactive Computer System,' IBM Systems Journal, vol. 13, no. 1, pp. 2-18.

Cabrera, J. B. D., Lewis, L. & Mehra, R. K. 2001, 'Detection and classification of intrusions and faults using sequences of system calls,' SIGMOD Rec., vol. 30, no. 4, pp. 25-34.

Cheung, D. W., Kao, B. & Lee, J. 1998, 'Discovering user access patterns on the World Wide Web,' Knowledge-Based Systems, vol. 10, no. 7, pp. 463-470.

Cockburn, A. & McKenzie, B. 2001, 'What do web users do? An empirical analysis of web use,' International Journal of Human-Computer Studies, vol. 54, no. 6, pp. 903-922.

Denning, D. E. 1987, 'An Intrusion-Detection Model,' Software Engineering, IEEE Transactions on, vol. 13, no. 2, pp. 222-232.

Ellis, D. R., Aiken, J. G., Attwood, K. S. & Tenaglia, S. D. 2004, 'A behavioural approach to worm detection,' in Proceedings of the 2004 ACM workshop on Rapid malcode, ACM, Washington DC, USA, pp. 43-53.

Eugene, S. 2008, 'James P. Anderson: An Information Security Pioneer,' IEEE Security & Privacy, vol. 6, no. 1, pp. 9.

Forrest, S., Hofmeyr, S. A., Somayaji, A. & Longstaff, T. A. 1996, 'A sense of self for Unix processes,' in Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on, pp. 120-128.

Gu, G., Cardenas, A. A. & Lee, W. 2008, 'Principled reasoning and practical applications of alert fusion in intrusion detection systems,' in Proceedings of the 2008 ACM symposium on Information, computer and communications security, ACM, Tokyo, Japan, pp. 136-147.

Hofmeyr, S. A., Forrest, S. & Somayaji, A. 1998, 'Intrusion detection using sequences of system calls,' Journal of Computer Security, vol. 6, no. 3, pp. 151.

Imsand, E. S. & Hamilton, J. A. 2007, 'GUI Usage Analysis for Masquerade Detection,' in Information Assurance and Security Workshop, 2007. IAW '07. IEEE SMC, pp. 270-276.

Julisch, K. & Dacier, M. 2002, 'Mining intrusion detection alarms for actionable knowledge,' in Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, Edmonton, Alberta, Canada, pp. 366-375.

Lane, T. & Brodley, C. E. 1997, 'An Application of Machine Learning to Anomaly Detection,' in Proceedings of the 20th National Information Systems Security Conference, pp. 366-380.

Li, L., Sui, S. & Manikopoulos, C. N. 2006, 'Windows NT User Profiling for Masquerader Detection,' in Networking, Sensing and Control, 2006. ICNSC '06. Proceedings of the 2006 IEEE International Conference on, pp. 386-391.

Lindqvist, U. & Porras, P. A. 2001, 'eXpert-BSM: A Host-based Intrusion Detection Solution for Sun Solaris,' in 17th Annual Computer Security Applications Conference, IEEE Computer Society, New Orleans, Louisiana, pp. 2-40.

Leggett, J., Williams, G., Usnick, M. & Longnecker, M. 1991, 'Dynamic identity verification via keystroke characteristics,' International Journal of Man-Machine Studies, vol. 35, no. 6, pp. 859-870.

McKinney, S. & Reeves, D. S. 2009, 'User identification via process profiling: extended abstract,' in Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies, ACM, Oak Ridge, Tennessee, pp. 1-4.

Monrose, F. & Rubin, A. 1997, 'Authentication via keystroke dynamics,' in Proceedings of the 4th ACM conference on Computer and communications security, ACM, Zurich, Switzerland, pp. 48-56.

Pannell, G. & Ashman, H. 2010, 'User Modelling for Exclusion and Anomaly Detection: A Behavioural Intrusion Detection System,' in 18th Intl. User Modeling, Adaptation, and Personalization, Springer, Berlin, pp. 207-218.

Pusara, M. & Brodley, C. E. 2004, 'User re-authentication via mouse movements,' in Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, ACM, Washington DC, USA, pp. 1-8.

Revett, K., Jahankhani, H., Magalhães, S. T. & Santos, H. M. D. 2008, 'A Survey of User Authentication Based on Mouse Dynamics,' in Global E-Security, 4th International Conference, ICGeS 2008, Springer, London, UK, pp. 210-219.

Revett, K. 2009, 'A bioinformatics based approach to user authentication via keystroke dynamics,' International Journal of Control, Automation and Systems, vol. 7, no. 1, pp. 7-15.

Shavlik, J. & Shavlik, M. 2004, 'Selection, combination, and evaluation of effective software sensors for detecting abnormal computer usage,' in Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, Seattle, WA, USA, pp. 276-285.

Smaha, S. E. 1988, 'Haystack: an intrusion detection system,' in Aerospace Computer Security Applications Conference, 1988., Fourth, pp. 37-44.

Stolfo, S. J., Hershkop, S., Hu, C.-W., Li, W.-J., Nimeskern, O. & Wang, K. 2006, 'Behaviour-based modeling and its application to Email analysis,' ACM Trans. Internet Technol., vol. 6, no. 2, pp. 187-221.

Tan, K. 1995, 'The application of neural networks to UNIX computer security,' in Neural Networks, 1995. Proceedings., IEEE International Conference on, pp. 476-481.

Tauscher, L. & Greenberg, S. 1997, 'How people revisit web pages: emprical findings and implications for the design of history systems,' International Journal of Human Computer Studies, vol. 47, no. 1, pp. 97-138.

Umphress, D. & Williams, G. 1985, 'Identity verification through keyboard characteristics,' International Journal of Man Machine Studies, vol. 23, no. 3, pp. 263-273.

Vizer, L. M., Zhou, L. & Sears, A. 2009, 'Automated stress detection using keystroke and linguistic features: An exploratory study,' International Journal of Human-Computer Studies, vol. 67, no. 10, pp. 870-886.

Yampolskiy, R. V. & Govindaraju, V. 2008, 'Behavioural biometrics: a survey and classification,' Int. J. Biometrics, vol. 1, no. 1, pp. 81-113.