Comprehensive Event Log Monitoring

A White Paper prepared by

Executive Summary

Comprehensive event monitoring, to date, has been an expensive or time consuming activity for system administrators. However, the benefits of inexpensively being able to monitor a range of event log records, and use them to monitor on and report on system activities promises to enable a whole new range of security related services, and will add value to current Intrusion Detection systems.

The effectiveness and efficiency of event log collection can be undermined by cost blowouts, unstable event collection systems, an inundation of events from a variety or sources, adverse resource utilisation on host systems, lack of a full remote control functionality or a lack of flexibility in the reporting and alerting systems. This paper explores these problems, and proposes as set of functions required to minimise the risk, and maximise the success of, establishing an effective event logging system. The example of “traditional” intrusion detection systems is provided, as a basis to outlining some of the problems which could affect an event logging system.

This white paper aims to provide some ideas on these issues, and provide the concepts behind the release of event logging, Open Source tools, published by Intersect Alliance.

Contact :
George Cora and Leigh Purdie
Intersect Alliance Pty Ltd
white_paper@intersectalliance.com
+61 040 203 3347

Version Control:
1.0 1 March 2003 InterSect Alliance Pty Ltd

Table of Contents

1.0 Introduction

1.1 Intersect Alliance is a team of leading information technology security specialists, with extensive experience in both the policy and technical aspects of IT Security. The team undertakes national and international contracts to meet complex customer security requirements for organisations including key international finance and telecommunications companies, 'top-10' Australian corporations, and State and Federal Government agencies. The team and company are based in Canberra Australia.

1.2 Intersect Alliance developed the first security auditing and event logging software for the open source Linux operating system. The combination of tools, known as SNARE (System iNtrusion and Reporting Environment), provides information system owners, managers, security staff, and administrators with the ability to comprehensively monitor their information technology resources for incidents that do not meet the organisational security policy. The SNARE range of tools enhance an organisation's ability to detect suspicious activity by monitoring system and user actions, and provides an organisation with important evidence to use against potential and actual intruders. Besides the release of the Open Source tools, Intersect Alliance will also release a SNARE server code base, to undertake collection and analysis of events received from a range of operating systems and applications. Intersect Alliance will continue to release and improve on the SNARE tools, and will also continue to contribute to the International community through the release of the SNARE collection agents under the terms of the GNU General Public Licence (Reference A).

1.3 The purpose of this paper is to put forward ideas on comprehensive event logging collection and associated controls. In this paper "comprehensive" refers to the ability to collect events from a range of operating systems and applications, and ***not*** simply one source. It discusses the major impediments to the collection, collation, analysis and archiving of event logs, and what needs to be done to ensure event logging becomes a stable and relatively inexpensive way of monitoring security and activity in general within an organisation. As a result of the extensive experience of the Intersect Alliance team, the many and varied problems that have been experienced first hand are discussed in this white paper, along with features and discussions on the functionality required to overcome these problems. These ideas are put forward to assist users of Intersect Alliance tools to better understand the rationale behind some of the designs, and to hopefully contribute to the process of developing better open source tools for the community. It is also designed to stimulate discussion in the field of event log analysis.

1.4 The problems, resolutions and recommendations in this paper are designed to stimulate discussion in the infrastructure required to undertake event collection and analysis. It is hoped that new ideas will be forthcoming from the community, which will help to resolve some long standing issues in this field of IT security. We therefore welcome any feedback on the contents of this paper. Please send any and all comments to the Intersect Alliance team via the email address white_paper@intersectalliance.com.

2.0 Traditional Intrusion Detection Systems

2.1 Until recently, event collection and analysis was closely associated with what we refer to in this paper as “traditional intrusion detection systems”. These systems have evolved with the requirement for establishing an infrastructure which has the potential for handling events from many and varied sources. These events are required to be centrally managed or controlled, with the view of providing a set of analytical reports and management services to further control and manage the (potential) flood of event data. Prior to discussing event logging systems and requirements, how do traditional intrusion detection systems work?

2.2 Traditional intrusion detection systems work by capturing predefined “signatures”, to detect possible subversive activity. These signatures are based on known vulnerabilities such as those published by Common Vulnerabilities and Exposures (CVE – see Reference B). Signatures are based around these vulnerabilities that then alert the system administrator. SNORT is an example of a signature based system, and is in fact the only Open Source signature based detection tool (see Reference C). SNORT is a network intrusion detection system, that analysis traffic to determine whether it matches against a variety of known attack methods, which map directly to a formulated signature. The SNORT tool will use a database that may contain thousands of known vulnerabilities. As an example only and for demonstration purposes only, a rule (and there are many) in the SNORT database will search for possible “Netbus” trojan infections. “Netbus” is a trojan tool that allows a remote attacker to potentially take full control of the infected host. It involves a client (the attacker) and a server (the infected host), whereby the client can connect to the server usually via port 12345, though this is configurable. This vulnerability is detailed in the CVE dictionary as follows:

Row	Item	Details
1	CVE Name	CAN-1999-0660 (under review)
2	CVE Description	A hacker utility or Trojan Horse is installed on a system, e.g. NetBus, Back Orifice, Rootkit, etc.
3	Whitehats Reference	IDS401 TROJAN-ACTIVE-NETBUS-12345
4	Whitehat signature	alert TCP $INTERNAL 12345 -> $EXTERNAL any (msg: "IDS401/trojan_trojan-active-netbus-12345"; flags: A+; content: "NetBus"; classtype: system-success; reference: arachnids,401;)
5	SNORT SID	114
6	SNORT Signature	alert tcp $HOME_NET 12346 -> $EXTERNAL_NET any (msg:"BACKDOOR netbus active"; flags: A+; content: "NetBus"; reference:arachnids,401; sid:114; classtype:misc-activity; rev:3;)

Table 1: SNORT Signature for the “Netbus” Vulnerability

Table 1 shows the details for the Netbus signature. Row 1 shows the CVE Dictionary name, along with the description in Row 2. Rows 3 and 4 detail the relevant Whitehats signature, which is similar to the actual SNORT signature shown in Row 6. The actual signature states, in plain English, that an alert should be generated whenever a connection to an internal machine on port 12345 is detected from any external host. Note that the only discriminator for this signature is the requirement to match the internal port number as 12345. If any service other than Netbus is accepting requests on this report, then SNORT will interpret this as a Netbus attack and report an alert. So if a service is operating on this port, it will take an experienced network or security administrator the time to assess this situation and include a rule to reject alerts to the specific host that is running on the Netbus port. This leads to the next point.

2.3 Based on the above discussions, there is a clear limitation with the approach of using traditional intrusion detection systems. An administrator can be inundated in a sea of alerts if they do not configure the system carefully. Reference D is a report from the NetworkWorldFusion website, which details the analysis of live tests conducted on a production network to determine the usefulness of signature based intrusion detection products. This analysis was conducted on a production system to move away from the sometimes sterile and predictable lab/test network. It concluded that the large number of alerts adversely affected the usefulness of the products. Some quotes from this reference supporting this argument include: “We found most IDSs reported far too much rather than too little, making it difficult to pick out actual attacks from all the noise.” and “By far the biggest problem was a huge number of false positives, with sensors sending alarms for insignificant events - or even worse, for vulnerabilities that didn't exist.” also “Don't expect IDSs to be plug-and-play devices. To be effective, they require a lot of tuning, and a fair amount of security expertise.”. Reference E is another report on intrusion detection products from CIO Magazine. This reports states in part: “The problem with all intrusion detection systems is that they are not, and probably never will be, plug-and-play. Unlike firewalls, most intrusion detection systems require considerable technical smarts to set up and configure properly.” and “"Intrusion detection is extremely high maintenance," says Bruce Larson, a system vice president and director of special network operations for San Diego-based SAIC International (he designs and deploys network security architectures for SAIC clients, including several government agencies and utilities). He estimates that you need at least one full-time network engineer to monitor and tune an IDS or about $150,000 in fully loaded annual salary costs.”.

2.4 Tests conducted on a live network segment with approximately 20 workstations demonstrated an overwhelming number of alerts, with out any prefiltering. Over an approximate period of 18 hours, of which only 7 hours were during normal business hours, approximately 15.2 million alerts were generated, creating a log file over 2.1GB in size. This was generated using the SNORT Intrusion Detection system.

2.5 Clearly this is a well known problem with most intrusion detection systems, and understandably so, since they have been designed to detect possible intrusions based on a general, worldwide list of accepted and published signatures. The next section deals with Event Analysis as a form of security and administrative monitoring, and discusses how it differs from, and can add to, the signature based systems.

3.0 Event Logging to Detect System Activity

3.1 What is event logging? It is basically the collection, monitoring, analysis and archiving of log events generated by a system. The events may have been generated by an operating system, or by an application such as a database. Figure 1 below shows the event generated in the Windows Security Event Log, when the native Windows "Calculator" application is executed.

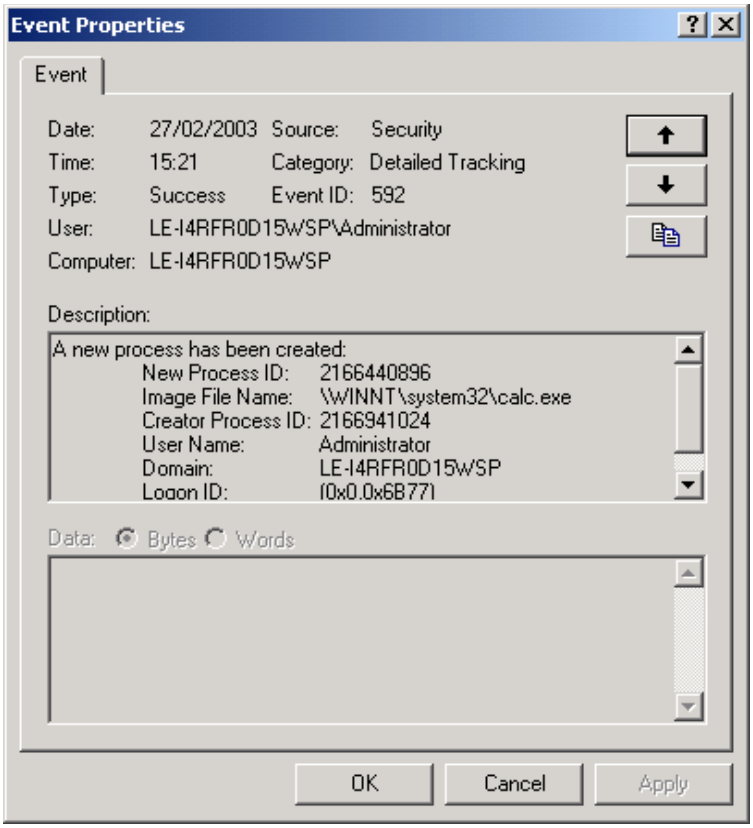

Figure 1: Windows Event Details

3.2 Unlike signature systems, rules need to be formulated to interpret an event as an item that should be alerted to administrators. For example, an event similar to that shown in Figure 1 but generated when (say) a database server exits (meaning it has ceased execution) will be of interest to database administrators, since it may indicate a failed service. It is highly unlikely that this sort of event would ever be included in a signature database for intrusion detection systems discussed previously, since this form of alert is very specific to an organisation and its security or monitoring objectives. As an another example, consider a situation where access to a database is based on a user's membership to a series of Windows domain global groups. These groups are called GLOBAL-DBASE-READ, GLOBAL-DBASE-WRITE, GLOBAL-DBASE-ADMIN. Membership of the first group will allow read access, the second write access, and the third will allow a user to execute some form of privileged access. A security monitoring goal for this organisation might then be to monitor the " GLOBAL-DBASE-ADMIN" group to determine which users have been added to or deleted from the Windows global group. Note that this is very different from simply reviewing the configuration of the global group, since reviwing the membership will only provide a "snapshot" of the configuration, and not the changes effected between the taking of the snapshot.

3.3 The role of user activity or event monitoring, especially inside corporate networks, is important in determining the activities of users. However, the way these event logs are managed will determine the success of failure of such an undertaking. It is clear from the above discussions that the key security objectives of an organisation must be known and agreed for a comprehensive event logging system to work. There may be some scope for generic "signatures" to be included in an event

logging system, but this will not provide the detailed analysis of specific events that is most likely required for most organisations that rely on IT systems. A generic "signature" may be something like: "Report on any user that has been granted 'Global Admin" group membership in a Windows domain". Whilst this broad security objective could be considered a signature, it only becomes a useful tool when the list of approved users is also known. In this way, some form of exception reporting can be undertaken, and thereby reducing the amount of "noise" of false-positive reporting that could quite easily plague a comprehensive logging system. In the database example provided in the above example, the data(base) owner must be involved in the process of specifying the authorised users of the system, for any meaningful analysis of the exception reports to be interpreted. If this is not undertaken, then the data owner or representative is faced with the potential of pouring through a list of (potentially) hundreds or thousands of users, in an effort to determining authorised user access.

3.4 The above discussions briefly highlight the need, and potential applications, of a comprehensive event logging system. The next chapters will discuss the "comprehensive" nature of event collection, ie; the collection from many and varied sources. The role of the next few chapters is to provide some insight into the key problems facing such a system, and some suggested solutions that may allow the community at large to tackle this problem in a manner which produces meaningful results.

4.0 Problems Associated with Effective Event Log Monitoring

4.1 Anyone that has been involved in collecting, analysing and archiving event logs, unless the scope has been small or targeted, will have noticed some key fundamental problems that plague this form of endeavour. The following sections detail the problems associated with developing an infrastructure to deal with the collection, analysis and archival of potentially millions of records per day, along with the task of analysing and reporting on such a collection of records.

Database Overload

4.2 In order to scope the problems associated with log event collection, the following statistics are provided as an example of the likely log sizes that could be expected. These statistics are based on actual values derived from real systems, although they are obviously not intended to cover all situations of expected log collection. The following volumes are provided for a single Windows 2000 workstation, with all Windows 2000 events were turned on (including some NTFS file auditing), no special client or server processing, apart from word processing, email and web clients, with some file browsing and the occasional use of the application "ssh". On an average day, approximately 20,000 Security events were collected, with only about 20 or so Application and System events. This translates to approximately 6MB of data per day (assuming 300 bytes per record). If a server generates (as a conservative estimate) a log file that is 10 times the size, then a 1000 workstation/50 server organisation can expect approximately (20M + 10M =) 30 Million events, which amounts to approximately 9GB of records every day, only from Windows NT/2000 Security events logs (primarily). Add to this another 1 Million events per day from one proxy server, three database servers and one mail server, and the event log collection process will now be collecting 35 Million events per day, or approximately 10.5GB of data per day.

4.3 In the example detailed above, the resources required to undertake this level of event logging need not be considered a burden (in terms of CPU/disk usage) on the host's operating system. For example, in the case of the workstation with 20,000 events being generated a day, then this represents 13 events per minute, which is well within the CPU resource limits. Although on a heavily loaded server or workstation this will add to the overall CPU usage and may make a noticeable difference to the server or workstation responsiveness, it is unlikely to be considered a problem that would preclude event logging in most environments.

4.4 Given the above statistics, it is clear that after a few months of operation, the database used to collect the events will grow to quite significant proportions. Based on the above statistics and after 3 months of continuous operation, the database size would grow to contain over 3 Billion events or close to 1TB (1000GB) of data. Whilst this in itself does not represent an insurmountable collection and storage problem, it should be expected that non-index key searching and analysis of a database this size may be too time and resource consuming. Unfortunately there is sometimes very limited capacity to "choke off" the number of records. For example, if using the Solaris operating system and its auditing sub-system (known as BSM – Basic Security Module), then file audit events cannot be effectively choked at the "front end" (ie; at the point of collection) which could amount to millions of events per day. If using Windows, and file auditing is enabled, then only certain directories/files can be choked, but not by user or filename.

Network Overload

4.5 Related to the above discussion, but of lesser importance, is the issue of network overload. The potentially large number of events can cause a significant loading of a local network, which may persist over time if the number of events being generated by (say) a cluster of heavily loaded servers is continuously high. Again, there is very little in the way at the disposal of the security administrator, in event of event records causing an increase in network loading, except for the Administrator to scale back the number of events being generated by the host system(s). Again, this

facility may not be available on some operating system and/or application event logging systems.

Collection from Application Servers

4.6 Although there is plenty of discussion about collecting events from the operating system, it is in some cases equally or more important to collect and analyse events from application servers. These servers include, but are not limited to proxy servers, firewalls, routers, database servers, web servers, email servers, document management systems, etc. In some cases, application events may be much more important in identifying trends or incidents. This means that the Windows OS collection agents may not be enough, and application collection agents would be required. This may be required for applications such as SQL Server, DB2 or Oracle databases, IIS or Apache Web Servers, Sendmail or Exchange email servers, Lotus Notes/Domino servers, etc. Without the use of collection agents for these servers, then the ability to automatically collect application event logs is greatly reduced.

Stability and Reliability of Collection Agents

4.7 The collection agents themselves must be stable and reliable, since they will be required to co-exist with the application or operating system for which they collect events. For example, a Windows agent must be able to work on a hardly-used workstation, or be able to operate on a highly loaded server. Either way, it is most critical that the agent interfere as little as possible with the host system, and undertake its tasks so that the overall resource loading is as minimal as possible. Any form of event filtering, for example, must place as small load as possible on the server. Special care must be taken to ensure that if encryption is added to protect events, then this does not add further load to a server/workstation. In the case of encryption, the question may be asked as to why events may need to be encrypted when organisation files are transmitted in the clear via standard Windows/SMB shares. There may, for example, be a requirement to checksum the events to ensure they have not been tampered.

Caching of Events

4.8 Some system managers require that every effort be made to ensure that all events are collected, and accounted. In order to achieve a goal of not losing events, then the collection architecture must be designed in such a way such that the collection agents must cache or store events when the central collection host is not available. Designing an ability to cache or store events will increase the client resource requirements, especially in terms of storage. If, for example, the network or central collection server(s) is not available, then events stored on the remote host will need to be stored until such time as the server(s) becomes available. In addition, once the central server(s) becomes available, then the collection agents could flood the network and collection server if the event forwarding process is not choked. Whilst on face value it is an easy statement to make that “all event delivery should be guaranteed”, administrators should be extremely careful on the implication. If a collection server is not available, then each and every agent will have to cache events. After a few days of downtime, this may begin to cause problem on the host systems, in terms of storage, which could become critical if the central server is not available for a week or more. Once the server or network is then available, care should be taken to ensure that the network is flooded while agents attempt to “catch up”.

Agent Remote Configuration Control

4.9 Once a target event log has been chosen as the source of events, and an agent has been configured to collect events, there is a problem of reconfiguring the remote agent, in event of a change in event collection requirements. If the scope of the network is extensive, or the reconfigurations must be undertaken frequently, then this may take an inordinate amount of effort. For example, changing the Windows NT audit configuration requires that an administrator log onto the server, and manually change the audit parameters as well as the event log collection sizes (if need be), and the policy on overwriting. This may be undertaken automatically if a product such as SMS was being used. In the case of Solaris, the method used to change the audit parameters can be quite involved for an

administrator, with the need to potentially change some BSM configuration files. If the security policy changes to the extent that the collection requirements change significantly for agents that collect event logs from operating system and application event logs (such as would be expected in a changed threat environment), then the administrative burden may preclude effective event collection.

4.10 In the event of a security compromise, the audit configuration may be deliberately altered which may mean no events are generated or collected. Alternatively, an administrator's error may also result in event logging being stopped or misconfigured. Without an effective remote control method, the ability to check the configuration and undertake a "heartbeat" (see whether the agent is still functioning) is not possible. Also, if the configuration needs to be changed rapidly and without undue attention (due to a suspected security compromise), then the most effective and efficient way is to issue new collection instructions to the agents. In implementing a remote control sub-system, care should be taken to strictly control access to the remote control functions. These functions should not be provided over public networks, unless some strict encryption or access control system is in place. Care should also be undertaken even on deployment within an organisation's internal networks.

Event Filtering

4.11 Usually, a security objective may be quite specific in its event collection requirements. For example, it may state something like "send an alert when the service called 'an_application.exe' is stopped". Unfortunately, Windows or Solaris operating systems will not allow the event logging system to be configured so as to report on only process tracking or executable events relating to a specific application. Instead, all process tracking or execution events on a specific host must be collected and later analysed to find any events that relate specifically to the event in question. There is therefore very little granularity in the collection process, which in turn leads to the inevitable problem of potentially being flooded with unrelated events.

4.12 In an ideal system, there will be some form of filtering, which would require a number of discriminators to work in unison. Discriminators may be different depending on the "objective". Extending the example above, an "objective" may require that "an alert be raised when the service called 'an_application.exe' is successfully stopped, by a user other the 'Administrator'". In this case there are essentially 4 filtering discriminators, namely;

a. A "process tracking" event must be trapped, and
b. It must relate to an executable called 'an_application.exe', and
c. It must have stopped, and not ***attempted*** to be stopped, and
d. It must have been undertaken by a user other than "Administrator'.

The agent must therefore work closely with filtering component, otherwise events will be lost, or wrong events will be collected. Failure to have an ability to filter events may lead to a situation where the risk of the central server(s) being flooded with unrelated events.

Costs

4.13 In some cases the ability to implement an effective event monitoring system is constrained by the cost, if using some commercial products. Costs associated with event collection may be based on the number of agents, and may vary between US$30 per agent to US$2000 per agent. The central server may vary between US$1000 and US$20,000. Some of these products include the ability to undertake signature matching, which would introduce a range of problems discussed in Section 2. Additionally, the agents may be limited in scope, and may not (for example) have the ability to collect from a range of operating systems and applications.

4.14 Using the above issues, we can now construct a series of functions that would ideally be included in

an event collection system. The next chapter details these functions.

5.0 Functionality for Effective Event Monitoring

5.1 This chapter details the functionality of an ideal event monitoring system. It details these functions based on "core" and "non-core" functions. This distinction has been made purely from the point of selecting those functions that are considered of higher priority.

Core Functions

5.2 **Filtering**. There must be an ability to filter on selected fields of a given event record. Even a simple filtering ability will allow the ability to greatly reduce the event log volumes to manageable levels. The down side in any filtering capability is that there will be some load on the host system, although the actual loading will depend on a number of factors including the complexity of the filtering, the number of filtering "objectives" and the number of events.

5.3 The filtering capability will need to be included within the agent collection system. In this way, the filtering can be undertaken before the events are sent out over the network and well before thy reach the central server collection node. It would also be ideal to include a capability whereby the filtering sub-system is bypassed, in situations where all events are required, and/or the CPU resource load caused by the filtering is too great for the host system.

5.4 **Stable Agents**. It is imperative that the collection agents minimise any interference with the host's operating system, and applications. It is therefore required that the agents themselves be as stable as possible, and with as little dependencies as possible. The only way of achieving a truly stable set of agents is to ensure a robust design, along with plenty of user testing. An example of a robust design, as will be discussed in the next section, is the tool SNARE for Solaris. This tool has been developed with a "wrapper" program that handles failures of the actual daemon, which will occur as a result of known bugs in the "praudit" application bundled in the Solaris operating system distribution. The "wrapper" program will detect failure of the main agent service, and restart the service with minimal (but inevitably some) loss of events. Although not ideal, this wrapper program provides a level of reliability that would simply not be achieved otherwise.

5.5 **Remote Control**. The ability to remote control the collection agents would be required when the audit/event logging configuration of the target system needs to be dynamically changed. The extent of the remote control functionality should include the ability to manage the filtering "objectives" of the remote agents, along with the ability for the remote agent to reset the host's event logging system, where appropriate. The remote control functionality should be able to control almost all facets of the agent's operation, short of de-installing the agent. The control over the agent's operation should not affect the host system such as rebooting to effect a change. In other words, the changes should be as seamless as possible.

5.6 **Central Collection Server**. The central collection server will most likely implement a myriad of functions, that include data management, reporting and archival tasks. Two of the most important functions that would be required in any implementation of a central audit server, is the abilities to synchronise the agents to the central collection "objectives", and to monitor the size of the storage databases. The first point relates to the fact that filtering "objectives" may become "out of sync" between the server and collection agents. This means that whilst the server is undertaking tasks to expect collection of certain events, the agents themselves are not passing the events to the central server, since they may filtering for other events. The second point relates to the size of the databases. As Reference D detailed, failure to monitor the databases may result in the overall collection system becoming unmanageable or even unstable if it consumes almost all storage resources. This point applies generally to any database system, but more critically to event log collection systems, that may be faced with a situation where they are attempting to collect millions or even billions of

records per day.

5.7 **No Agent Caching**. The agents should not implement any form of caching, for the reasons discussed in the previous Chapter. It may required to understand and scope how many events are being lost, if any, by the lack of agent caching. In this instance, the use of sequence number checking by the central server will provide the necessary statistics to determine how many events were lost, and to estimate the period (eg; time of day) when the losses may have occurred. Whilst this may not compensate for loss of events, it will allow for some scoping of the problem.

Non-Core Functions

5.8 **Encryption**. In some instances, it is required that the events themselves be protected either by encrypting the contents, providing an integrity check of the event record, or both. Care must be taken to ensure that this functionality does not place undue burden on the host's system resources, since a high event throughput will mean that a considerable amount of processing will need to be done to encrypt and/or sign event records.

5.9 **Easily tailorable to event log format**. Most event logs are simply “flat” text files, in which a system or application appends event records. In this case, the only discriminators required to read any type of event log would be the location of the log file, and the record structure. This could easily be coded so that these parameters are tailorable by the user, and hence able to be adapted to a wide range of event logs. Unfortunately, Windows event logs (for example) do not necessarily use this approach to record events, and specific Windows APIs are required to be used, to read from these event logs.

5.10 **Remote installation**. The remote installation would enable the collection agents to be installed into a wide range of workstations and servers, with minimal effort from administrative staff. In other words, the collection agents should lend themselves to having the ability to be either automatically or manually installed.

5.11 **Low cost**. The cost to implement all components, as with most endeavors, must be reasonable. Whilst this is obvious, the cost associated with deployment of commercial products could quickly escalate beyond reasonable limits, especially if workstation event logging is required to be in place.

6.0 SNARE Event Collection Tools

6.1 The SNARE tools have been developed to tackle the difficult issue of log collection and analysis. These tools have been developed, wherever possible, as Open Source tools and therefore free for anyone to use, and are available from the Intersect Alliance website. As can be seen from the preceding chapters, the task of effectively developing a comprehensive event logging capability can be full of traps, that may render an implementation inefficient at best, or ineffective at worst. The purpose of this chapter is to briefly detail the work undertaken by Intersect Alliance in this field.

Current Status

6.2 Intersect Alliance have, at the time of writing of this white paper, developed a series of Open Source agent tools to undertake the event collection tasks detailed in this white paper for Linux, Solaris and Windows operating systems. Thee tools have been titled SNARE for Linux, SNARE for Solaris and SNARE for Windows, and are all available for free, as Open Source, from the Intersect Alliance website, http://www.intersectalliance.com . In addition, an agent has been developed to collect Microsoft IIS web server logs, although this agent has no filtering or remote control capability. These basic agents are called "BackLog" agents, and are designated as such to indicate the lack of a full featured SNARE capability.

6.3 The two main purposes of releasing the SNARE units as Open Source has been to contribute to the community in the field of event log collection, and to enhance the reliability and stability of the SNARE agents. Intersect Alliance recognise that the success of a comprehensive event monitoring solution will hinge on the ability of the collection agents to report on the required events without inundating the central server. Releasing the Open Source SNARE agents means that there is a much higher likelihood that:

a. Users will make a serious attempt at collecting and analysing event logs,
b. The costs associated with such a venture will be kept to a minimum,
c. The reliability and stability of the agents will be enhanced as the user base grows, and
d. Functionality will be improved, again given a wider user base.

6.4 The BackLog tools will still be released, although the SNARE agent functionality will be used to replace BackLog agents, over time. The SNARE agent and BackLog tools differ in functionality in three critical areas, namely SNARE agents have:

a. Full remote control using the HTTP protocol (ie; a tiny, custom web server).
b. Full filtering capabilities, ie; to filter events based on the contents of record fields, and
c. Native Graphical User Interface (GIU) to set the configuration and view events.

Figure 2 below shows the filtering "objective" window for SNARE for Windows, which is used to ensure only certain events, as filtered by the set objectives, are passed to the central collection server.

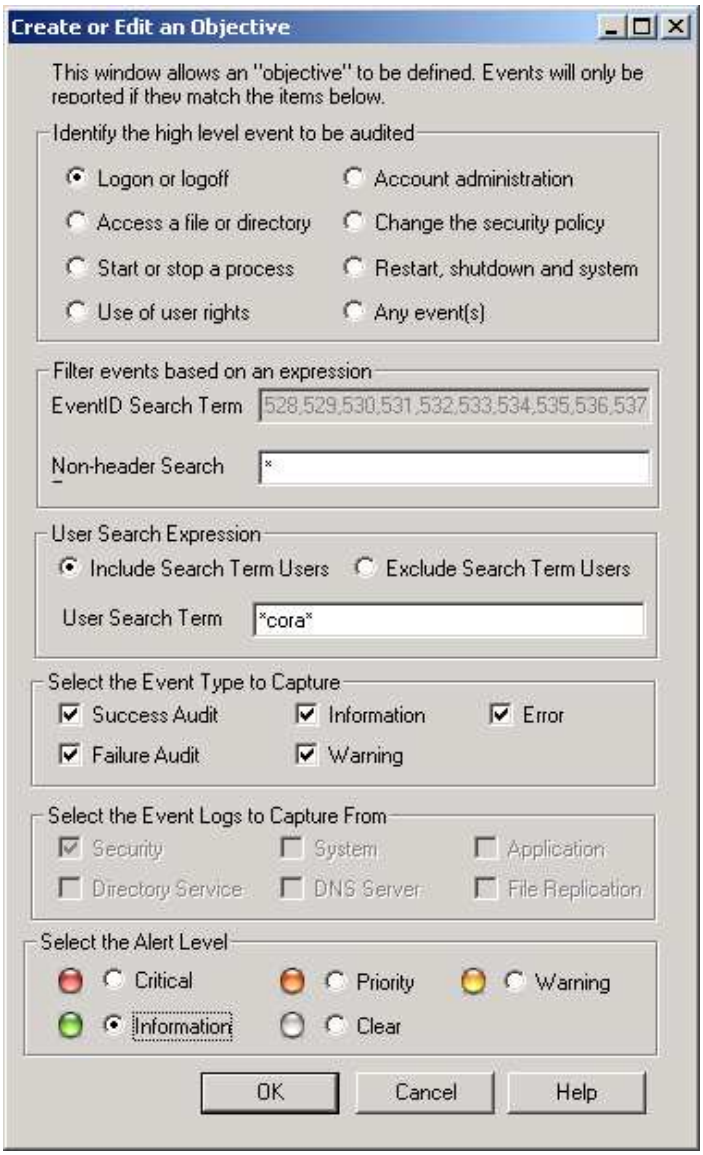

Figure 2: SNARE for Windows "Objective" features

Future Enhancements

6.5 There are a number of initiatives underway which will enhance the functionality and reliability of the SNARE agents. Firstly, a central server used to collect, analyse, report and archive events from a wide variety of users will be released in some form as an Intersect Alliance product. At this stage, this release will most likely be a product released as a "code base", that mirrors a commercial product, but without the closed/proprietary source implications. This will allow users of the central server to either use the server "as is", or change it through an agreed development process to meet their extended requirements. This server will be released along with a Linux distribution, in the form of an ISO image, to facilitate a quick installation. The server itself has been designed around the PHP language, using the Apache web server and MySQL database server. Full details of the implementation will be available in the months following the publication of this white paper.

6.6 Work is underway to include the SNARE for Linux module functionality directly into the production Linux kernel. This is being coordinated with the Linux kernel developers, and may take some time to integrate. Until then, the functionality available in SNARE for Linux kernel module will still be available to Linux users.

6.7 It is planned to release more SNARE agents, with some of the more common agents designed to

collect event logs from Apache web server, IIS web server, Lotus Notes application and Oracle database server logs. These will be released in due course over the next few months, and all will be released as Open Source.

Annex A - References

The following references have been used in the drafting of this white paper:

A. The GNU General Public Licence. http://www.gnu.org/licenses/gpl-faq.html.

B. Common Vulnerabilties and Exposures. This is a dictionary of names, which reflect known and candidate vulnerabilties and exposures. See http://www.cve.mitre.org/

C. SNORT. This is an Open Source network intrusion detection engine. www.snort.org

D. NetworkWorldFusion. Crying Wolf: False alarms hide attacks. “Eight IDSs fail to impress during the month long test on a production network.” Dated 24 June 2002. http://www.nwfusion.com/techinsider/2002/0624security1.html

E. CIO. “Intrusion Detection”. Dated 15 September 2002. http://www.cio.com/research/current/intrusion/. Originally published as: http://www.cio.com/archive/091502/et_article.html

F. “Intrusion Detection”. Terry Escamilla. Wiley Computer Publishing. ISBN 0-471-29000-9.

G. SANS Institute Intrusion Detection FAQ. http://www.sans.org/resources/idfaq/#top