A Deterministic Profiling Approach to support Behaviour Anomaly Detection

H Rukmal M Fernando (rukmal_fernando@yahoo.com)
Department of Computing, Informatics Institute of Technology, Sri Lanka

Vidura Saminda Gamini Abhaya (vidura@dialogsl.net)
Department of Computing, Informatics Institute of Technology, Sri Lanka

Abstract

Tracking Behavior Anomalies for identifying user impersonations is a field of active research. While various approaches have been used, it remains true that this process is still not deterministic or controllable enough for practical use.

We present a new approach that aims to make the process of profiling behavior patterns of users deterministic by using a Rule-base to assign scores to various events. This also makes such a system highly controllable by permitting real-time modifications to the rule base. By combining asymmetric encryption with an XML based representation of behavior information, we have also made the profiling process a one-way task, where a compromised profile does not reveal sensitive information about the profiled user.

We developed and tested a software system to evaluate the concept, and through this we have demonstrated how it is possible to ensure that reporting and monitoring components distributed over a network are not compromised, and our entire system functions on an XML based open, secure protocol. We also discuss the applicability of network packet capture as a source of behavior information. Our results indicate that we have been able to develop a deterministic, secure and extensible framework for distributed behavior anomaly detection.

1. Introduction

Authentication – the process of establishing the identity of a user - is perhaps the most fundamental security activity for Information Systems. The concept of the 'identity' of the user provides the basis for other security processes like authorization and secrecy.

Authentication security is complemented by a host of technologies like passwords, smart cards and biometrics and also advanced protocols e.g. [14], Kerberos. Users' mistakes, software vulnerabilities, attacks and a host of other risks are however threats to password based systems. Protocols like Kerberos are themselves known to have vulnerabilities [17], as do Biometrics [10]. All this leads to an ever-increasing risk of impersonation.

2. User Profiling

Monitoring user's behavior patterns, modeling such in a profile and detecting anomalies which may indicate impersonations is a concept that has established itself [1], [9]. The most notable approach taken has been command sequence matching [3], [5], [6] and system call monitoring [8]. Previous work has also focused almost entirely on the Unix environment, except for e.g. using CPU usage [12] and PerfMon audit data [16] in Windows environments.

Various techniques have also been used to represent behavior profiles and to detect anomalies. These include Artificial Neural Networks (ANN) [2], Support Vector Machines (SVM) [13], Bayesian Networks [12],[15], Data Mining [18] and many others.

We see many problems with these approaches, primarily:

a. Artificial Intelligence systems offer very little controllability due to the fact that the system's 'intelligence' depends on the learning/ training data, and that there are very few factors, if at all, which can be modified while the system is in operation (e.g.: An Expert System's Rules can be modified, but a Neural Network's weights typically cannot be).

b. Preserving the privacy of behavior information is near impossible and the threat to privacy is generally proportional to the richness of the behaviour information stored in the behavior 'profile'.
c. Some approaches, notably Statistical Analysis, Data Mining and related techniques require enormous data bases of past behavior information.
d. Detecting patterns of behavior and highlighting statistical anomalies is usually a highly processor intensive operation.
e. Finally, we are yet to see a solution that protects components which monitor and report behavior information against tampering.

Additionally, we propose that modern networked environments benefit more from modeling user's activities across network resources, rather than limiting to metrics such as CPU or network usage levels or command and process execution. These metrics are seen as ones that even an impersonator might generate, and metrics like CPU usage do not permit real-time anomaly detection.

As will be elaborated later, our profile describes activities such as protocol in use and operation performed etc, and we propose that such a rich profile is necessary for effective anomaly detection.

3. A Novel Approach

Our work [7] proposed a novel approach that aimed to overcome the weaknesses pointed out previously, and focused on

a. the use of network packet capture as a behaviour information source
b. a deterministic profiling approach using neither activity logging, nor Statistical or AI techniques
c. creating a reusable and extensible architecture for future work

The strategy for achieving each of these objectives is below:

3.1. Network packet capture for user behaviour monitoring

WinPcap [19] was used as a packet capture driver, and the packet is decoded, where fields like the source and destination IP Addresses and ports are used to map each packet to a user. We achieved this by identifying the ports that each application has opened on the host, and then mapping the packet to the application, and the application to its owner.

Our attempt however focused on HTTP and Server Message Block (SMB) protocols. With HTTP, we report the actual HTTP GET request's target as the destination, and the packet owner is identified using the above mapping process.

Where NTLM or Kerberos authentication is used, the mapping is somewhat more complex with the SMB protocol given that packet exchange does not involve user applications, and a dynamic UserID is assigned to each SMB user which must be mapped to an actual user account. With the simpler dialects of SMB however, this mapping is straightforward with the SMB header specifying the username. We further map SMB commands to a collection of high-level operations (Read , Delete etc.) to facilitate our scoring algorithm.

Our approach has thus been to identify information about the logical action specified by each packet and map this to a user. This Profile Event is then used as an input tuple to the profiling process.

$$\text{Profile Event} = \{\text{User, Protocol, Operation, Target, DateTime}\} \quad (1)$$

3.2. A deterministic profiling approach

A rule-based specifies various events that the system 'listens' to, and assigns a risk factor based on the protocol, operation, target and time combination. Thus, for example a higher significance can be given to an SMB Delete operation on a hitherto un-accessed host rather than a HTTP GET operation from a known host.

While monitoring, the system evaluates each event, and the score obtained via the rule-base is aged and averaged with the existing score. This averaged score is an indication of the probability of that event happening for the given user, and this algorithm is presented later as equation 3.

While this may seem as a very simple technique, such a rule set effectively enables an expert to use common knowledge and corporate security policies and focus the system to look for specific events that may be expected during a malicious impersonation attempt.

It also makes an administrator's task more involved and also makes the system highly configurable.

Protocol	Target	Operation	Time-band	Score
HTTP	Existing	Read	Any	0.0
SMB	Existing	Read	New	0.1
SMB	New	Delete	New	0.9

Figure 1: Sample rule set

The above is an example of how different events may be given a higher 'risk' or 'threat' score.

Once evaluated, the event is added to the profile, which is hierarchical in structure viz.:

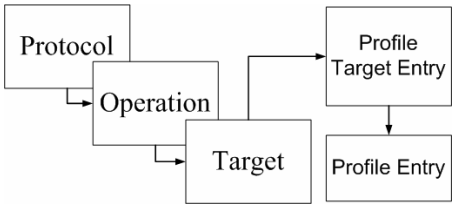

Figure 2: Profile structure

Here the arrowheads point to the 'parts' that build up the 'whole'. The "Profile Target Entry" (PTE) has 29 "Profile Entry" (PE) records of four Time-band categories: Time of day, day of week, week of month and month of year. Appendix A outlines the time attributes we tracked. Each Profile Entry is a tuple of the form below:

$$\text{Profile Entry} = \{\text{First time-stamp, Last time-stamp, Count, Score}\} \quad (2)$$

Each event thus creates a PE tuple, and adds it for each matching entry in the PTE. Where an entry is found, the two scores are aged and aggregated using the algorithm defined below. The existing entry and the new entry also provide some useful parameters about the behaviour.

Age (A)	First(Old) – Now
Active age (A_A)	Last(Old) – First(Old)
Inactive age (A_I)	Now – Last(Old)
Frequency (F)	Count / Active age

Figure 3: Parameters from the profiling event entries

These values are used in 'aging' the existing score (S_{old}) against the event score (S_{new}) to reduce its overall impact. The score (S) is thus obtained

For $(A_A > A_I)$,

$$S = \{(A_A / A * S_{old} * C) + S_{new}\} * (C + 1) / C \quad (3)$$

Otherwise, (3) may be presented as

$$S = \{((1 - A_I / A) * S_{old} * C) + S_{new}\} * (C + 1) / C$$

Here, C ($\neq 0$) denotes the total previously detected event count. For $(C = 0)$, $S = S_{new}$.

The profiling process is also carried out so that events are evaluated against the profile, but new uncertain events are recorded as a short term attribute. Whether this short term entry qualifies to become a normal, long-term element of the profile is decided by two more variables, the minimum number of events that must be detected ($Watch_{min}$) and the duration within which this number of events must be detected ($WatchAge_{max}$).

Thus, the predicate IsNormalPattern may be expressed as

$$\text{IsNormalPattern} = (C \geq Watch_{min}) \text{ AND } (A_A \leq WatchAge_{max}) \quad (4)$$

Further, the profile belongs to either the "Learning" state where it accepts all new behaviour patterns or the "Monitoring" state where the above predicate governs adding new patterns. The profile automatically changes its state when a predetermined minimum number of EventTargetEntry tuples are present, or a minimum profile age has been observed. It is also possible to switch the profile between these two states as needed.

This profile structure is inspired by probability trees and the hierarchical nature can be easily represented by an eXtensible Markup Language (XML) object.

3.3. A reusable and extensible framework for impersonation detection

Three objectives drove the framework development:

a. Permit the behaviour source to provide any kind of behaviour information as long as it can be expressed as a Profile Event tuple (equation 1).
b. Allow a reasonable ability to 'tune' the system to various type of events and responses, thus allowing it to be adapted to varying needs
c. Permit the behaviour monitoring to occur in a network of computers, thus with storing and sharing of profiles, in a secure manner

Objective (b) has been supported by the $Watch_{min}$ and $WatchAge_{max}$ parameters. Objective (c) is supported by a mutual authentication phase between monitoring components, as described in the following section.

3.4. Preserving privacy of behavior information

While preserving privacy in data mining has been studied, e.g. [18], how privacy can be achieved in probability representations of behavior has not been studied much.

A simple solution was devised thus:

Each profiling agent, at it's startup, gets access to the server component's public key $K(S)^+$, and caches it. On each action it detects, it encrypts the target using this public key, and stores this information in the profile. This provides means of allowing each agent to update score information, but does not allow the profile to be decoded easily to reveal its content.

We do, however, realize that it is possible to check the profile to see if an entry exists for a potential or suspected target, and it is also possible to delete segments of the profile.

The profiles are stored encrypted using a short-term symmetric key, and the key itself is stored using the protected storage service, on Windows computers. Agents also use the protected storage service to store their last used session key, and this is presented to the server at the next connection phase to reasonably assure that the agent has not been compromised.

Provided that the agent service is protected by a separate user account, and that protected storage, and the disk location of the profiles, are secured with reasonable OS security, we expect this solution to provide an adequate balance between usability and trust.

4. Implementing a Prototype

A prototype was implemented to test the concepts expressed. As outlined earlier, we targeted the Windows 2000 environment and used WinPcap as the packet capture driver. The packet decode is carried out by a Visual C++ based COM component, and the events are reported to a C#.Net based Windows Service. The components form the Monitoring Agent which carries out all profiling and response activities. A separate Server component was created to coordinate the sharing of profiles and facilitate dynamically configuring the agent. A Monitoring Console was also developed to allow monitoring and configuring the system centrally.

4.1. Profile format

The hierarchical nature of the Profile lends itself easily into an XML representation. This also gives added advantages by making the entire system extensible and language independent and also permits encryption of arbitrary fields.

4.2. Mutual authentication

Asymmetric encryption is used to exchange a symmetric session key that encrypts the communication between the various entities. This also permits mutual authentication provided the private key can be made secure (e.g. submitted to the server via a smart card or token on startup) and the agent is able to securely cache a used session key and present it on its next session key negotiation phase. A UML Sequence model summarizes this process.

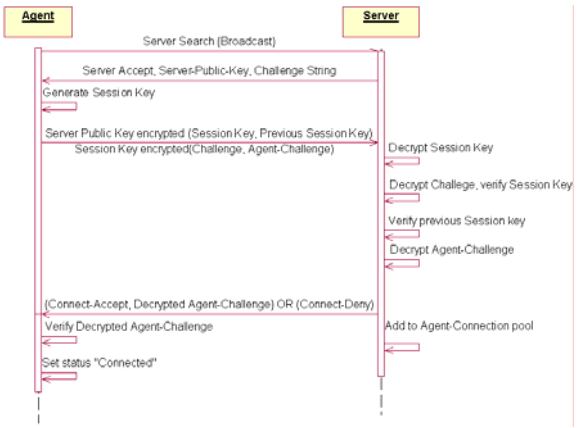

Figure 4: Key exchange mechanism

4.3. Performance sensitive architecture

All three components follow the same multithreaded operation model with separate threads listening for messages and a separate thread for the main processing.

5. Behaviour Information sources

It is notable that the majority of previous work seems to have focused on application usage, CPU utilization, network traffic levels and other performance oriented metrics as behaviour information sources. The authors are however of the view that such performance metrics are not suitable for several reasons:

a. A valid user and an impersonator could both use the same application set and command set to access different information resources and thus evade such detection.
b. Increased use of GUI systems reduces the value of monitoring application usage, and system call monitoring may be too low level. Additionally, modern Operating Systems like Windows 2000 have system calls that can be executed remotely and sometimes without being connected to a particular user session. Correlating these to a user session is difficult at best.
c. Monitoring of local command execution does not indicate in any sensitive remote information resources are being accessed.
d. CPU usage etc can vary heavily over short time periods, and thus must be monitored at very high frequencies. It also makes real-time detection impossible, as e.g., daily CPU usage minutes may be experienced in an hour of impersonation.

We propose network behaviour monitoring as a better source of behaviour information for these reasons:

a. Accessing network based information resources can be easily tracked with this approach
b. It provides real-time information on possible anomalous behaviour

It is also noted that the use of Artificial Intelligence (AI) and Machine Learning techniques (e.g. Artificial Neural Networks, Bayesian Networks, Support Vector Machines, *k*-Nearest Neighbor Classification etc) has also been explored. These are again seen by the authors as not ideal for security related systems, as these techniques are somewhat non-deterministic and susceptible to being tainted by random behaviour of the legitimate user. Thus, we propose that a Rule-based approach could overcome this problem by providing a deterministic approach to classifying behaviour as normal or not.

6. Analysis of Prototype

The approach discussed was an attempt at user profiling using a network based behaviour information source and was also a novel attempt that did not use large logs of activities. The technique discussed creates profiles for which an example profile segment is indicated below.

This was also then coupled with two known facts:

a. The user indeed frequently carried out the said activity
b. No other area of the profile has entries for such significant behaviour

```
<Profile username="domain\user1" FirstEvent="3/25/2005 2:37:51 PM"
        LastEvent="4/25/2005 8:15:20 AM" EventCount="161" >
  <protocol protocolname="smb">
    <target id="192.168.1.6">
      <operation operationName="Read" >
        <entry="1" first="632500569188906250" last="632500569202031250"
                   count=137 score=0.2 />
        <entry="4" first="632500569188906250" last="632500569202030190"
                   count=151 score=0.1 />
        <entry="6" first="632500569188906250" last="632500569202031235"
                   count=99 score=0.1 />
        <entry="20" first="632500569188906250" last="632500569202027540"
                    count=104 score=0.1 />
      </operation>
      ...
```

Figure 5: Profile sample segment

We find the profiling process to be thus quite successful in recording behaviour patterns. It must however be noted, that the Rule set used in scoring the events is also a significant factor in deciding which type of events must be given significance.

We see a considerable advantage in choosing network events as the source of behaviour information for this system, as this means that access to possibly sensitive information resources and heterogeneous behaviour across networks can be tracked with ease.

As far as disadvantages or limitations are concerned, the key limitation seen is that the success of the profiling process depends on the attributes that have been chosen, the granularity of the EventTargetEntry and the Rule set used for scoring. It is however, a very simple matter of, e.g. returning event information that considers custom application protocols, increasing the resolution of the EventTargetEntry or adding more rules, that is required for improving the performance.

It is true that this solution depends on WinPcap for packet capture. WinPcap's advanced features like the Network Packet Filter for kernel level packet filtering [4] gives it a performance advantage. The prototype was also developed in .Net due to its support for rapid prototyping and performance has been quite satisfactory. It will however be required that either the existing prototype is optimized further, or rewritten in e.g. Visual C++ for improved performance.

7. Conclusions and Future Work

The primary conclusion that can be drawn is that the work is promising of the hypothesis that network monitoring is a rich source of information for behaviour anomaly detection. While the prototype could not be tested extensively enough to conclusively demonstrate detection of impersonations, analysis of the profile indicates that the profile does indeed contain significant subset of recurring behaviour patterns, and is therefore capable of achieving the stated objectives.

Other techniques need to be explored as well (e.g. monitoring Network Session setup for information on a user's network activity) that can give the same richness of information without the use of packet capture.

The solution also stands to possibly benefit from some machine learning techniques and can probably use a better Rule-base engine for evaluating the scoring. However, this system must be viewed as only a companion technique to other proven approaches both in behaviour anomaly detection and in detecting attacks.

8. References

[1] J. P. Anderson, "*Computer Security Threat Monitoring and Surveillance*". 15 Apr. 1980. (Technical paper).

[2] Cannady, J.: "*Artificial Neural Networks for Misuse Detection*", School of Computer and Information Sciences, Nova Southeastern University, USA, 1998.
URL: http://csrc.nist.gov/nissc/1998/proceedings/paperF13.pdf

[3] S. Coull, J. Branch, B. Szymanksi and E. Breimer, "*Intrusion Detection: A Bioinformatics Approach*", in proc. 19th Annual Computer Security Applications Conference, 2003.

[4] L. Degioanni, M. Baldi, F. Risso, and G. Varenni, "*Profiling and Optimization of Software-Based Network-Analysis Applications*", in proc. 15th Symposium on Computer Architecture and High Performance Computing, 2003.

[5] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "*A Sense of Self for Unix Processes*" in proc. IEEE Symposium on Security and Privacy, Los Alamitos, CA, 1996, pp. 120-128.

[6] S. Forrest, S. A. Hofmeyr, and A. Somayaji, "*Intrusion Detection using Sequences of System Calls*", URL: http://cs.unm.edu/~forrest/publications/int_decssc.pdf, 1997

[7] H. Rukmal M. Fernando, "*BigBrother: Recording, Monitoring and Matching User Behaviour Patterns to Support Authentication of Network Users*" (BSc Thesis), Informatics Institute of Technology, Sri Lanka, 2005.

[8] A. K. Ghosh, A. Shwartzband, and M. Shatz, "*Learning Program Behaviour Profiles for Intrusion Detection*", in proc. 1st USENIX Workshop on Intrusion Detection and Network Monitoring, Apr. 1999, pp 51-62.

[9] Hollmén, J.: "*User Profiling and classification for fraud detection in mobile communication networks*" (Doctoral thesis), Department of Computer Science and Engineering, Helsinki University of Technology, Finland, 2000.

[10] C. J. Hill, "*Risk of Masquerade Arising from the Storage of Biometrics*" (BSc Thesis), Department of Computer Science, Australian National University, Nov. 2001.

[11] T. Lane, and C. E. Brodley, "*Detecting the Abnormal: Machine Learning in Computer Security*." in proc. National Information Systems Security Conference, Baltimore, USA, 31 Jan. 1997.

[12] Laskey, K., Alghamdi, G., Wang, X., Barbará, D., Shackelford, T., Wright, E., Fitzgerald, J.: "*Detecting Threatening Behaviour using Bayesian Networks*", George Mason University, USA, 2004.
URL:
http://ite.gmu.edu/~klaskey/papers/BRIMS04_InsiderThreat.pdf

[13] Y. Lioa, "*Windows NT User Profiling with Support Vector Machines*", 2002. URL:
http://ailab.das.ucdavis.edu/~yihua/research/LiaoStd02.pdf

[14] R. M. Needham, and M. D. Shroeder, "*Using Encryption for Authentication in Large Networks of Computers*", Communications of the ACM, volume 2, number 12, pp 993 – 999, Dec. 1978.

[15] Schiaffino, S. N. and Amandi, A.: "*User profiling with Case-Based Reasoning and Bayesian Networks*", ISISTAN Research Institute, Facultad de Ciencias Exactas, Universidad Nacional del Centro de la Pcia. de Buenos Aires, Argentina, 2000. Open Discussion Proceedings, IBERAMIA-SBIA 2000 - Atibaia, Brazil - November 2000 – pp. 12 – 21.
URL : http://www.exa.unicen.edu.ar/~sschia/beramia00.pdf

[16] J. Shavlik, M. Shavlik and M. Fahland, "*Evaluating Software Sensors for Actively Profiling Windows 2000 Computer Users*" in proc. 4th International Symposium on Recent Advances in Intrusion Detection, 2001.

[17] W. Stallings, *Network Security Essentials: Applications and Standards*, Pearson Education, Inc, USA, 2000.

[18] Vaidya, J. S. (2004, August). "*Privacy Preserving Data Mining over Vertically Partitioned Data*" (Doctoral thesis), Centre for Education and Research in Information Assurance and Security, Purdue University, USA.

[19] WinPcap downloaded from http://winpcap.polito.it/

9. Appendix A

Each event applies to one attribute from each of the following time-band categories. (e.g.: 12.45 pm on 12th September 2006 applies to time bands 4, 9, 16 and 26). The granularity can be modified to change the system's sensitivity to time-variance of events.

	Category	Attribute
1	Time of day	Dawn
2		Morning
3		Daytime
4		Afternoon
5		Evening
6		Late night
7	Day of week	Sunday
8		Monday
9		Tuesday
10		Wednesday
11		Thursday
12		Friday
13		Saturday
14	Week of month	1
15		2
16		3
17		4
18	Month of year	January
19		February
20		March
21		April
22		May
23		June
24		July
25		August
26		September
27		October
28		November
29		December