Snare Toolset White Paper

Table of Contents

1. About this Guide 3
2. Executive Summary 4
3. Introduction to IA 5
4. Traditional Intrusion Detection 6
5. The Basics of Event Log Monitoring 8
6. Key Issues in Event Log Monitoring 10
7. Effective Event Monitoring 14
8. The Snare Event Logging Tools 16
9. Snare Server Capabilities 19
10. About Intersect Alliance 27
11. Appendix A - References 28

1. About this Guide

This guide is a white paper for the Snare tools. It describes some of the design philosophies and constraints that have shaped the development of the Snare tools. It also describes the design philosophies for the Snare Server.

Other guides that may be useful to read includes:

- Snare Server User Guide
- Snare Agent Documentation

2. 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 and report on system activities promises to enable a whole new range of security related services. It also helps to meet certain regulatory requirements facing specific organizations that need to monitor activity on their systems.

The effectiveness and efficiency of event log collection can be undermined by:

- costly software,
- unstable event collection agents,
- an inundation of events from a variety of sources,
- adverse resource utilization on host systems,
- lack of a full remote control functionality or
- lack of flexibility in the reporting and alerting systems.

This paper explores these problems, and proposes a set of functions required to minimize the risk, and maximize 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 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. It also details the features of the commercial Snare Server in meeting most of the needs of the ideal collection systems, in a cost effective manner.

3. Introduction to IA

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 provides event monitoring solutions to meet complex customer security requirements for organizations including key international finance and telecommunications companies, and State and Federal Government agencies.

Intersect Alliance developed the first security auditing and event logging software for the open source Linux operating system, and has continued to develop and deploy tools to facilitate effective audit analysis on a wide range of platforms and applications, including Windows, Solaris, AIX, Irix, Tru64, IIS Servers, Apache, ISA Servers, Squid, Lotus Notes, and many others. 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 may indicate a compromise of their organizational security policy. The Snare range of tools enhance an organization's ability to detect suspicious activity by monitoring system and user actions, and provide important evidence to use against potential and actual intruders. 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 License (Reference A).

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 related activity within an organization. As a result of the extensive experience of the Intersect Alliance team, some of the many and varied problems that have been experienced first hand are discussed in this 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 design features, 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.

The problems, resolutions and recommendations in this paper are designed to stimulate discussion on 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 component 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 our online contact page: https://www.intersectalliance.com/contact-us/.

4. Traditional Intrusion Detection

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 capacity for handling events from many homogeneous 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?

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. Signatures based around these vulnerabilities are used by security tools to alert the system administrator when a particular pattern is seen on the network, or within log entries. SNORT is an example of a signature based system. Snort is an Open Source signature based network intrusion detection tool, that analyzes traffic to determine whether it matches against a variety of known attack methods, as described by the installed signatures. The SNORT tool will use a database that may contain thousands of known vulnerabilities. As an example for demonstration purposes only, a rule (and there are many) in the SNORT database will search for possible 'Netbus' trojan infections. 'Netbus' is a old trojan tool that allows a remote attacker to potentially take full control of the infected Windows-based 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, eg. 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 Whitehat's 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 port, then SNORT will interpret this as a Netbus attack and report an alert. So if a valid 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.

Based on the above discussions, there is a clear limitation with the approach of using traditional intrusion detection systems. An administrator can be inundated by a flood of alerts, if they do not configure the system carefully.

Reference D is a report from the Network World Fusion 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. Reference F, a SANS resource, also details the problems associated with signature based systems.

Tests conducted on a live network segment with approximately 20 workstations demonstrated an overwhelming number of alerts, without any pre-filtering. 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.

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.

5. The Basics of Event Log Monitoring

What is event logging? It is basically the collection, monitoring, analysis and archiving of log events generated by an operating system, appliance, application, data base or other specific systems. In most cases, these events will be generated by the operating system, but there has been a significant increase in application-generated event logs recently. Figure 1 below shows the native Windows event generated in the Windows Security Event Log, when the Windows 'Calculator' application is executed.

Note that for this event to be generated, the operating system must be configured to report on these types of events, and the systems administrator must filter on the specific event, or search through a long list of events to find a specific occurrence.

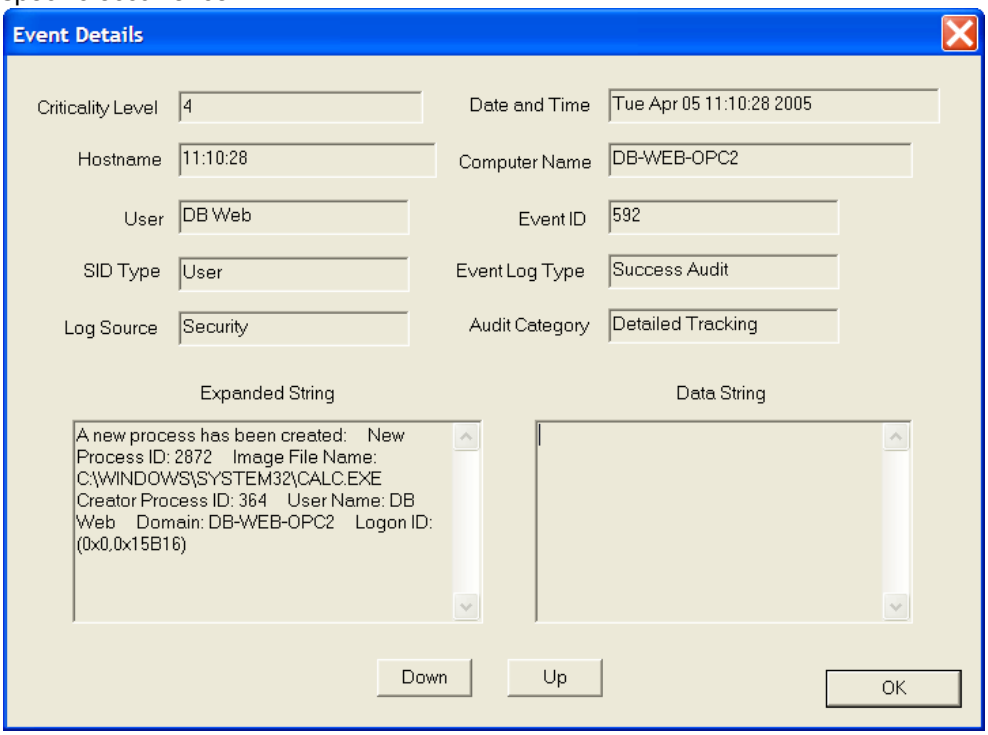

Figure 1: Windows Event Details

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 organization and its security or monitoring objectives. As 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 organization 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 reviewing the membership will only provide a 'snapshot' of the configuration, and not the changes effected between the snapshots.

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 or failure of such an undertaking. A crucial success factor of event logging systems, as implied from the above discussions, is that the key security objectives of an organization must be determined for a comprehensive event logging system to work effectively. 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 organizations that rely on IT systems. A generic 'signature' may be something like: "Report on any user that has been granted access to the

'Domain Admin' group, 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 – and perhaps excluded from the list. In this way, some form of exception reporting can be undertaken, and thereby reduce the amount of 'noise' of false-positive reporting that could quite easily plague a comprehensive logging system. In the database example provided above, the data(base) owner must be involved in the process of specifying the authorized 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 determine authorized user access.

The above discussions briefly highlight the need and potential applications of a comprehensive event logging system. The next chapters will discuss, in part, 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.

6. Key Issues in Event Log Monitoring

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

6.1. Data Volumes

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, in which 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 organization 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. 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 potentially collecting 35 million events per day, or approximately 10.5GB of data per day.

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 being generated per minute, which is well below any reasonable CPU resource usage. 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. It is not uncommon to see upwards of 300,000 events per 15 minute period for Snare Server production systems, for relatively small numbers of servers.

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 searching and analyzing a database this size could be time and resource intensive. 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 configured as auditable, but not by user wildcard or by file wildcard, unless a Snare tool is used to filter unwanted events at the source. More on this later.

6.2. Network Load

Somewhat related to the above discussion, but of lesser importance is the issue of network overload. The potentially large number of events may 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 are very few options available to the security administrator, if event records cause an increase in network loading, other than 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.

In the case of (say), 500,000 continuous events being received every 15 minutes, this would represent a considerable collection profile of about 50 million events per day. In terms of volumes, this represents about 165KB

of data per second, assuming 300 bytes per event. On a 100Mbps network, this is about 1.5% bandwidth usage at the collection (Snare Server) end, and about 0.15% bandwidth for a 1Gbps network. This bandwidth usage falls off to negligible levels at the point of event generation (i.e. at the host end).

6.3. Collection from Application Servers

Although there is plenty of discussion on collecting events from the operating system, it is in some cases equally or more important to collect and analyze events from application servers. These servers include, but are not limited to proxy servers, firewalls, routers, database servers, web servers, email servers, authentication 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 for systems such as SQL Server, DB2 or Oracle databases, IIS or Apache Web Servers, Sendmail or Exchange email servers, Lotus Notes/Domino servers, etc could significantly add to the overall security picture of an organization. Without the use of collection agents for these servers, then the ability to automatically collect application event logs is somewhat reduced.

6.4. Stability and Reliability of Collection Agents

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 lightly-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 a 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 significant extra load to a server/workstation.

Encryption may introduce significant overhead to the event logging agent. The use of encryption should be carefully considered, especially if no encryption is currently used on internal network traffic.

The question may be asked as to why events may need to be encrypted when organization 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 with, or simple sequence number checking may suffice. If however, encryption is required by the agency security policy or a Government standard, then it should be enabled.

6.5. Caching of events

Some system managers require that every effort be made to ensure that all events are collected, and accounted for. In order to achieve a goal of not losing events, then the collection architecture must be designed in such a way 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 for extended periods of time, 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. After a few days of downtime, this may begin to cause problems 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 not flooded while agents attempt to 'catch up'.

6.6. Agent remote configuration control

Once a target event log has been chosen as the source of events, and an agent has been configured to collect events, there may be a need to reconfigure 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 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 Linux or Solaris (say), the method used to change the audit parameters can be quite involved for an administrator, with the need to potentially change some BSM/audit crucial configuration files, and potentially even reboot the system. 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 or timely event collection.

In the event of a security compromise, the audit configuration may be deliberately altered which may result in no events 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' (ie. to see whether the agent is still functioning as required) is not possible. Also, if the configuration needs to be changed rapidly and without undue attention (due to, say, an ongoing, 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 organization's internal networks.

6.7. Event filtering

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 analyzed 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.

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 above example, 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:

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

The agent must therefore work closely with a 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 there is the risk of the central server(s) being flooded with unrelated events.

6.8. Costs

In some cases the ability to implement an effective event monitoring system is constrained by the cost, if using commercial products. Costs associated with event collection may be based on the number of agents, and may vary between US$50 per agent to US$2000 per agent. The central server may vary between US$3000 and US$200,000. Some of these products include the ability to undertake signature matching, which would introduce a range of problems discussed previously. 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.

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.

7. Effective Event Monitoring

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.

7.1. Core Functions

7.1.1. Filtering

There must be an ability to filter on selected fields of a given event record. Even a simple filtering implementation could 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.

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 they 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. Filtering should be undertaken by inclusion or exclusion.

7.1.2. Stable agents

It is imperative that the collection agents minimize any interference with the host's operating system, and applications. It is therefore required that the agents themselves be as stable as possible, 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 an internal monitoring capability, that handles failures of the internal binary-to-text event translation tool caused by known bugs in the 'praudit' application bundled in the Solaris operating system distribution. The 'monitor' program will detect failure of the praudit tool, and restart the service with minimal (but inevitably some) potential loss of events. Although not ideal, this internal monitoring facility provides a level of reliability that would simply not be achieved otherwise.

7.1.3. Remote Control

The ability to remotely 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 uninstalling 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.

7.1.4. 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 ability to provide detailed yet informative reports, and to monitor the size of the event datastore. The first point relates to the fact that reports may be too cumbersome and detailed for some users (ie; provide too many false positives), or may not provide enough detail. The second point relates to the size of the databases. As Reference D detailed, failure to monitor the data volume 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 event records per day.

7.1.5. Agent Caching

The agents should be able implement a form of caching, for the reasons discussed in the previous Chapter. It may be required to analyse 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. In any case, the ability to send events using an unreliable protocol (such as UDP), or a reliable protocol (such as TCP), as well as for the agent to store events when the central server is offline, should be available to the security administrator.

7.1.6. Event Redundancy Issues

The agents should be able to send events to 2 or more locations, in real time. This feature would aid in the establishment of a fully redundant installation, where 2 independent collection systems are established and operated. In most situations, a single collection system will be required. However, for those systems where full redundancy is required, then the most effective way is to ensure full redundancy from the "collection end".

7.2. Non-core functions

7.2.1. 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. Any encryption undertaken by the agents should be in accordance with approved encryption standards, and not "proprietary algorithms".

7.2.2. 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, operating system event logs (in general) do not necessarily use this approach to record events, and specific OS-specific system calls are required to read from these event logs.

7.2.3. Remote installation

The remote installation would enable the collection agents to be installed on 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.

7.2.4. 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. The key aspect of this item is the scalability. In almost all situations, users implement only a subset of the required logging infrastructure, usually to test the product and proposed implementation. At other times, users implement logging only on selected servers. Once the initial implementation is successful, the costs required for full implementation across all devices, servers and workstations can prove too onerous. This is an unfortunate side effect of the costs associated with product vendors that charge on a per "agent" or "node" basis.

8. The Snare Event Logging Tools

The Snare tools have been developed to tackle the difficult issue of log collection and analysis. These tools have been developed and deployed, wherever possible, under Open Source licenses, and are therefore free for anyone to use. They are available from the Intersect Alliance or Sourceforge websites. As per 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.

8.1. Current status

Intersect Alliance have, at the time of writing this white paper, developed a series of Open Source Snare agent tools to undertake the event log collection for the following operating systems; Linux, IRIX, AIX, Solaris, Tru64 and Windows (all versions). In addition, Intersect Alliance have developed the "Epilog Agent" for UNIX and Windows, that allows for generic log collection from specific applications in either Windows or UNIX platforms. These specific event logs, for which the Epilog agents are available, include applications such as Apache, SQUID, IIS and ISA web/proxy server event logs. The Snare tools are available from the Intersect Alliance website, http://www.intersectalliance.com

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 recognize 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:

1. Users will make a serious attempt at collecting and analyzing event logs,
2. The costs associated with event logging will become more reasonable,
3. The reliability and stability of the agents will be enhanced as the user base grows,and
4. Functionality for the central log collection server will be improved.

Figure 2 below shows the 'filtering objectives' 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. This figure is a snapshot of the remote control page, which can be used to access the Snare agent's configuration, using any common web browser. If remote control is considered too risky for a given network, it may be further protected, or disabled altogether.

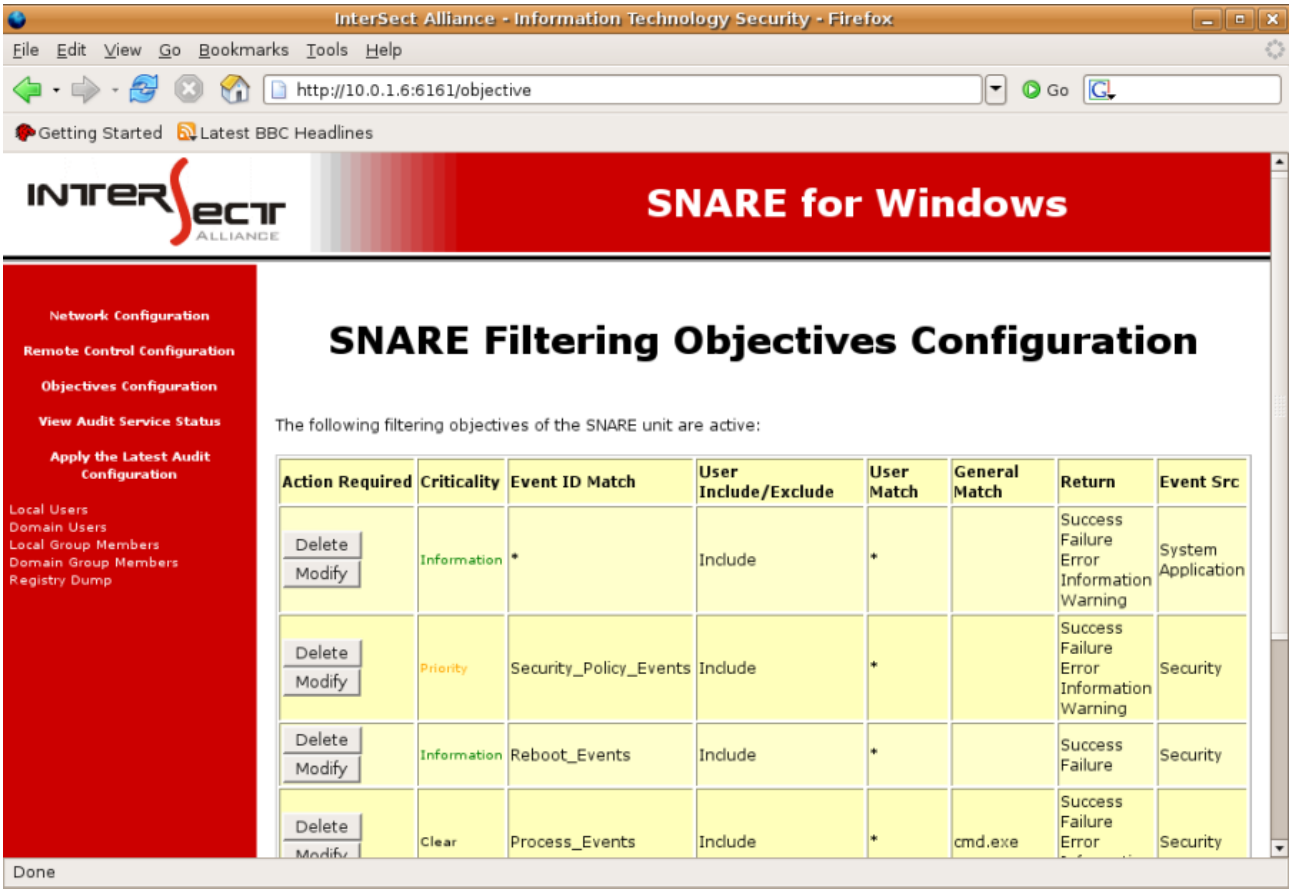

Figure 2: Snare for Windows 'Filtering Objective' Remote Control Page

8.2. Advantages of the Snare Open Source agents

The Snare tools have a number of advantages which allow them to be deployed in almost any situation. These are discussed below:

8.2.1. Latency and real time features

The Snare agents operate in real time mode. This means that as the events are generated, they are automatically copied and sent to the Snare Server with no delay. There is therefore no latency between event generation and reception by the central Snare Server collection node, except for delays introduced by the host operating system or network components. The clear benefit in having no latency is that it becomes more difficult to compromise a system, without being detected. Deleting the local log files will not remove the events which have already been sent to the remote Snare Server.

8.2.2. Remote control

The Snare agents are able to be fully remote controlled, via a standard web browser. This means that all aspects of their operation can be changed, without having an Administrator making the changes. If required, the Snare agents are also able to change the operating system's native audit settings to match the audit collection requirements, without the need for an administrator.

8.2.3. Objective Filtering

Most operating system logging sub-systems can generate a flood of events, under some configurations. It is therefore important that the agents are able to filter those events which do contribute to the agency's security requirements, or to trap only those that are required whilst ignoring other events. In some situations, the majority of generated events will be required to be collected by the collection Snare Server. Tailoring the required events, whilst filtering or discarding the unwanted events, can be undertaken by the Snare agents. The Snare agents include the ability to filter events by inclusion or exclusion, using standard or complex expressions to filter on content, event type, user, and/or success/failure of the event record. Multiple objective filtering expressions may exist at any one time.

8.2.4. Native OS audit control

The event generation or event sub-system on most modern operating systems includes the ability to control how the event logs are generated, configured and produced. On some systems this can be quite complicated and confusing. Fortunately, the Snare agents are able to configure the native event sub-system, and if so desired allow only specific events to be generated which are required or defined by the security policy. Also, the Snare agent can be configured so that it does not, in any way, reconfigure the underlying operating system. In these cases, the systems administrator should be careful to enable the event sub-system to collect the events of interest.

8.2.5. Redundancy

The best redundancy measures in a logging architecture, is to duplicate the events at the generating "end" or the source of the events. Some Administrators focus on providing dual hardware/disks, or undertaking regular backups. However, the best redundancy is provided using full mirrored hardware and software devices. This means that the agents must send copies of all events, in real time, to one or more separate locations. This function is built into the Snare agents, and therefore allows for full redundancy in those situations where a continuous logging operation is required.

8.2.6. TCP and event caching

The Snare agents send events to the central collection server using the UDP protocol. However, they can also send events using the TCP protocol. Whilst the TCP protocol adds some overhead to each sent packet, it does provide a way of reliably delivering events to the central logging server. In "TCP Mode", the Snare agent will also cache or store events locally until the Snare Server is available. In "UDP Mode", this will not be undertaken.

8.2.7. Encryption

Some of the Snare agents are able to encrypt log records, if so desired by the security administrator. Work is currently being undertaken to enable user selectable encryption for all Snare agents.

9. Snare Server Capabilities

The Snare Server is a proprietary product designed to act as the central point of collection for all the Snare agents. The Snare Server has been designed to be able to collect from as many agents as possible, without the need to 'cap' the number of collection agents. Based on the collection issues discussed in the previous chapters, the features of the Snare Server are discussed in the following paragraphs.

9.1. Data management

If left unchecked, collection of event logs into the Snare Server will continue to grow unabated. In most commercial installations of the Snare Server, the log collection rate varies between 1 and 20 million records per week, averaging about 10 million per week (from an average of about 20 servers). Whilst these figures will vary greatly depending upon the agent configuration, an average rate of collection will still mean approximately 500 million records after 6 months of collection. A database of 500 million records, even on the most powerful servers, will become unresponsive to search queries.

The Snare Server tackles this problem in two ways. The first is by allowing an administrator to migrate analysis to a 'scheduled task', whereby the large data store is queried overnight, with results available in the morning, and by ensuring that event records can be automatically migrated out of the Snare data store, to optical media on a regular basis, in a compressed text format log. Compressed, archived data can be held for years and recalled into the Snare Server database if required.

The automated migration strategy will ensure that events do not 'clog up' the datastore. Since there will be requirements, from time to time, for the security administrator to search for events that have been previously archived, it is important that the Snare Server be able to call in old files for analysis and review.

9.2. Infrastructure design features

The Snare Server operates using a Linux base (hardened version of Debian/Ubuntu), PHP scripts to provide a clean user interface and reporting capability, and the Apache web server to allow users to gain access to the Snare Server via any web browser. The Snare Server is able to collect event logs either via UDP or TCP port 6161 or 514 (SYSLOG) in real time, or via batch transfer for selected file types. The batch transfer allows for those situations where it is not possible to send the events real time, or a greater degree of event log integrity is required. In the case of batch transfers, this may be undertaken using the 'scp' or 'ftp' protocol, which (in the case of 'scp') allows for a fully authenticated and encrypted tunnel to be established between the remote system and the Snare Server.

The Snare Server also includes a capability called the 'Snare Reflector', which facilitates 'event mirroring' to another Snare server. The secondary Snare server can be used either as a 'warm spare', which takes over the role of the primary server if a network, or hardware-related problem is encountered, or alternatively, as the 'master server' at the head of potentially multiple 'feeder' Snare Server installations, amalgamating the data from several self-contained organizational sub-units.

The Snare Server collects events and logs from a variety of operating systems, applications and appliances including, but not limited to: Windows NT/2000/XP/2003, Solaris, AIX, Irix, Linux, Tru64, ACF2, RACF, CISCO Routers, CISCO PIX Firewall, CyberGuard Firewall, Checkpoint Firewall1, Gauntlet Firewall, Netgear Firewall, IPTables Firewall, Microsoft ISA Server, Microsoft IIS Server, Lotus Notes, Microsoft Proxy Server, Apache, Squid, Snort Network Intrusion Detection Sensors, IBM SOCKS Server, and Generic Syslog data of any variety.

The central datastore is divided into groups of files, with metadata such as date, time, system, and log type, used to segregate log data for ease of access.
Since the Snare Server user interface operates using mainly high level programming languages, the development, testing and distribution for new features can be measured in days rather than weeks.

The clear problem with developing a complex event log analysis and reporting system using a low level language such as C or C++, is that the design effort required to develop secure and reliable code can be in the orders of magnitude greater than for applications developed in lower level languages. The Snare Server avoids this problem by not only coding in PHP, but also by providing a feature-rich core, which allows Snare Server 'objectives' to be

written very simply. These objectives perform queries and reports against the Snare Server datastore. This is discussed further below.

9.3. User access

All users access the Snare Server either via standard port 80 (http) or via port 443 (https – Secure Web). Any common web browser may be used to access the Snare Server user interface. This includes browsers such as Mozilla Firefox, Google Chrom or Internet Explorer.

The Snare Server is organized into 'objectives'. These objectives represent specific and/or common queries and reports. An example of an objective is 'Windows Users Created and Deleted', as shown in Figure 3 below. This objective will display all the users created or deleted on specified systems, within the specified time period (ie. say the last week), by specific staff (or all staff), and produce a report at specified intervals.

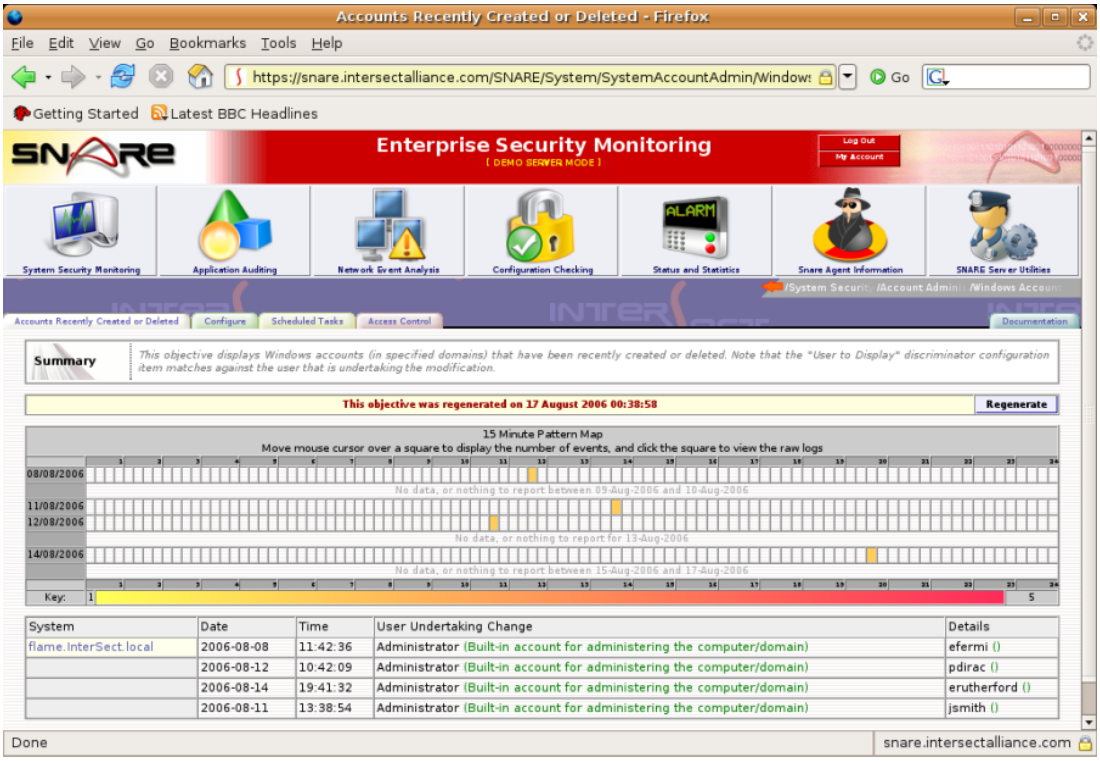

Figure 3: Users Created or Deleted Objective

Figure 3 shows the main report page associated with the specific objective. Above the objectives, there is the 'Configure', 'Scheduled Tasks', and 'Access Control' tabs. The 'Configure' tab can be used to set a number of discriminators for the objective, such as which system(s) to report, which users should be reported, etc. The type and quantity of available discriminators will depend on the type of objective, and in particular, the event log type on which the report is based. The 'Scheduled Tasks' tab is used to enable the report regeneration frequency (eg. daily, weekly, etc.) and which staff should receive emails on report production.

The details for each and every objective is stored within the Snare Server database. This information acts as a metadata repository on the objectives which should be viewed at a particular site, and the specifics about report generation and access. Similar in action to a data warehouse, these tables become the metadata repository on the actions to be taken at a specific site.

The 'Access Control' tab is used to ensure only authorized users have either READ and perhaps WRITE access to the objective. This function allows specific access to specific objectives by authorized users. Access Control is organized via groups, so as to ease the administrative burden for users that require similar access profiles. Access by users can be arranged either via local files, or authentication derived either from a Microsoft Active Directory or

LDAP database. User access to the Snare Server web interface is fully audited, so that the user name, date, time and objective(s) that were accessed are all recorded in the Snare Server database.

Full details on user access is contained in the Snare Server document titled 'Snare Server User Guide'.

9.4. Snare agent support

The Snare Server has been designed by the same staff that designed the Open Source Snare Agents. This means that the features designed into the Snare agents are also available in the Snare Server, and the two entities work seamlessly.
The key points of the Snare agents are that they are able to send a plain text version of the event logs, in real time, for only those events that are specifically required (and all the rest are filtered), to a nominated host, whilst affecting the host system as little as possible. The ability to be remote controlled is used by the Snare Server to monitor the configuration of all remote controllable agents, and to report on them as a 'group' which share the same configuration.
Given that the agents are independent entities that MAY be controlled by the Snare Server, it is not desirable to allow the Snare Server to automatically control the version of the Snare agents. However, it may be appropriate to control the configuration in some instances. In any case, new or updated versions of the Snare agents are able to be installed by relatively inexperienced staff, or via experienced systems administrators using common application distribution tools such as MSI, SMS or custom designed scripts.
The Snare Server has been designed to collect either from a handful of agents, or from thousands of agents if so desired. In essence, the Snare Server has been designed to cater for as many agents as possible. This is the reason why the full Snare Server version is not licensed for a limited number of agents, and will not prevent the user from collecting from thousands of agents if need be. However, the Snare Server administrator should pay particular attention to not overloading the Snare Server with too many events. If need be, the agents should be tailored to collect only those events which are reasonable in terms of collection volumes, security and forensic requirements and reporting.

9.5. Event data security issues

Under the normal course of events, events from specific agents are sent using the UDP protocol. This provides for real time transmission of events, without the overhead of having to establish a bi-directional circuit (such as TCP), and the attendant problems of caching events. Snare agents are able to handle transmission modes in UDP or TCP. In the case of selected logs such as Mainframe (MVS) logs, it may be desirable to be able to copy the logs to the Snare Server in batch mode. In these instances, the ftp or scp protocol may be used to batch copy the logs to the Snare Server. The disadvantage with this configuration is that, so long as the logs reside on the host system, they are open to compromise from unauthorized users or activity. This is why the real time transfer of event logs is preferred, and is recommended to be implemented on all systems.
The UDP transfer method allows for the most secure of data transfers from what is commonly known as a one way gateway or a 'data diode' (see Reference I). The data diode allows for collection of information from untrusted or lower classified networks, without compromising the higher classified network. It does this by only allowing for an electrical connection that allows for data to be passed 'up' but not 'down', or alternatively, it allows for data to be passed in one direction, much like a half duplex link but only able to communicate in one direction. Using the UDP protocol, event records can then be sent to another network since the protocol does not require a reverse 'control' channel or circuit establishment in the same manner as the TCP protocol.
It is common practice for most agencies to employ a number of firewalls in a 'DMZ' arrangement to effectively create a compartmentalized area between the Internet and the internal network. The firewall policies in place usually do not allow, or severely restrict, logs to be sent directly from the DMZ to the internal network, since this involves the creation of a firewall rule which would allow connections from the DMZ to the internal network, and hence may lead to a compromise of the internal network. In most situations therefore, it is not possible to send logs that have originated in the DMZ, to the internal network in real time. A less risky arrangement is to allow an internal network process to make a connection from the internal network to the DMZ and 'pull' the logs back into the internal network. As discussed previously, the 'scp' capability may be used to undertake batch transfer of selected log types (such as Mainframe MVS logs, and Firewall 1 logs) where there is no agent available to transfer the logs, or where it is the preferred method of transfer. Clearly, using 'scp' will mean that the logs are transferred securely via an encrypted tunnel to the Snare Server. Since the 'scp' transfer may be originated via either end, this form of protocol can be considered to be firewall friendly, in the sense that the connection request could be originated from either 'end' of the connection.

9.6. Analysis and reporting

The Snare Server comes complete with the ability to provide reports immediately after installation. It does this by provided 'objectives' which are common to almost all agencies. The 'objectives' allow the security administrator to begin reporting without having to understand the data structure of the Snare Server database tables. These objectives also represent years of experience accumulated by the Snare Server developers in understanding and interpreting the needs of the Snare user community.

In addition to the 'canned' objectives, there are many 'clonable objectives', which as the name suggests, allows for the creation of any number of objectives that suit specific local needs. For example, almost all of the firewall related objectives allow for 'clonable objectives'. This means that the user may create an objective with as many discriminators as is required, and scheduling and access control to suit local conditions.

In addition to 'clonable objectives', there are also 'dynamic clonable objectives'. These objectives allow for full queries against any of the Snare Server database tables, without any restrictions. They not only allow for permanent storage of these queries (along with scheduled and access control), but also the ability to dump the contents of the objectives to a text or spreadsheet (eg. MS Excel) format. An example of dynamic query is shown in Figure 4 below.

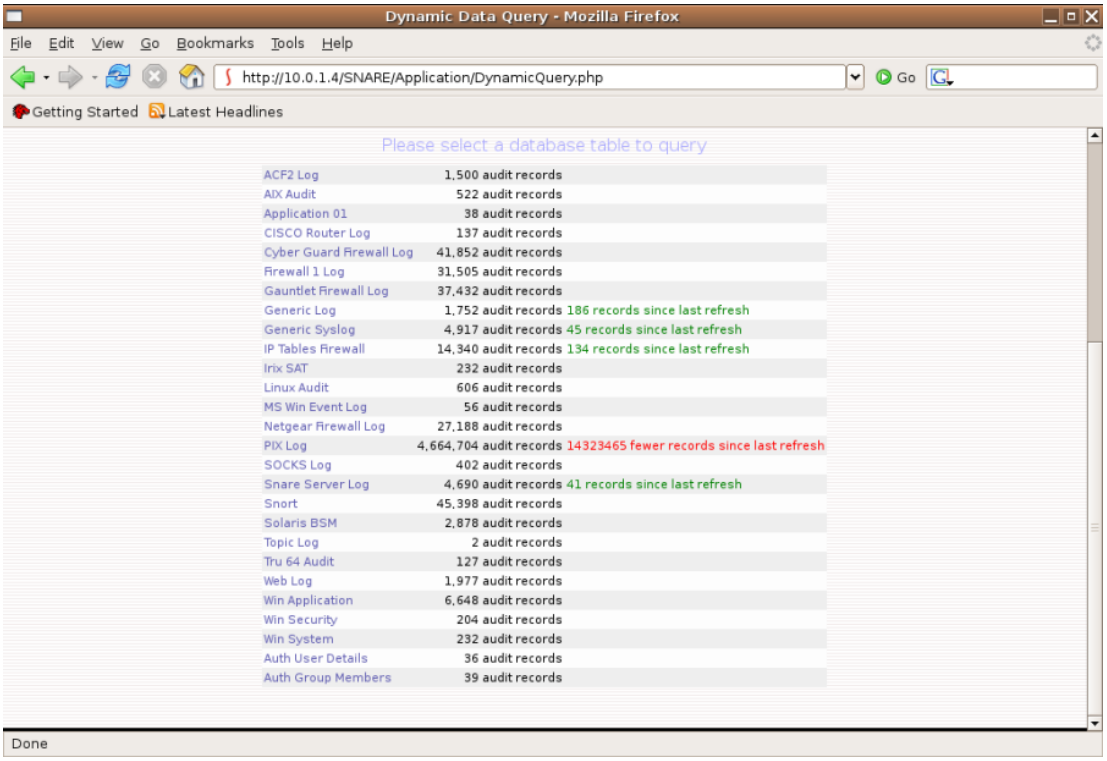

Figure 4: Dynamic Query Objective

The Snare Server offers a number of different types of report formats to aid in the interpretation of complex or voluminous amounts of data. There are many types of reports, but three of the most common report formats are presented in this section, for the sake of brevity. The first type of report is the '***time plot***'. This report type is shown in Figure 5 below, and depicts the *number* of events in a 15 minute periods for a whole day or for a pre-defined number of days. In the case of Figure 5, the reporting period is 35 days. Placing the mouse cursor over any of the active squares shows a number which details the number of events for the given 15 minute period. 'Clicking' the square will take the user to the raw logs that are used to calculate the number of events.

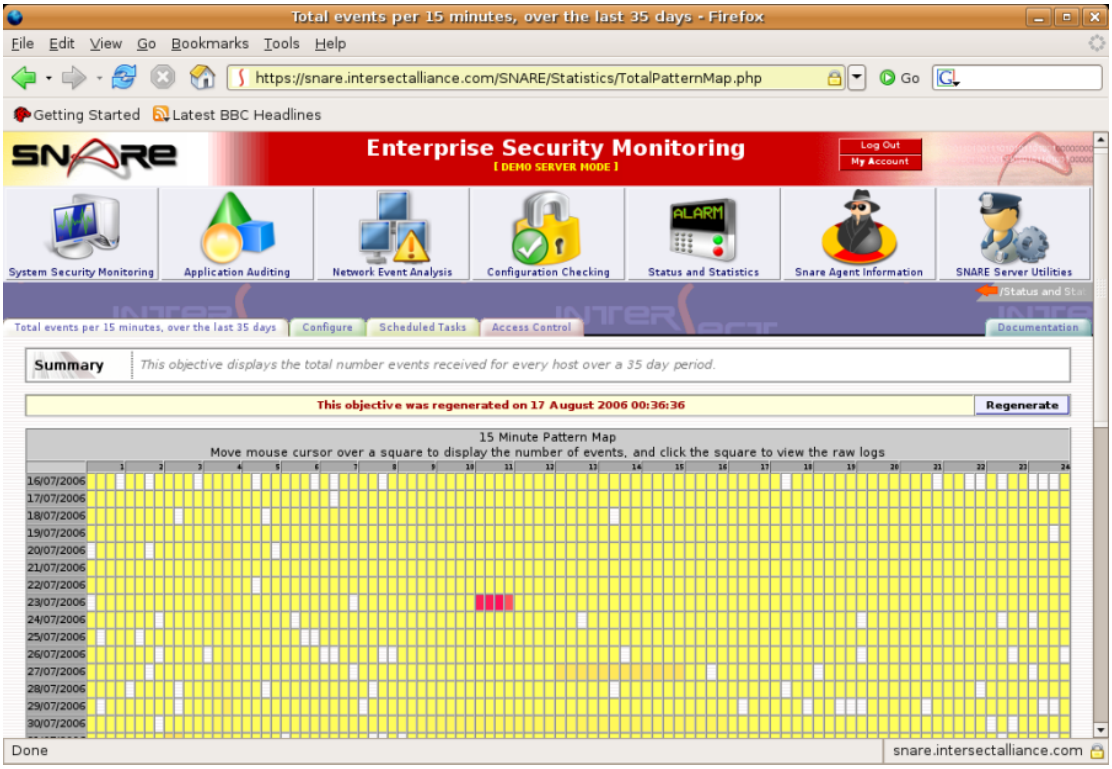

Figure 5: Time Plot of One Week

Figure 6 shows a '***user-time plot***' in which the 15 minute squares are occupied with colored representations of users rather than number of events. In the event where more than one user occupies a given square, then a 'cross-hatched' square is used. Placing the mouse cursor over the square will show the users that occupy a given square ***and*** the number of events against each type of user. As for the time plot graph, clicking a square will take the user to the raw logs which are used to populate the 15 minute 'square'.

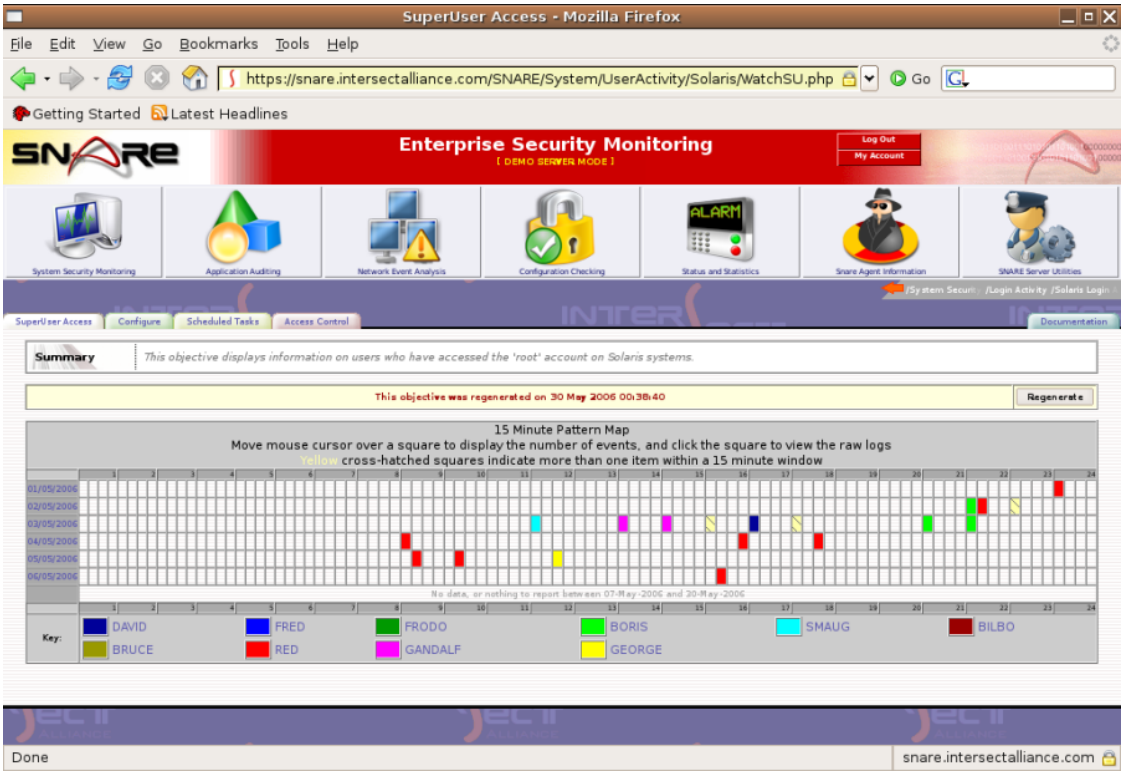

Figure 6: User Time Plot

In some cases, a user time plot may be accompanied by another 2D graph, 3D graph or table showing the details of the raw event logs in easily readable format. In the case of firewall objectives, a graph showing a histogram of the unsuccessful connections to destination ports may be useful, whereas a 'details' table may be useful for the objective showing the users created and deleted.
The type of reporting will thus depend on the information that is required to be displayed, and the Snare Server is able to provide any number of different report formats that cater for these requirements.

9.7. Archival, forensics and redundancy

In any event record collection system, the issues of redundancy, archival and forensics are crucial in meeting the requirements of most normal installations. Since the logs are mostly collected to allow historical review of actions, these three topics are inextricably linked to one another.

The Snare Server will collect log data, and store it on the local hard disk. Once enough data has been saved to the hard disk, on a scheduled or ad-hoc basis, the data on disk may be copied to optical media such as a CD or DVD. If required, a Windows compatible file-server can be started on the Snare Server to allow Windows based archival solutions to securely gain access to, and archive the log data in a format which is readable and accessible by practically any operating system. Since the archived data is stored in 'flat file' format, ie. one that is not marked up in any way, any application or system that is able to read text files will be able to read the archived logs.

The logs themselves are compressed and stored in text format. Compression of logs is around the 13:1 ratio, so the archival process is able to store a significant amount of information. Using DVD to write the logs will allow potentially months of information to be backed up to one optical media disk. The Snare Server allows for data to be either deleted once a successful backup has completed, or to be retained if it is required (for example) that a number of archives be undertaken before the data is removed.

The ideal redundancy infrastructure is to allow the Snare agents to send two or more copies of the same event to 2 or more locations. In fact, the Snare agents do have this capability, and are able to provide full redundancy of event logs at the source end. However, if the Snare Server is collecting events from (say) a SYSLOG appliance, then the Snare Server needs to provide the redundancy at the collection end. In these instances, the Snare Server

implements a "Snare Reflector", which will pass copies of all event logs received to the Snare Server, to another Snare Server. The Snare Reflector works in real time, and is an integrated feature of the Snare Server.

10. About Intersect Alliance

Intersect Alliance, part of the Prophecy International Holdings Group, is a team of leading information technology security specialists. In particular, Intersect Alliance are noted leaders in key aspects of IT Security, including host intrusion detection. Our solutions have and continue to be used in the most sensitive areas of Government and business sectors.

Intersect Alliance intend to continue releasing tools that enable users, administrators and clients worldwide to achieve a greater level of productivity and effectiveness in the area of IT Security, by simplifying, abstracting and/or solving complex security problems.

Intersect Alliance welcomes and values your support, comments, and contributions. For more information on the Enterprise Agents, Snare Server and other Snare products and licensing options, please contact us as follows:

The Americas +1 (800) 834 1060 Toll Free | +1 (303) 771 2666 Denver
Asia Pacific +61 8 8213 1200 Adelaide Australia
Europe and the UK +44 (797) 090 5011
Email intersect@intersectalliance.com
Visit www.intersectalliance.com

11. Appendix A - References

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

1. The GNU General Public License. {+}http://www.gnu.org/licenses/gpl-faq.html+
2. Common Vulnerabilities and Exposures. This is a dictionary of names, which reflect known and candidate vulnerabilities and exposures. See http://www.cve.mitre.org
3. SNORT. This is an Open Source network intrusion detection engine. www.snort.org
4. 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
5. 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
6. SANS Website "Why does my intrusion-detection system generate false alarms / no alarms?" http://www.sans.org/resources/idfaq/false_alarms.php
7. 'Intrusion Detection'. Terry Escamilla. Wiley Computer Publishing. ISBN 0-471-29000-9
8. SANS Institute Intrusion Detection FAQ. http://www.sans.org/resources/idfaq/#top
9. An implementation of an Optical Data Diode http://www.dsto.defence.gov.au/publications/2110/