A conceptual intrusion monitoring architecture and thoughts on practical implementation

P.S.Dowland and S.M.Furnell

Network Research Group, Department of Communication and Electronic Engineering, University of Plymouth, United Kingdom
e-mail : sfurnell@plymouth.ac.uk

Abstract

The paper presents a conceptual description of the Intrusion Monitoring System (IMS) architecture, which is designed to facilitate detection of system penetration and other anomalous activity in a networked environment. The architecture is based upon eight functional elements, distributed between a monitoring host and a series of monitored client systems. The discussion also considers how the approach could be integrated within the Windows NT environment.

1 Introduction

The concept of real-time intrusion monitoring has been of interest in the IT security domain for a number of years, with the original idea having been proposed by Denning (1987). Such an approach is valuable as a means of combating a number of classes of system abuse, including penetration by unauthorised persons and misuse of privileges by registered users. In addition, abuse may be perpetrated by malicious software, such as viruses and Trojan Horse programs. Although a number of IDS have been developed (Mukherjee et al. 1994), these have generally targeted large systems or specific domains (e.g. military). Commercial implementations are generally restricted in their monitoring functionality. However, the increasing interconnection of corporate systems, coupled with reported increases in computer abuse incidents (CSI 1999), suggests that the use of more advanced intrusion monitoring functionality would be advantageous. This paper presents the conceptual architecture of the Intrusion Monitoring System (IMS), which aims to detect anomalous activity in a networked environment, followed by consideration of how to realise the approach in practice under Windows NT.

2 Intrusion Monitoring System overview

The IMS architecture was originally proposed by Furnell (1995) and is based upon the concept of a centralised *Host* handling the monitoring of one or more *Clients* on local workstations. The Clients collect the required data relating to system activity and respond to any suspected intrusions detected by the Host. Monitoring is based upon the comparison of current activity against two categories of stored information, namely user behaviour profiles and generic intrusion rules. These approaches are common to other intrusion monitoring architectures, such as the Intrusion Detection Expert System described by Lunt (1990). User profiles could conceivably hold a range of identification, authentication and behavioural information relating to registered users. Examples of potential characteristics would include system access times and locations; typical levels of system resource utilisation; application and file usage; methods of user interaction; and biometrics (i.e. physiological and behavioural characteristics). Biometric monitoring is considered to be particularly appropriate to prevent impostor penetration and a number of options exist that could be employed in this context, including keystroke analysis, face recognition and voice recognition (Cope 1990). Other well-known biometrics, such as fingerprint recognition, are less strongly favoured in the IMS context, as less opportunity exists to integrate them in a manner that is transparent (and, hence, non-intrusive) to the legitimate user.

Some classes of intrusion or misuse can be trapped without identifying departures from historical patterns of user behaviour. The occurrence of some events will be suspicious in themselves and, therefore, the system requires a means to monitor for these as well. Examples of generic indicators would include consecutive access violations, out of hours access, account overuse / simultaneous access, use of inactive accounts and extensive use of help systems. While none of these alone would provide sufficient indication to state that an intrusion was in progress, the combination of two or more could be more persuasive. In the IMS context, these attack signatures would be represented via Intrusion Rules that, if satisfied, would increase the alert status of the system.

3 The IMS Architecture

Anomaly Detector. The *Anomaly Detector* analyses activity for suspected intrusions, comparing it against the behaviour profile of the current user's (claimed) identity, as well as against the generic intrusion rules. The detector is comprised of further sub-modules, each handling specific monitoring tasks (e.g. keystroke analysis, tracking of resource usage etc.). The detector maintains an *alert status table*, with entries existing throughout the life of each user-initiated session or process to indicate the level of detected anomalies and thereby the confidence of a potential intrusion. This information would be examined and updated each time activity data relating to the relevant user / process is analysed. The alert status level would increase in response to departures from the user-specific behaviour profile or the satisfaction of generic intrusion indicators. The level would be reduced after successful challenges or after a sufficient period of normal activity to allow the system to discount the previous anomaly. The alert status level can be linked with the types of activity that a subject is permitted to perform. In this way, a phased reduction of permitted behaviour would occur as the level increases. Sensitive activities / information could, therefore, be denied if doubt exists over the legitimacy of the current user, whilst still allowing more mundane activities to continue. The approach would demand that a maximum alert status threshold be associated with each of the activities or objects that IMS is to control.

Profile Refiner. User behaviour may legitimately alter over time. The *Profile Refiner* aims to provide an automatic means to account for such changes, using neural networks to analyse and recognise behavioural characteristics that might not be apparent to a human observer. In this way, the effectiveness of the system has the potential to improve over time. It might also be possible to determine which of the profiled characteristics provide the best discriminators for each user and thereby establish various levels of behavioural indicator (with the primary level representing the most reliable verifiers). This hierarchy could also be extended to allow for the fact that some characteristics may represent negative indicators (i.e. those that, despite refinement, are found to cause a high level of false rejection).

It would be undesirable for the *Profile Refiner* to utilise data that is later found to be anomalous. Refinement should, therefore, only take place after the termination of non-anomalous user sessions. User-specific profile records would also incorporate a series of flags to indicate whether the individual behaviour characteristics are ready to be used in supervision or still being developed. This will allow a gradual training period to be defined for new user profiles without the IMS continually generating intrusion alerts (the flags would also allow a specific 'refinement only' period to be established for existing profiles that have proved to be inadequate for the legitimate user). The purpose of associating flags with each profile characteristic is so that some degree of monitoring could still continue whilst other aspects are being (re)trained.

Recorder. The *Recorder* handles the short-term storage of user-related activity data during a session and focuses specifically upon the collection of data relating to the profiled characteristics of a given user (e.g. collection of keystroke data in relation to the typing profile). Upon termination, the information will be used as input to the *Profile Refiner*, provided that the session was not considered anomalous. In the event of a proven anomaly, the *Recorder* can discard the session data.

Archiver. The *Archiver* collects data relating to *all* system activity and stores it in a long-term archive, providing a more permanent record of activities and suspected anomalies. The storage occurs regardless of whether sessions / processes are regarded as anomalous and details of all security relevant events are archived. Such events include login failures, intrusion alerts, authentication challenges and suspended sessions. However, in order to conserve storage space, it may be desirable to only record details of certain types of event. The *Archiver* is therefore configurable to suit the preferences of the organisation involved (note that the same would *not* be true for the *Recorder* as this would always need to collect information on any activities for which profile refinement may later occur). The long-term retention period of archived details would be organisation-dependent.

Collector. The *Collector* is responsible for obtaining information on all relevant client-side activity. The module must operate in such a way as to encompass, but be independent of, all system applications. It is envisaged that this could be best achieved by implementation at the operating system (OS) level, such that key events also lead to IMS notification. For example, a significant proportion of data collection could be based around the interception and redirection of selected OS service requests (such as file input / output, application execution, keyboard input). In some cases, data could be obtained directly from audit trail records – as in previous systems, such as Wisdom & Sense (Leipins and Vaccaro 1989.). However, with certain aspects (e.g. keystroke analysis) the required information will not be held by audit trails and implementation may, therefore, require a significant number of OS links.

Whilst this would serve to make this aspect of IMS very system specific, it would be more efficient than attempting to modify individual applications to provide relevant information to IMS.

Responder. This module resides in the Client and responds to anomalies detected by the Host. The operation centres around the continuous monitoring of the alert status transmitted by the Host, with increases in the level triggering appropriate actions. The nature of response at different levels would vary and a detailed discussion of the possible options is beyond the scope of this paper. However, appropriate responses might include: issuing of an explicit challenge for further authentication; recording of details in an intrusion log for later investigation; notification of the system manager; phased reduction of permitted behaviour; locking of the intruder's terminal; and termination of the anomalous session / process.

Communicator. The *Communicator* provides the network communications interface between the Host and the local Client(s). The principal functions include transmitting user and process information to the Host and then subsequently keeping the Client(s) informed of the current alert status. If implemented in a heterogeneous environment, the Client side would be responsible for resolving any operating system differences that exist within the monitoring domain, so that information could be presented to the Host in a consistent, standardised format. The actual communication could then be handled via a standard sockets approach, with protection provided by a technology such as Secure Sockets Layer (Frier et al. 1996).

Controller. This module allows the operation of the IMS system to be configured. On the Host side, this applies to the *Anomaly Detector* (e.g. behaviour characteristics to utilise, generic rules in operation), the *Profile Refiner* (e.g. frequency of refinement, acceptable thresholds for challenges) and the *Archiver* (e.g. level of detail required, specific events to record or exclude from logging). On the Client side, configuration relates to the *Collector* (e.g. the level of data collection, which could be automatically linked to the characteristics being monitored by the *Anomaly Detector*) and the *Responder* (e.g. the level of response required at each alert status level). These settings would be controlled and recorded through the Host system. Local Client(s) would then be configured at the time of session initiation. Other features would also be provided under the auspices of the *Controller*, including user profile management and update of the generic rulebase.

4 IMS Implementation

Work is currently being conducted to develop an implementation for Windows NT. This requires replacement of the Graphical Identification aNd Authentication (GINA) Dynamic Link Library (DLL) - the interface through which a user can provide his/her identification, typically in the form of a username and password. However, it can be replaced with any desired authentication method (e.g. commercial products are available using fingerprint and faceprint methods). In addition to the GINA replacement, the IMS would also require software to provide the required continuous monitoring, together with a remote security server. The security server would be used to store, maintain and update the user profiles. This server would process all authentication requests together with local system audits and updates to profiles. This role is slightly different to that of a network server, which usually only authenticates requests for access at the beginning of a session. Instead, the security server would be responsible for ongoing authentication throughout a session. A user login would be performed locally (or remotely via a domain controller) and once the user's credentials are confirmed the monitoring program would be loaded to provide continuous user authentication. To prevent tampering, the IMS system would store user profiles remotely on the security server. These would be encrypted and downloaded at login (although for higher security the profiles could be maintained on the server with authentication requests being handled by the server). To ensure monitoring hardware has not been tampered with, a local machine audit can also be initiated together with checks for dependent entries in configuration files or registry keys.

To reduce network traffic, it is envisaged that the user authentication would be performed on the local computer with only warnings or profile updates being fed back to the security server. Under certain scenarios it may be necessary to lock local computers if contact is lost with the security server to ensure an intruder had not removed a computer. However, it should be noted that this creates a weak point and appropriate measures will be needed to prevent a single server stopping the entire network. This could take the form of a backup server, in a similar fashion to a secondary DNS server. Alternatively, the range of facilities available to the user can be restricted until they can be re-authenticated.

Once a user has been authenticated by the replacement GINA DLL, the IMS client would be activated. The IMS client would then check the IMS security server (host) for the users monitoring characteristics and rules, to allow it to select the most appropriate monitoring programs. At this point, the selected characteristics would be loaded and initialised. To ensure ease of implementation and future modification, each distinct monitor program would be implemented as a system DLL. Taking the keystroke analysis example, the monitor program would install a system-wide hook to intercept all keystrokes received by the keyboard buffer and pass these to an analysis algorithm within the DLL. To ease the processing burden, the DLL would pass periodic samples of keystroke activity (either time or quantity based) to be analysed. The results of this analysis would then be passed on to the IMS client program to be either compared to the local copy of the user's profile or to be returned to the IMS Host for remote verification and subsequent action. In the event that the IMS detects a potential intruder, a call can be made to the GINA DLL to provide a request for further user authentication. (e.g. question and answer challenge or biometric identification request). As the GINA DLL provides the login interface for NT, it is impossible to perform a local user login without authenticating via the GINA DLL. This can be used to enforce a variety of security rules. For example, the system may refuse login without the presence of the IMS host, the system may only accept a user once a secondary authentication has been made or the system may disallow local logins if monitoring hardware (e.g. a camera) has been removed.

Under Windows NT, the *Anomaly Detector, Profile Refiner, Recorder* and *Archiver* would be implemented on the IMS Security server (Host) as a software suite. To facilitate future upgrades, each component would be contained within a separate DLL, with a front-end provided through a single executable. Depending on the size of the system being monitored, it may be necessary to distribute the tasks over multiple hosts to cope with the level of data analysis and profile updating. It may also be beneficial to implement these programs as services under NT – ensuring they are loaded at host boot-up. The *Collector* would be implemented as a mediator on the Client, collecting information via hooks that intercept system messages (e.g. keystrokes, mouse movements etc.) and forwarding this information on to the *Communicator*. This would again be held in a DLL, which would be called by the replacement GINA. The *Responder* would be implemented within the GINA DLL and provide a replacement login interface for NT. This would also interface to the *Communicator* and *Collector* for data acquisition and client-host communication. The *Communicator* provides the interface between the client and server IMS software. It would be implemented as a shared DLL (used by both client and host). The *Communicator* would use standard TCP/IP communication, with all data being encrypted using a standard algorithm. The *Controller* provides a management interface to the IMS server software and would be implemented on the host as a single executable linking to the *Anomaly Detector*, *Profile Refiner*, *Recorder* and *Archiver* DLL's**.** Further details of the practical implementation approach can be found in Dowland and Furnell (2000).

5 Conclusions

Intrusion detection systems have the potential to provide an important contribution to system security, protecting against abuse by both external persons and organisational insiders. The IMS architecture represents an example approach and the paper has sought to describe the main functional elements. A system such as IMS is considered to represent a useful addition to Windows NT, which increasingly has a role in enterprise-level IT and, hence, an increasing requirement for strong protection. The paper has provided an indication of how IMS would be realised in the NT environment. The detailed mapping of the IMS approach to the NT architecture is currently in progress.

References

Cope, B.J.B. 1990. "Biometric Systems of Access Control", *Electrotechnology*, April/May: 71-74.

CSI. 1999. "Issues and Trends: 1999 CSI/FBI Computer Crime and Security Survey", USA, March 1999.

Denning, D.E. 1987. "An intrusion-detection model", *IEEE Transactions on Software Engineering*, SE-13(2):222-232.

Dowland, P.S. and Furnell, S.M. 2000. "Enhancing Operating System Authentication Techniques", Proceedings of INC 2000 (3-6 July, Plymouth, UK).

Frier, A.; Karlton, P.; and Kocher, P. 1996. "The SSL 3.0 Protocol", Netscape Communications Corp., Nov 18, 1996.

Furnell, S.M. 1995. *Data Security in European Healthcare Information Systems*. PhD Thesis. University of Plymouth, UK.

Leipins, G.E and Vaccaro, H.S. 1989. "Anomaly Detection: Purpose and Framework", In *Proceedings of the 12th National Computer Security Conference* (USA), 495-504.

Lunt, T.F. 1990. "IDES: An Intelligent System for Detecting Intruders", In *Proceedings of the Symposium: Computer Security, Threat and Countermeasures* (Rome, Italy, Nov. 1990).

Mukherjee, B.; Heberlein, L.T.; Levitt, K.N. 1994. "Network Intrusion Detection", *IEEE Networks 8*, no.3: 26-41.