User Impersonation Detection: A Behaviour Anomaly based Deterministic Approach

H Rukmal M Fernando (rukmal_fernando@yahoo.com)

Supervised by: Mr. Vidura Saminda Gamini Abhaya (vidura@dialogsl.net)

Department of Computing – Informatics Institute of Technology - Sri Lanka

Abstract

With organizations becoming evermore dependent on Information Systems for their day-to-day activities, there is a growing realization of the critical need to secure these systems given the reality that malicious attackers as well as insiders account for a large proportion of misuse.

Authentication utilizes a variety of technologies like passwords, smart cards, and protocols like Kerberos. In spite of these, impersonation is still known to occur, and there is no proper method for detecting a breach that may result from password disclosure or smart card loss etc.

We present a study on monitoring a user's network behaviour to derive a profile, and how distinguishing such from an impersonator's can serve as a means of detecting and overcoming the impersonation problem. Network Packet Capture was explored for such monitoring and a configurable profiling engine was developed, that uses a Rule based deterministic approach. This is complemented by an extensible and secure framework for distributed impersonation detection.

1 Introduction

Authentication – the process of establishing the identity of a user - is perhaps the most fundamental security activity for Information Systems. Authentication security is complemented by a host of technologies like passwords, smart cards and biometrics and also advanced protocols e.g. [11], Kerberos. Users are however known to write passwords down and share them with others. Vulnerabilities like buffer overflows and sniffing, along with 'Human Engineering' - posing as a trusted entity to elicit sensitive information - are other threats to password -based systems. Protocols like Kerberos are themselves known to have some vulnerabilities [12], as do Biometrics [8].

All this leads to an increased risk that a user may be impersonated on an Information System. The fundamental issue here is that the impersonator is now assumed to be a valid user, and so his actions are not questioned or monitored by the system, and no techniques exist for easily detecting such an occurrence.

2 User Profiling

Building a 'profile' of a user's behaviour and using this as the basis of detecting attacks is a concept that has existed for some time [1] and the most notable approach taken has been command sequence matching [2], [4], [7] and system call monitoring [5]. Previous work has also focused almost entirely on the Unix environment, except for e.g. using CPU usage [10] and PerfMon audit data [13] in Windows environments.

It is notable that the majority of such 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 over a network, which is again undetectable

c. Local execution monitoring does not indicate accessing remote information resources

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 Neighbour 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.

3 A Novel Approach

Our work [6] proposed a novel approach that aimed to overcome the two 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 [14] was used as a packet capture driver, and the packet is decoded, with fields like the source and destination IP Addresses and ports used to map each packet to a user. This can be achieved by identifying the ports that each application has opened on the host, and the mapping the application to its owner. Our attempt has 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 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. SMB commands are further mapped to a collection of high-level operations (Read 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.

Profile Event = {User, Protocol, Operation, Target, DateTime} (1)

3.2 A deterministic profiling approach

A rule-based technique assigns a score for each event to describe how significant an anomaly would be. 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 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 to specific scenarios.

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.:

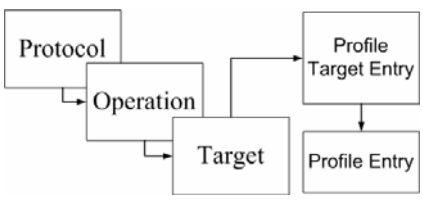

Figure 2: Profile Architecture

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. Each Profile Entry is a tuple of the form below:

Profile Entry = {First time-stamp, Last time-stamp, Count, Score} (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 ($A_A > A_I$) and ($C > 0$)

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

For ($A_I > A_A$) and ($C > 0$)

$$S = \{[1 – (A_I / A)] * S_{old} + S_{new}\} * [(C + 1)/C] \quad (4)$$

Here, C 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

$$\text{IsNormalPattern} = (Watch_{min} \geq C) \text{ AND } (WatchAge_{max} \leq AA) \quad (5)$$

Further, the profile belongs to either the "Learning" mode where it accepts all new behaviour patterns or the "Monitoring" mode where the above predicate governs adding new patterns. The profile automatically changes its status 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.

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.

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 eXtensible Markup Language (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.

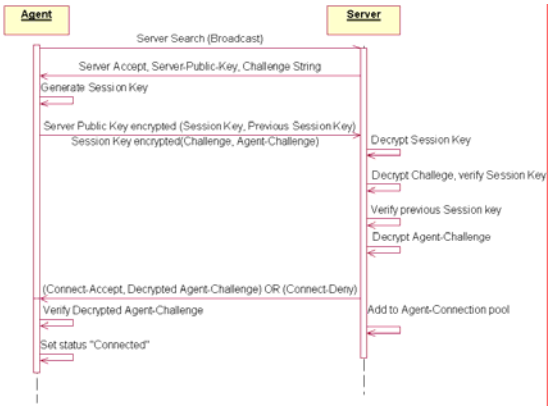

Figure 4: Key exchange mechanism

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 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.

```
<Profile Username="domain\user01" FirstEvent="5/13/2005 12:56:14 PM"
LastEvent="5/14/2005 4:04:31 PM" EventCount="1233" IsLearning="False" >
  <Protocol ProtocolName="smb">
    <Target ID="192.168.1.3">
      <Operation OperationName="Connect" >
        <Entry Index="1" First="632515857748759687"
Last="632516714450523528" Count="165" Score="0.006133029" Status="1" />
        <Entry Index="2" First="632515857748759687"
Last="632515907127884167" Count="163" Score="0.07535887" Status="1" />
        <Entry Index="3" First="632516714149123198"
Last="632516714450523528" Count="2" Score="0.07600001" Status="0" />
        ...
```

Figure 5: Profile sample segment

This was coupled with two known facts:

a. The user has indeed frequently carried out the said activity

b. No other area of the profile has entries for such significant behaviour

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 [3] 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.

6 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.

7 References

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

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

[3] 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.

[4] 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.

[5] 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

[6] 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.

[7] 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.

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

[9] 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.

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

[11] 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.

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

[13] 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.

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