

# SNARE

System iNtrusion Analysis & Reporting Environment

## Guide to Snare for Solaris

INTERSECT  
ALLIANCE

## Documentation History

Version No.	Date	Edits	By whom
1.0	16 November 2003	First draft for the Guide to Snare for Solaris documentation, in the updated format.	Leigh Purdie
1.1	15 July 2004	Addition of remote control access to users/groups. Included Snare Server issues. Corrections and additions.	George Cora
2.0	2 April 2005	Minor rewording.	George Cora
2.1	30 November 2005	Formatting changes and detailed current use of GUI	George Cora
2.2	15 May 2006	Included documentation for supported agents	David Mohr
2.3	4 June 2008	Updated supported features	David Mohr

© 1999-2008 Intersect Alliance Pty Ltd. All rights reserved worldwide.

Intersect Alliance Pty Ltd shall not be liable for errors contained herein or for direct, or indirect damages in connection with the use of this material. No part of this work may be reproduced or transmitted in any form or by any means except as expressly permitted by Intersect Alliance Pty Ltd. This does not include those documents and software developed under the terms of the open source General Public Licence, which covers the Snare agents and some other software.

The Intersect Alliance logo and Snare logo are registered trademarks of Intersect Alliance Pty Ltd. Other trademarks and trade names are marks' and names of their owners as may or may not be indicated. All trademarks are the property of their respective owners and are used here in an editorial context without intent of infringement. Specifications and content are subject to change without notice.

## About this guide

This guide introduces you to the functionality of the Snare Agent for the Solaris operating system. Snare for Solaris wraps the default Solaris BSM auditing subsystem, and facilitates objective-based filtering, and remote audit event delivery. Snare for Solaris will also allow a security administrator to fully remote control the application through a standard web browser if so desired.

Other guides that may be useful to read include:

- Snare Server User’s Guide.
- Installation Guide to the Snare Server.
- Snare Server Troubleshooting Guide.
- The Snare Toolset - A White Paper.

### Table of contents:

<b>1 Introduction.....</b>	<b>4</b>
<b>2 Overview of Snare for Solaris.....</b>	<b>5</b>
<b>3 Installing and running Snare.....</b>	<b>6</b>
3.1 Snare installation.....	6
3.2 Running Snare.....	7
<b>4 Setting the audit configuration.....</b>	<b>8</b>
4.1 Audit configuration.....	8
4.2 Auditing control.....	8
4.3 Log rotation.....	10
4.4 Objective Configuration.....	11
<b>5 Remote control and management.....</b>	<b>14</b>
<b>6 Retrieving user and group information.....</b>	<b>16</b>
<b>7 Snare Server.....</b>	<b>18</b>
<b>8 About InterSect Alliance.....</b>	<b>20</b>
<b>Appendix A - Event Output Format.....</b>	<b>21</b>
<b>Appendix B - Snare Configuration File.....</b>	<b>23</b>
<b>Appendix C - Known BSM 'bugs'.....</b>	<b>25</b>

## 1 INTRODUCTION



The team at InterSect Alliance have experience with auditing and intrusion detection on a wide range of platforms - Solaris, Windows NT/2000/2003/XP, Netware, Tru64, Linux, AIX, IRIX even MVS (ACF2/RACF); and within a wide range of IT security in businesses such as - National Security and Defence Agencies, Financial Service firms, Government Departments and Service Providers.

This background gives us a unique insight into how to effectively deploy host and network intrusion detection systems that support and enhance an organisation's business goals.

The development of 'Snare for Solaris' will allow event logs from the Solaris BSM Basic Security Module (BSM) to be collected from the operating system, and forwarded to a remote audit event collection facility. Snare for Solaris will also allow a security administrator to fully remote control the application through a standard web browser if so desired. Snare has been designed in such a way as to allow the remote control functions to be easily effected manually, or by an automated process.

In the spirit of the release of the Snare agents, InterSect Alliance are proud to release Snare for Solaris as an open source initiative. Other event audit modules for AIX, IRIX, Linux, Windows and other applications have been released under the terms of the GNU Public License. The overall project is called 'Snare' - **System iNtrusion Analysis & Reporting Environment**. The '*Snare Server*' is a commercial release of software beneficial to organisations that wish to collect from a wide variety of Snare agents and appliances such as firewalls or routers.

InterSect Alliance welcomes and values your support, comments, and contributions. Our contact details are available from our contact page at [www.intersectalliance.com](http://www.intersectalliance.com).

## 2 OVERVIEW OF SNARE FOR SOLARIS

Snare operates through the actions of the '*SnareCore*' daemon, which interfaces with the Solaris kernel.

The *SnareCore* service interfaces with the Solaris kernel to read, filter and send event logs from the BSM event logging sub-system (known in Solaris as the Basic Security Module - BSM) to a remote host. The events to be sent will depend on the objectives chosen, and not on the configuration of the BSM files. Snare will automatically take control of the BSM subsystem without the requirement of any System Administrator intervention. The logs are then filtered according to a set of 'objectives' chosen by the administrator, and then passed over the network, using the UDP or TCP protocol, to a remote server. *The TCP protocol capability, and the ability to send events to multiple hosts is only available to those users that have purchased a Snare Server, through the supported agents. See Chapter 7 of this document for further details.* The *SnareCore* daemon is able to be remotely controlled using a standard web browser, or via a custom tool that acts as a web client.

The *SnareCore* daemon reads event log data directly from the Solaris kernel. *SnareCore* converts the Solaris event log from binary format to text format. Unfortunately, the Solaris BSM audit subsystem has a number of identified 'bugs', which are further detailed in Appendix C - Known BSM Bugs. For this, and other reasons, the *SnareCore* daemon incorporates a 'wrapper' program which will watch for these failures, and restart the service if a key Solaris component fails. It does however, mean that the event that caused the fault to be lost, is an unavoidable consequence of the bugs. The event sent by the *SnareCore* daemon is a text-format, TAB separated series of tokens, which are described in detail by the BSM documentation. This format, is also discussed in Appendix A - Event Output Format. The net result is that a *raw* event, as processed by the *SnareCore* daemon may appear as follows:

```

phoenix SolarisBSM      1      header,146,2,execve(2),,Mon Dec 9 22:23:42 2002, + 140001416 msec
      path,/usr/bin/grep  attribute,100555,root,bin,136,379861,0  exec_args,2,grep,snare
      subject,red,root,other,root,other,12228,12212,8236 131095 10.0.1.1  return,success,0
      sequence,65941
    
```

All of the fields in the above record are sent to the remote server, whether this is a Snare Server, or a custom tool. The Snare Server is then used to interpret some or all of the tokens in the log file to determine the value of the event to the security or system administrators.

## 3 INSTALLING AND RUNNING SNARE

### 3.1 SNARE INSTALLATION

The Snare installation file is available in tarball format, and includes an installation script to allow for simple installation and configuration of all critical components. The 'snareinstall-*<version>*.tar.gz' file (where '*<version>*' is the version of the Snare for Solaris Agent), includes the following critical components.

#### ▶ WHAT YOU NEED...

- **SnareCore**  
The audit daemon is provided by the '*SnareCore*' binary. This binary contains all the code required to read the event log records, filter the events according to the 'objectives', provide a web based remote control interface, handle known Solaris bugs, configure the BSM subsystem based on required objectives, and provide all the necessary logic to allow the binary to act as a daemon.
- **install.sh/uninstall.sh**  
These two scripts undertake the installation and uninstall functions required to ensure Snare for Solaris works as required. The scripts prompt the user for guidance on the type of installation to be undertaken (discussed in detail below). The script is designed to be executed interactively.
- **Configuration files**  
A myriad of configuration files are required to correctly run the BSM sub-system. These configuration files have been tailored to meet Snare requirements, and include the files: audit\_event, audit\_control, snare.conf, and so on. These configuration files are copied to the /etc/security directory during the installation process.

It is important to remember that a Solaris installation does not normally *activate* the utilities necessary to run the auditing sub-system. As such, it must be separately *activated* on the Solaris host, before the *SnareCore* and associated files will work in collecting and filtering events. The next section details how the auditing sub-system may be *activated* on a Solaris workstation using the 'bsmconv' script.

#### ▶ HOW TO...

#### Install the Snare package for Solaris

1. Download 'snarecore-*<version>*.tar.gz' from the Intersect Alliance website.
2. Ensure that the BSM subsystem has been installed by typing 'bsmconv' at a root prompt.
3. As 'root', type 'tar -zxvf snarecore-*<version>*.tar.gz' at the root prompt, where *<version>* is the latest version of the Solaris agent available on the Intersect Alliance web site. A 'snarecore-*<version>*/' directory will be created. Enter this directory by typing 'cd snarecore-*<version>*'.
4. In order to commence the installation, type in './install.sh'. A series of prompts will then be displayed, requesting that various parameters be set. Read these settings carefully, using this manual as reference. Most of the references are discussed later in this guide, so it pays to read this guide first, before installing the software.
5. Once the installation process has completed the *SnareCore* daemon should be started by issuing the command '/etc/init.d/snare restart'. However, in order to ensure that auditing is applied to all processes, it is recommended that your Solaris system be rebooted - particularly if you have just run the 'bsmconv' script.

### 3.2 RUNNING SNARE

The *SnareCore* daemon must be running if the events are to be passed to a remote host. The *SnareCore* daemon may be stopped, started or restarted, by issuing the command: '/etc/init.d/snare stop', '/etc/init.d/snare start' or '/etc/init.d/snare restart', respectively.

## 4 SETTING THE AUDIT CONFIGURATION

### 4.1 AUDIT CONFIGURATION

The audit configuration is stored as `/etc/security/snare.conf`. This file contains all the details required by the audit daemon to successfully execute. Failure to have a correct configuration file available in this location will not 'crash' the daemon, but may result in selected events not being able to be read.

**Tip:** Manual editing of the `snare.conf` configuration file is possible, but care should be taken to ensure that it conforms to the required format for the audit daemon. Also, any use of the Remote Control Interface to modify security objectives or selected events, may result in manual configuration file changes being overwritten. Details on the configuration file format can be viewed in Appendix B - Snare Configuration File.

The most effective and simplest way to configure the Snare audit daemon is to use the Remote Control Interface. The Remote Control Interface can be access locally via the URL `http://localhost:6161` or remotely via `http://hostname:6161` where "hostname" is the DNS name or IP address of the target machine.

### 4.2 AUDITING CONTROL

The initial audit configuration parameters to consider are:

- The hostname, IP address and UDP or TCP port of the remote collection servers,
- The requirement to maintain a log file, or send the events to a remote server or servers, or both, and
- The location of the logfile (optional).
- The following features are only available to users who purchase a Snare Server. These are not part of the Open Source toolset. See Chapter 7 below for more details on the supported versions of the Snare agents.
  - TCP - Selecting this option will guarantee message delivery across the network.
  - Define multiple servers - This feature allows you to define multiple destination hosts as well as multiple remote control hosts.
  - Event cache size - For each TCP host, a limited cache will be available to store events if the agent cannot communicate with the Snare Server. This value should be between 1 and 1024 MB.
  - Encrypt Messages - Encrypt messages between the agent and the Snare Server. This option requires matching Remote Access passwords on both the agent and the Snare Server.

These parameters can be configured manually through the `snare.conf` file, or through the remote control interface discussed in Chapter 5. Figure 1 shows a web browser connecting to a Snare agent via the Remote Control Interface.

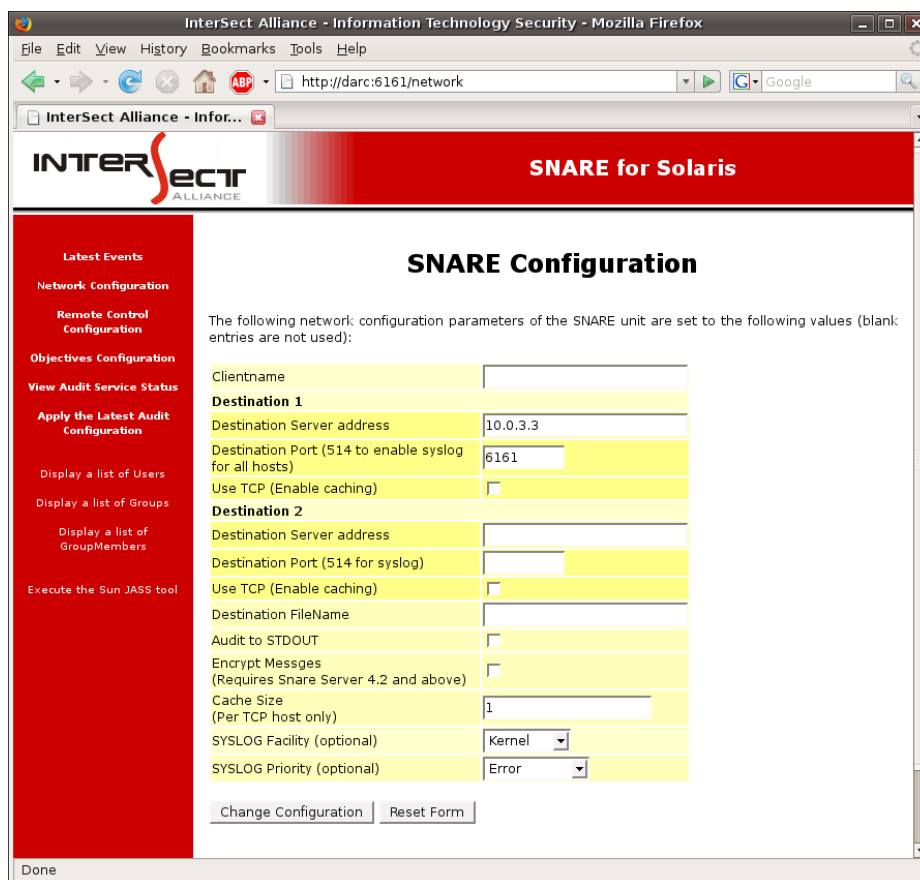


Figure 1 Network Configuration Window

The Clientname field can be used to override the name that Snare automatically picks up from the host's fully qualified domain name. Unless a different name is required to be sent in the processed event log record, leave this field blank. Note that executing the command `hostname` on a command prompt will display the current host name allocated to the system.

Snare provides the facility to send events to one or more remote hosts, using UDP or TCP. In conjunction with the audit log archive script provided within the Snare distribution, this will facilitate the remote storage of audit logs for later analysis. *The TCP protocol capability, and the ability to send events to multiple hosts is only available to those users that have purchased a Snare Server, through the supported agents. See Chapter 7 of this document for further details.*

The Destination Port is the remote server's port that will be used to collect the events (currently, TCP and multiple hosts can only be enabled through the supported agents). If, for example, the Snare Server is installed on a remote system, and is listening on port '6161', then the Solaris agent will need to be configured with the IP address of the remote Snare Server, and port 6161 in the 'port number' field. Since Snare for Solaris receives events directly from the kernel, there is also a configuration option to write events to a local file. Note however, that writing events to disk is completely optional if you only wish to forward events to a remote server without storing them locally.

---

A simple collection and archive script for receiving logs from a remote Snare node, is available from the Snare project page. The collection/archive script receives data from one or more Snare clients, and saves the data off to a file per-date, per-system in /var/log/audit. Files will be given names based on the date, and the audit source, in the format YYYYMMDD-host.name.SolarisBSM (eg. /var/log/audit/20040321-test.intersectalliance.com.SolarisBSM).

### **4.3 LOG ROTATION**

Depending on the Snare configuration, the log file may be small or large. In any case, it is normal housekeeping practice that logs either be rotated or archived. Depending on the site requirements, a rotation scheme that keeps old copies of the last (say) 7 days may be sufficient. In this case, it may be sufficient to simply include a CRON job, and use a program such as logrotate to ensure the current log file does not grow to an unmanageable size. Alternatively, you may wish to archive all log files to backup media such as tape or CDROM. This may be scheduled using CRON or undertaken manually. In either case, it is important to note that the audit daemon should be restarted, so that it opens the new log file for writing the events.

## 4.4 OBJECTIVE CONFIGURATION

A major component of the Snare audit subsystem is the capability to filter events. This is accomplished using auditing 'objectives'. Any number of objectives may be specified, and are displayed within the 'Objectives' tab, in the Remote Control Interface (see Chapter 5).

Each of the objectives provides a significant level of control over which events are selected and reported. Events are grouped into broad categories, which include system calls common to a particular audit task, however, individual system calls will also be accepted by the *SnareCore* daemon. The 'high level' groups are as follows:

- Read, write or create a file or directory.
- Modify system, file or directory attributes.
- Start or stop a program execution.
- Change user or group identity.
- Open a file for reading only.
- User logon or logoff.
- Establish an outgoing network connection.
- Any event(s).

Note that the groups above are provided to service the most common security tasks required by a system or security administrator. If other types of events are required, then the 'Any event(s)' objective will allow the administrator to specify arbitrary audit events. From each of these groups, a criticality level can be applied. These criticality levels are critical, priority, warning, information and clear. These security levels are provided to enable the Snare administrator to map audit events to their most pressing business security objectives, and to quickly identify the criticality of an event.

The following filters can be applied to incoming audit events:

### 1. Filter on the event-specific matchable item

Each event contains a particular piece of information that represents the core data that needs to be communicated. For the 'Open a file for reading only' group, for example, this would be the name of the file and/or directory opened or created. For the 'Start or stop a program execution' group, this would be the name of the program executed. The event match allows a 'partial', 'exact' or 'regular expression' match term to check against the event-specific matchable item. A 'partial' match will look for the sequence of characters specified somewhere within the event-specific matchable item. For example, if the partial match of 'pass' is specified for 'Read, write or create a file or directory', then the following example filenames would all match the term:

- /etc/passwd
- /usr/lib/passfilt.so
- /home/red/khyber\_pass.txt

An 'exact' match will match the specified term exactly. For example, if the term is /etc/passwd, then the file /etc/passwd would match, but the file /etc/passwd.backup will NOT match.

A 'regular expression' match matches the supplied extended regular expression against the event-specific matchable item. Regular expressions are an advanced form of specifying filters, and should only be used by those with regular expression experience. For example, the term `'.*[Pp]ass(word|wd).*` would match the following:

- `/etc/passwd`
- `/tmp/PasswordFile`

but would not match

- `/etc/PASSWD/`
- `/home/red/PaSsWoRd.txt`

Regular expression EXCLUSIONS are also possible, using the following syntax: `[^(searchstring)]`. For example: `^/etc/[^(passwd)]` would match:

- `/etc/shadow`
- `/etc/group`
- `/etc/PASSWD`

but would not match

- `/etc/passwd`

## 2. Filter on user

Any number of users can be selected, and should be entered with commas separating each user. If no users are entered, ALL users are assumed to be audited. Alternatively, specific users may be EXCLUDED from any individual objective, leading to objectives such as "tell me whenever any user except 'root' or 'red' generate an event". If the user exclusion function is selected, Snare will only report users that DO NOT match the supplied list of users.

**NOTE:** If this is being undertaken via the web (remote control) pages, then this field must be treated as a regular expression. Multiple users may be selected, but since it is a regular expression field, the users must be separated by the pipe symbol '|'. So in the case of specifying the users 'root' and 'red', this must be written as `'^(root|red)$'`.

## 3. Filter on return value

An audit event will either return a success or failure return code. Snare allows a user to filter on the return value.

## 4. Filter on special, event-specific fields

Some events, including `open()` and `socketcall()`, allow additional filters to be specified, to provide more flexible search criteria.

- `open()`

The open event provides the additional capability to filter on open-flags. The flags are specified in regular expression format, and can include (in the following order):

O\_WRONLY  
 O\_RDWR  
 O\_RDONLY  
 O\_CREAT  
 O\_EXCL  
 O\_NOCTTY  
 O\_TRUNC  
 O\_APPEND  
 O\_NONBLOCK  
 O\_SYNC  
 O\_NOFOLLOW  
 O\_DIRECTORY  
 O\_LARGEFILE

For example, the following flags can be logically 'or'ed together (in regular expression form) to specify an objective that translates to 'Let me know whenever a file is opened in WRITE mode':

- `open(O_WRONLY|O_RDWR|O_CREAT|O_TRUNC|O_APPEND)`

Whereas the following three examples all specify 'Read or Write':

- `open(.*)`
- `open(O.*)`
- `open(O_WRONLY|O_RDWR|O_RDONLY|O_CREAT|O_EXCL|O_NOCTTY|O_TRUNC|O_APPEND|O_NONBLOCK|O_SYNC|O_NOFOLLOW|O_DIRECTORY|O_LARGEFILE)`

More information on the flags associated with the `open()` system call are available from the Solaris manual pages (see 'man open').

- `socketcall()`

The Solaris kernel uses the 'socketcall' system call to serve the requirements of the 'connect', 'accept' and related system calls. In order to monitor 'connect' and 'accept' calls only, the system call 'type' can be included within the objective.

For example, `socketcall(ACCEPT)` only monitors `accept()` calls. `socketcall(CONNECT)` only monitors `connect()` calls. `socketcall(.*)` or `socketcall(CONNECT|ACCEPT)` monitor both `accept` and `connect`.

Once the above settings have been finalized, they need to be saved to the 'snare.conf' file either manually or using the Remote Control Interface (see Chapter 5). Figure 2 shows a web browser connecting to a Snare agent.

However, to ensure the *SnareCore* daemon has received the new configuration, the *SnareCore* daemon **MUST** be restarted. If using the Remote Control Interface function the *SnareCore* daemon can be restarted via the 'Apply the Latest Audit Configuration' menu item, or by issuing the command: `/etc/init.d/audit restart` on a command line (as root).

## 5 REMOTE CONTROL AND MANAGEMENT

The **SnareCore** service is a separate standalone component of the Snare for Solaris package. The **snare.conf** file can be used to manage the tool. However, the audit configuration can also be managed using the Remote Control Interface.

The audit daemon can be restarted directly from the *command* `/etc/init.d/snare restart (as root)`. This will instruct the audit daemon to re-read the configuration file, clear the buffers and restart. Note that only the 'root' user can restart the **SnareCore** Service. This function is useful when changes to the audit configuration have simply been saved to the configuration file, without being 'applied'. The user can therefore select when to activate a new configuration by executing this command. This restart process is shown as follows:

- As root, execute the command: `ps -ef | grep snare`
- It should return something like:

```
bash-2.05# ps -ef | grep snare
   root   459   458   0 11:19:10 ?           0:00 /etc/security/snarecore
   root   460   458   0 11:19:10 ?           0:00 /etc/security/snarecore
   root   458     1   0 11:19:10 ?           0:00 /etc/security/snarecore
   root   477   425   0 12:05:04 pts/2       0:00 grep snare
bash-2.05#
```

- As root, execute the command: `/etc/init.d/snare stop`
- As root, execute the command: `/etc/init.d/snare start`
- (or `/etc/init.d/snare restart` in place of the above 2 commands)
- As root, execute the command: `ps -ef | grep snare`, and check that the processes have been restarted by ensuring the '**snarecore**' processes have new process IDs.

A significant function of the **SnareCore** service is its ability to be remote controlled. This facility has been incorporated to allow all the functions normally available through a front end Snare tool, to instead be available through a standard web browser. The **SnareCore** service employs a custom designed web server to allow configuration through a browser, or via an automated system such as the Snare Server's "Check and Set Configuration" feature. Figure 2 below shows a web browser connecting to a Snare agent.

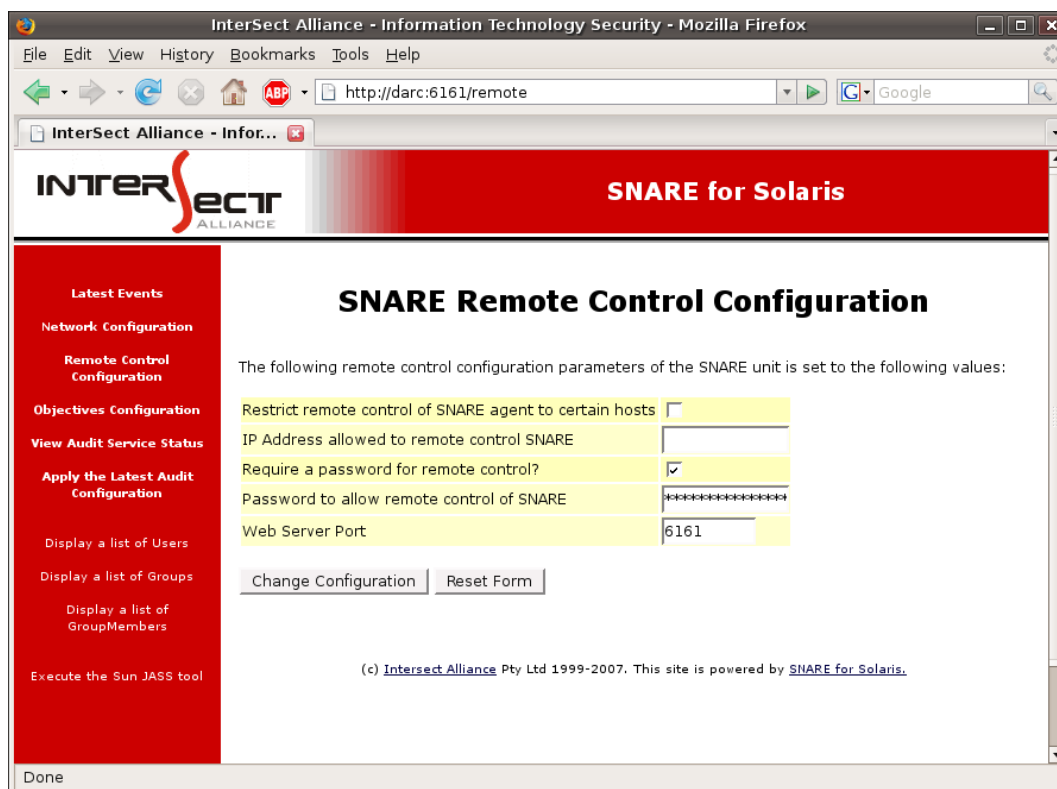


Figure 2 Remote Control Window

The functions available through the web browser are identical to those available through the *snare.conf* file. In either case, the actual remote control parameters may be controlled in a similar fashion as the audit configuration. The parameters which may be set for remote control operation are shown in Figure 2 and discussed in detail below:

- **IP Address allowed to remote control Snare.** Remote control actions may be limited to a given host. This host, entered as an IP address in this field, will only allow remote connections to be effected from the stated IP address. Note that access control based on a source IP address is prone to spoofing, and should be considered as a security measure used in conjunction with other countermeasures.
- **Password to allow remote control of Snare.** A password may be set so that only authorised individuals may access the remote control functions. If accessing the remote control functions through a browser or custom designed tool, note that the UserID is always 'snare', and the password is whatever has been set through this setting. This password is not encrypted when being transmitted via the http session, but is encrypted when stored in the *snare.conf* file.
- **Web Server Port.** Normally, a web server operates on port 80. If this is the case, then a user need only type the address into the browser to access the site. If however, a web server is operating on port (say) 6161, then the user needs to type **http://mysite.com:6161** to reach the web server. The default SnareCore web server port may be changed using this setting, if it conflicts with an established web server. However, care should be taken to note the new server port, as it will need to be placed in the URL needed to access the Snare agent.

## 6 RETRIEVING USER AND GROUP INFORMATION



The *SnareCore* daemon also has the ability to retrieve users, groups and group membership from accounts local to the host that is running the agent. In those cases where the primary authentication source is defined as a NIS+ domain or an LDAP directory, then the users, groups and group membership retrieved will be those defined in the remote domain.

This feature is available through the Remote Control Interface, and can be accessed through any standard web browser. The menu structure on the remote web pages (as shown in Figure 2) shows the selections 'Display a list of Users', 'Display a list of Groups', 'Display a list of GroupMembers'. Selecting any of these items will then display the relevant details. For example, Figure 3 below shows the output of selecting 'Display a list of Users'. The output from these commands has been designed with no HTML markup, so as to assist automated services, such as the Snare Server, to interrogate the users, groups and group membership.

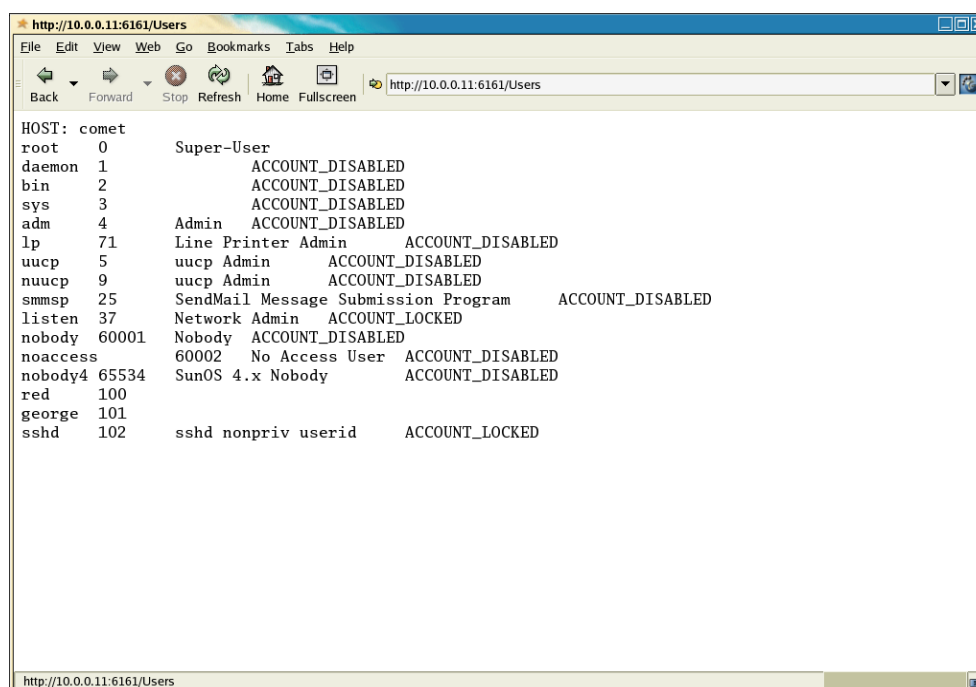


Figure 3 Output of 'Display a list of Users'

In the case of 'Display a list of Users', the output shows a number of tab delimited entries, per line. These entries should be interpreted as follows:

- a. **Username; UID; Description;** Password Expired (token will be: PASSWD\_EXPIRED); Account Inactive (token will be: ACCOUNT\_INACTIVE); Account Expired (token will be: ACCOUNT\_EXPIRED); Account Disabled (token will be: ACCOUNT\_DISABLED); Account Locked (token will be: ACCOUNT\_LOCKED); No Password (token will be: NO\_PASSWD).

---

The first three entries of Username, UID and description shown in bold and italics above will be displayed and be tab delimited. The remaining tokens will only be shown if they exist on a particular account.

In the case of Groups and Group Membership, the attributes displayed are ***Groupname; GID; Group Members***. Obviously, the group member list will only be shown when selecting the 'Display a list of GroupMembers' menu item from the Remote Control Interface. Additionally, the group members will be displayed as a comma separated list of usernames. As stated previously, the groups and associated membership displayed via the web browser may relate to NIS+ or LDAP users and groups, if the local host that is running the Snare for Solaris agent has been configured to derive user and group information from a domain.

## 7 SNARE SERVER



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.

In addition to the above, the benefits of purchasing the Snare Server include:

- Official support mechanism for the Snare open source agents. Note that official Snare agent support is not offered through *any* other channels.
- All future Snare Server versions and upgrades included as part of an annual maintenance fee.
- Ability to collect any arbitrary log data, either via UDP or TCP protocols.
- Proven technology that works seamlessly with the Snare agents.
- Snare reflector technology that allows for all collected events to be sent, in real time, to a standby/backup Snare Server.
- Ability to continuously collect large numbers of events. Snare Server collection rates exceed 60,000 events per minute using a low end, workstation class, Intel based PC on a 100Mbps network.
- Ability to drill down from top level reports. This reduces the amount of data “clutter” and allows a system administrator to fine tune the reporting objectives.
- Ability to create “cloned” objectives that allow very specific reporting against any collection profile. These reports, along with all Snare Server objectives, may be scheduled and emailed to designated staff.
- The Snare Server uses extensive discriminators for each objective, allowing system administrators to finely tune reporting based on inclusion or exclusion of certain parameters.
- Very simple, single CD installation for those users not requiring a hardware based appliance.

A screenshot from the Snare Server is shown in Figure 4. The Snare Server uses a hardened version of the Linux operating system base for stability and its ability to use a myriad of stable and functional open source tools. A Snare Server user, however need not be concerned with managing a Linux server. The Snare Server, once installed, is a fully contained appliance, and does not require any system administrator level maintenance. The Snare Server will operate on commonly available Intel based PCs, with hardware specifications shown on the next page.

There are supported versions of the Snare agents which are only available through the purchase of a Snare Server. Functionality includes, but is not limited to, ability to send events via TCP as well as UDP, and the ability to send events to many destinations, not just one host.

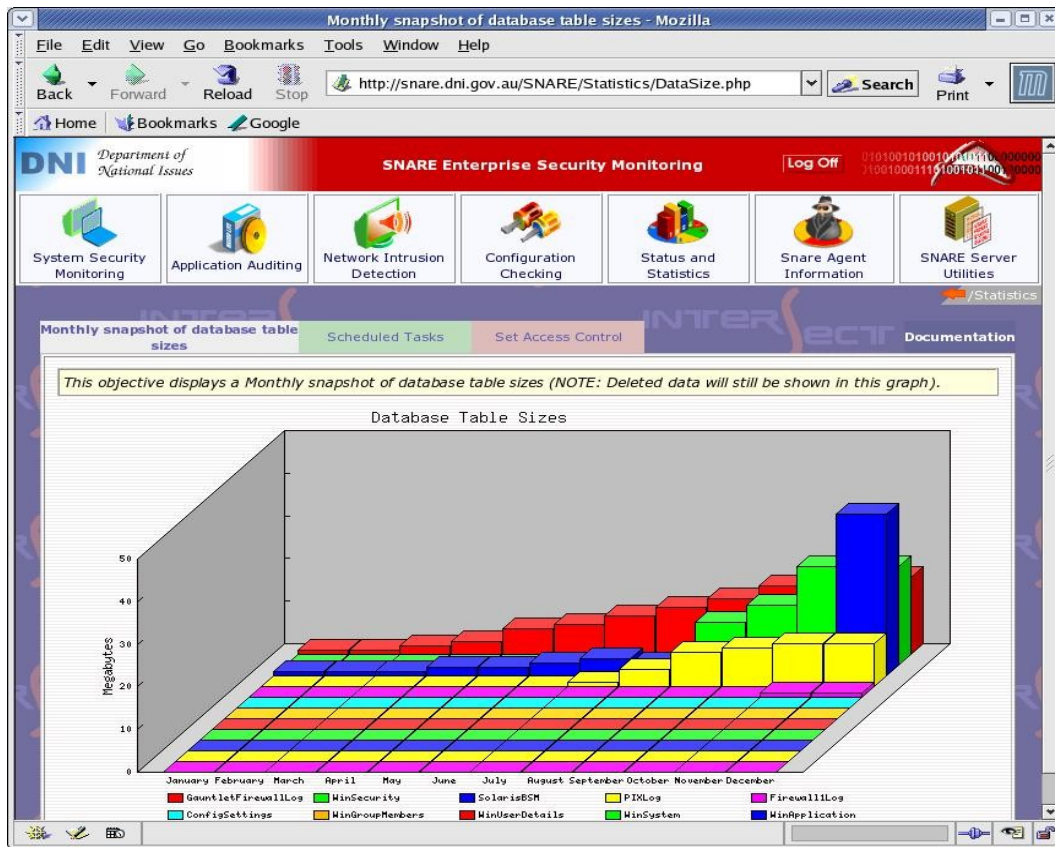


Figure 4 Screen shot of the Snare Server

## 8 ABOUT INTERSECT ALLIANCE



InterSect Alliance is a team of leading information technology security specialists in both the 'technical' and 'policy' areas. 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 consult and contract to number of agencies in Australia and the Asia Pacific, for both the business and Government sectors.

The Intersect Alliance business strategy includes demonstrating our commitment and expertise in IT security by releasing open source products such as Snare, and the proprietary Snare Server. 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.

Visit the Intersect Alliance website for more information at [www.intersectalliance.com](http://www.intersectalliance.com).

## APPENDIX A - EVENT OUTPUT FORMAT

The *SnareCore* daemon reads data from the Solaris kernel, via published APIs. It converts the binary audit data into text format using the 'praudit' utility, and separates information out into a series of token/data groups. Three different field separators are used in order to facilitate follow-on processing - TABS separate 'tokens', COMMAS separate data within each token, and SPACES separate elements within data.

A 'Token' is a group of related data, comprising a 'header', and a series of comma separated fields which make up data that relates to the header.

Examples of tokens:

- process,1628,gcc
- return,0
- path,/etc/audit/audit.conf
- arguments,ls -al

Groups of tab separated tokens make up an audit event, which may look something like this, depending on whether the audit daemon has been set to 'objective' or 'event' reporting mode (see the configuration section for more information):

```
phoenix SolarisBSM      1      header,146,2,execve(2),,Mon Dec 9 22:23:42 2002, + 140001416 msec
path,/usr/bin/grep      attribute,100555,root,bin,136,379861,0  exec_args,2,grep,snare
subject,red,root,other,root,other,12228,12212,8236 131095 10.0.1.1      return,success,0 sequence,
65941
```

A simple example PERL script for extracting data from a raw Snare log is as follows:

```
#!/usr/bin/perl
# Usage: cat /var/log/audit/audit.log | ./extract.pl
# Creates an associative array containing the elements of the event record, and prints the data.

while($input=<STDIN>) {
    chomp($input);
    %Record=();
    @tokens=split(/\t/, $input);    # Split the line into TAB delimited tokens.
    foreach $token (@tokens) {
        @elements=split(/,/, $token);    # Pick out the elements within each token.
        $header=$elements[0];
        if($header eq "objective") {
            $Record{$header}{"criticality"} = $elements[1];
            $Record{$header}{"datetime"} = $elements[2];
            $Record{$header}{"description"} = $elements[3];
        } elsif ($header eq "event") {
            $Record{$header}{"eventid"} = $elements[1];
            $Record{$header}{"datetime"} = $elements[2];
        } elsif ($header eq "user") {
            $Record{$header}{"uid"} = $elements[1];    # User ID
            $Record{$header}{"gid"} = $elements[2];    # Group ID
            $Record{$header}{"euid"} = $elements[3];    # Effective User ID
            $Record{$header}{"egid"} = $elements[4];    # Effective Group ID
        } elsif ($header eq "process") {
            $Record{$header}{"pid"} = $elements[1];    # Process ID
            $Record{$header}{"name"} = $elements[2];    # Process Name (max 16 chars)
        } elsif ($header eq "path") {
            $Record{$header}{"path"} = $elements[1];
        } elsif ($header eq "destpath") {
```

```

    $Record{$header}{"destpath"} = $elements[1];    # Destination path
} elseif ($header eq "arguments") {
    $Record{$header}{"args"} = $elements[1];
} elseif ($header eq "attributes") {
    $Record{$header}{"attrib"} = $elements[1];
} elseif ($header eq "return") {
    $Record{$header}{"code"} = $elements[1];
} elseif ($header eq "target") {
    $Record{$header}{"user"} = $elements[1];
} elseif ($header eq "owner") {
    $Record{$header}{"user"} = $elements[1];
    $Record{$header}{"group"} = $elements[2];
} elseif ($header eq "socket") {
    $Record{$header}{"sourceip"} = $elements[1];
    $Record{$header}{"destip"} = $elements[2];
    $Record{$header}{"sourceport"} = $elements[3];
    $Record{$header}{"destport"} = $elements[4];
} elseif ($header eq "sequence") {
    $Record{$header}{"number"} = $elements[1];
}
}
# We now have the data in an associative array.
# Roll through the array, and print the data in token groups.
foreach $header (keys(%Record)) {
    print "Header: $header\n";
    foreach $element (keys(%{$Record{$header}})) {
        print "$element = " . $Record{$header}{$element} . "\n";
    }
}
# In addition, if the event is execve, the effective user ID
# is 'root', but the real user ID is NOT, then display an alert.
if($Record{"event"}{"eventid"} eq "execve" && $Record{"user"}{"euid"} eq "root" &&
$Record{"user"}{"uid"} ne "root") {
    print "Danger: SetUID program " . $Record{"arguments"}{"args"} . " has been run by the user
" . $Record{"user"}{"uid"} . " .\n";
}

print "\n";
}

print "----- Done ----- \n";

```

## APPENDIX B - SNARE CONFIGURATION FILE

Details on the audit configuration are discussed in the [Audit Configuration](#) section. The purpose of this section is to discuss the makeup of the configuration file. The Snare configuration file is located at `/etc/security/snare.conf`, and this location may not be changed. If the configuration file does not exist, the *SnareCore* daemon will execute, but will not actively audit events until a correctly formatted configuration file is present, or unless specific instructions are passed to the audit module at load time.

Snare can be configured in several different ways, namely:

- Via the web server,
- By setting module options at load time, or
- By manually editing the configuration file

The format of the audit configuration file is discussed below.

[HostID]	This item stores the hostname, if it is different from the assigned Solaris hostname.
name=<hostname>	This is the name of the host.
[Output]	By default, if no output section exists within the configuration file, the audit daemon will send audit data out to standard out (STDOUT). Note that audit events will be sent to all valid destinations specified in the Output section. As such, events can be sent to one or all of a file, standard output and to a remote network destination (Only one file destination is supported however).
cache_size=5	If for any reason an event cannot be sent to a TCP configured host, the unsent events will be cached up to the defined number of megabytes (e.g. in this case, 5MB).
network=hostname:port:tcp network=hostname:port	Audit data can be sent to a remote system using the UDP (default) or TCP protocol. Data will be sent to the remote host, and network port specified here. Each additional host must be specified on a new line. Caching will be enabled for each host if TCP is enabled.
network=stdout file=stdout	If stdout is specifically defined within the Output section, the audit daemon will send data to standard out.
file=/fully/qualified/file/name	The audit daemon will send data to the fully qualified filename specified within the [Output] section. Note that if the audit daemon is not running as root, the file must be writable by the user under which the audit daemon is running.
encrypt_msg=1	If enabled (1 to enable, 0 to disable), all events sent by the audit daemon will be encrypted in a format compatible with the Snare Server.
[Objectives]	This section describes the format of the objectives. Objectives are composed of: <ol style="list-style-type: none"> <li>1. Criticality - an integer between 0 and 4 that indicates the severity of the event. 0 is "clear", 4 is "critical".</li> <li>2. The event ID - this must either correspond to a valid auditable event, or be set to "*" for any event. Note that</li> </ol>

the Remote Control Interface will convert the generic "groups" in the Audit Configuration window to the required events. For example, the abstracted group "Remove a file or directory", will result in the event entry "event=rmdir,unlink" being written, with the events comma delimited. Note also that additional filter flags may be specified, as discussed in section 4 above.

3. The return code defines whether to report event (system call) if it is a success, failure or both ("\*\*")
4. The user list is listed is used to audit events for selected users, and is in extended regular expression format.
5. The match term is the filter expression, and is again defined in extended regular expression format.

Note that whitespace will be trimmed from the start and end of items, but will be assumed to be valid when bracketed by other characters.

```
criticality=1 event=open_r return=*
user:^(red|george)$ match=^/etc/shadow$
```

Report at criticality level 1, whenever the users "red" or "george", open the file /etc/shadow in READ ONLY mode.

## [Remote]

This subkey stores all the remote control parameters.

allow=1

"Allow" is an integer, and set to either 0 or 1 to allow remote control. 1= allow remote, 0=do not allow.

listen\_port=6161

This value is the web server port. A missing "listen\_port" will default the web server to port 80.

restrict\_ip=10.0.0.1

This is an IP address, that will be used so that this address will be the only host that is allowed to connect to the web server. If this item does not exist, then the web server will not restrict by IP address.

accesskey=snare

This value is the password that is used to log into the Snare web server. If this item does not exist, then a password will not be requested when connecting to the web server. The password is encrypted when stored in the snare.conf, using the standard UNIX "crypt" facility (using MD5 encryption, if available).

hashkey=<md5sum>

Exactly the same as the accesskey, except that it is always an md5sum of the given password.

## APPENDIX C - KNOWN BSM 'BUGS'

The Solaris BSM subsystem has a number of 'bugs' (or 'features') which have been identified and documented in the course of the Snare for Solaris development process. These are detailed below:

- praudit dies when a network related system call is processed. Snare detects praudit failure, and restarts the process.
- Event log formatting relating to time is inconsistent - some events include a comma between seconds and milliseconds, others do not.
- The 'auditon' system call ONLY turns on system-related audit events (ie. events with an ID less than 512).
- Forked processes do NOT inherit audit settings. Each process is re-applied with the contents of /etc/security/audit\* at startup.
- The Solaris +cnt flag doesn't seem to work correctly on some versions of Solaris. Audit must be explicitly turned off, or buffer overflows can occur.
- A auditsvc call with a disk-space size of 0 will cause a kernel panic without appropriate Solaris 6/7 patches.
- Using a pipe as the output file for auditsvc will cause a kernel panic without appropriate Solaris 6/7 patches.
- Taking some of the new Solaris 9 extra audit events out of the audit\_event file causes the audit subsystem to fail.
- Solaris 6 requires at least one more patch over and above those discussed in the Snare installation script - we have yet to find out which it is, but any system that has applied the Sun Microsystems recommended patches post 2001 should be fine.
- Applying audit to the 'nscd' process will cause a kernel panic eventually.
- auditconfig -conf makes a range of decisions w.r.t the processes that need to have auditing applied, but it's decision making process is opaque, and some processes (including nscd, thankfully), are missed out. This leads to some reasonably important processes (such as the mount daemon, and openssh) being ignored by the Solaris BSM subsystem.
- Sun Microsystems have added a 'feature limiter' in Solaris 9, probably in response to complaints that BSM takes too much CPU or disk space. The auditconfig 'public' flag (off by default), ENABLES read-auditing for 'public' files. Public files are defined as files that are owned by root, readable by all, and not writable by all. Unfortunately, this means files such as '/etc/passwd', '/etc/group', '/etc/security/crypt.conf', and similar files, that are important from a security perspective, are NOT audited by default. The Snare init script now turns these on.