DREnet Traffic Analysis

Lessons learned in the deployment of an IDS on the DREnet

Julie Lefebvre
Information Operations, DREO

Joanne Treurniet
Information Operations, DREO

Defence Research Establishment Ottawa

Technical Memorandum

DREO TM 2001-100

November 2001

Abstract

To better understand the needs of intrusion analysts, the SHADOW Intrusion Detection System (IDS) was deployed on the DREnet. All network traffic at the DREnet-Internet border in a 16-day period was collected and analysed using SHADOW, and again analysed by replaying through the Snort IDS. In this report, Snort and SHADOW are compared and the benefits of deploying an ID System are discussed. Some interesting statistics for the period are presented, as well as a discussion of the analysis of a half-day of activity, which included scans, effects of denial of service attacks, false alarms and misconfigurations. Scripts developed for the analysis process are included.

Résumé

Le système de détection d'intrusion (SDI) SHADOW a été déployé sur le DREnet afin de mieux comprendre les besoins des analystes d'intrusion. En utilisant SHADOW, nous avons d'abord recueilli et analysé tout le trafic de réseau présent à la frontière DREnet-Internet durant une période de 16 jours. Nous avons ensuite effectué une nouvelle analyse en reproduisant le trafic à l'aide du SDI Snort. Ce rapport compare les systèmes Snort et SHADOW et examine les avantages du déploiement d'un système de DI. Il présente aussi certaines statistiques intéressantes pour la période ainsi que l'analyse d'une demi-journée d'activité, y compris les balayages, les effets d'attaques de deni de service, les fausses alertes et les erreurs de configuration. Les scripts développés pour l'analyse sont inclus dans ce rapport.

This page is intentionally left blank.

Executive summary

In this information age, modern military operations are increasingly dependent on information systems. These systems, especially when part of a computer network, could be vulnerable to intrusions, or attempts to compromise their confidentiality, their integrity, and their availability. For this reason, Intrusion Detection (ID) systems are now widely used and are recognized as a key element of a layered computer security model.

One of the goals of the Attack Detection and Analysis (ADA) group of the Information Operations (IO) section at DREO is to understand the needs of intrusion analysts to better serve our clients and to further develop our research program. To this end, we deployed a network-based ID System (IDS) on the Defence Research Establishment network (DREnet) for a period of 16 days from August 16 to September 1 of 2000.

This experiment gave us hands-on experience with ID systems and helped us to be more aware of the issues faced by intrusion analysts. These issues include storage of data, false positive rates and large number of alerts. Also, we found a need for up to date information about the state of the network and for "stateful inspection" of packets to determine whether proper protocols are being followed.

The data collected has already been put to use in training for and acquiring the SANS Institute GIAC Intrusion Analyst Certification for 2 members of the section. There are already plans within the section to use the data in experiments involving data mining and ID visualisation techniques. Potential short projects include the study of methods of replaying the data in a laboratory environment, as well as a study of the performance of UNIX-based operating systems for packet collection.

The data collected could be used in a variety of projects within the IO section. Multi-sensor data fusion of ID systems for the reduction of false positives is a research activity within the ADA group. Techniques for visualisation of intrusion alerts, network discovery and network mapping are also planned research areas within the ADA group.

Lefebvre, J., Treurniet, J. 2001. DREnet Traffic Analysis: Lessons Learned in the Deployment of an IDS on the DREnet. DREO TM 2001-100 Defence Research Establishment Ottawa.

Sommaire

À l'ère actuelle de l'information, les opérations militaires modernes dépendent de plus en plus des systèmes d'information. Ces systèmes, particulièrement ceux faisant partie d'un réseau informatique, peuvent s'avérer vulnérables aux intrusions ou aux tentatives visant à compromettre leur confidentialité, leur intégrité et leur disponibilité. Pour cette raison, les systèmes de détection d'intrusion (DI) sont maintenant largement utilisés et reconnus comme un élément clé d'un modèle de sécurité informatique en couches.

L'un des objectifs du groupe de détection et d'analyse des attaques (DAA) à la section des opérations d'information (OI) du CRDO consiste à comprendre les besoins des analystes d'intrusion afin de mieux servir nos clients et de poursuivre le développement de notre programme de recherche. Dans ce but, nous avons déployé un système de DI (SDI) sur le réseau du Centre de recherches pour la défense (DREnet) durant une période de 16 jours s'étalant du 16 août au 1er septembre 2000.

Ce déploiement nous a permis d'acquérir un 'expérience pratique sur les systèmes de DI, et il nous a aidé à mieux comprendre les problèmes auxquels font face les analystes d'intrusion. Ces problèmes ont trait au stockage des données, au taux de fausses alertes et au grand nombre d'alertes. Nous avons aussi constaté le besoin d'information à jour sur l'état du réseau ainsi que d'une inspection dynamique des paquets visant à déterminer si les protocoles appropriés sont respectés.

Les données recueillies ont déjà servi à la formation de deux membres de la section, qui ont obtenu le certificat d'analyste d'intrusion du GIAC au SANS Institute. La section a par ailleurs établi des plans en vue d'utiliser ces données pour des expériences faisant appel à des techniques d'exploration de données et de visualisation de DI. Nous envisageons aussi de courts projets sur l'étude de méthodes de ré exécution des données dans un laboratoire ainsi que l'examen des performances de systèmes d'exploitation du type UNIX pour la collecte de paquets.

Les données recueillies pourraient s'utiliser dans divers projets de la section des OI. Ces projets inclut la fusion de données provenant de capteurs multiples qu'exécutent les systèmes de DI pour réduire le nombre de fausses alertes. Les techniques de visualisation des alertes en cas d'intrusion, de découverte de réseau et de représentation cartographique de réseau font également partie des domaines de recherche planifiés pour le groupe de DAA.

Lefebvre, J., Treurniet, J. 2001. DREnet Traffic Analysis: Lessons Learned in the Deployment of an IDS on the DREnet. DREO TM 2001-100 Centre de recherches pour la défense Ottawa.

Table of contents

List of figures

List of tables

Acknowledgements

The authors would like to acknowledge Vincent Taylor for his assistance in getting the experiment approved and for helpful discussions, as well as Joe Spagnolo, for initial discussions about technical issues regarding deploying an IDS on the DREnet and for his help in understanding the traffic. Thanks also to Marc Grégoire for his help in creating the traffic plots with Excel charts and for the careful review of this document.

This page intentionally left blank.

1 Introduction

In this information age, modern military operations are increasingly dependent on information systems. These systems, especially when part of a computer network, could be vulnerable to intrusions, or attempts to compromise their confidentiality, their integrity, and their availability. For this reason, Intrusion Detection (ID) systems are now widely used and are recognized as a key element of a layered computer security model.

For large and active networks, the management of Intrusion Detection Systems (IDS) can be overwhelming. There are still many problems to overcome with respect to detection (the rate of false positives and false negatives), resources to deal with the volume of alerts, management and storage of large datasets, financial resources, *etc.* Research has led to the development of a variety of ID systems, both commercial and academic, all of which have positive and negative characteristics. Intrusion detection analysts, such as those at DND's Computer Incident Response Team (CIRT), face the chore of dealing with these problems on a daily basis.

One of the goals of the Attack Detection and Analysis (ADA) group of the Information Operations (IO) section at DREO is to understand the needs of intrusion analysts in order to better serve our clients and to further develop our research program. To this end, we deployed a network-based IDS on the Defence Research Establishment network (DREnet) for a period of 16 days from August 16 to September 1 of 2000. Both inbound and outbound traffic was logged, as well as "black-holed" traffic. This allowed us to see all types of network activity directed against our network, even if this traffic was eventually filtered by the DREnet routers or firewall. These logs provided us with a means to appreciate the difficulties presented to intrusion analysts, and also provided a data set of real traffic for future research and experiments in this area.

In choosing the IDS for the experiment, we required that it be flexible so we could make changes to it, that it be open-source so we could fully understand how the IDS works, and that it be well-supported. Two ID systems that fulfill these requirements are SHADOW [1] and Snort [2]. We chose to deploy the SHADOW IDS for data collection because it is capable of keeping well-organized and complete logs of all traffic. However, since SHADOW lacks alerting capability and does not analyse packet content (without great effort), we replayed the data through the Snort IDS, which is a rule-based IDS that can alert based on content analysis as well as header analysis.

Privacy was an issue in designing this experiment. Although the DREnet is in theory an experimental network, it is also used operationally and therefore much consideration into the privacy of its users was involved. In order to ensure that there was the least intrusion into the employees' privacy, it was decided that for this experiment we would only collect the first 68 bytes of each packet's header (SHADOW's default setting). This guaranteed that a typical TCP/IP header (of length 64 bytes, including Ethernet, IP and TCP headers) would be captured, allowing 4 bytes for IP or TCP options. It was acknowledged that we would often capture 4 bytes of packet content. In the end, these few bytes allowed us to detect a few content-based attacks. If outbound traffic is proxied by the firewall via network address translation (NAT), a layer of anonymity exists for the users.

A proposal detailing the purpose of the experiment (Annexe A) was approved by the appropriate personnel[1], and the DREnet Management Board and DREnet TPOCs were notified of the dates of the logging activity.

This report describes the lessons learned in setting up an IDS and interpreting network traffic. In the next section, we describe the hardware and software required for collecting and analysing the data. The results of the analysis are described in section 3. Here we report on the type of activity we saw between the Internet and DREnet and potential misconfigurations that should be addressed. We discuss the lessons learned, compare the analysis capabilities of Snort and SHADOW and discuss the benefits of using an IDS in section 4.

[1] Peter Lockwood, Vincent Taylor, Marc Lemoine, Robert Strachan and Joe Spagnolo formed the "DREnet Experiment Review Panel".

2 Experimental Setup

A SHADOW IDS was built, secured and subsequently deployed on the DREnet to collect data for a period of two weeks between August 16 and September 1, 2000. After collection, SHADOW filters were tailored to DREnet traffic and refined to produce a manageable set of logs. The data was also replayed through the Snort IDS to produce alerts and to filter for content. This section describes the configuration and procedure used.

2.1 The SHADOW IDS

The SHADOW IDS consists of 2 units: the "sensor", and the "analyser". The sensor is primarily responsible for capturing network packets in binary log files on an hourly basis. The analyser pulls the hourly logs from the sensor and filters the data for suspicious traffic. The filtered data is sorted by source IP address and stored in HTML format, ready to be viewed by an analyst.

2.1.1 Building the SHADOW IDS

Since SHADOW is Unix-based, two PCs running Red Hat Linux 6.2 were used for the sensor and analyser. The suggested hardware specifications for the SHADOW hosts are described in reference [3] and detailed in Table 1. Of course, the specifications vary depending on the size of the network to be monitored. Our hosts were Pentium III 650 MHz PCs with 384 MB of RAM. Unfortunately, we did not have SCSI interface cards or disk drives required for optimum I/O speed. We relied on standard IDE disk drives.

In terms of hardware, the sensor and analyser varied in two ways. Firstly, the sensor required two network interface cards (NICs) whereas the analyser only needed one. The first sensor's NIC was required for the tap (capturing network traffic) while the second was used to network (hence, communicate) with the analyser's NIC. Secondly, disk space on the analyser (28 GB) was double the amount available on the sensor since the analyser was expected to store all of the logs collected over the two-week period whereas a maximum of 5 days of logs would be kept on the sensor at one time.

Table 1: Hardware requirements of the SHADOW sensor and analyser.

SENSOR	ANALYSER
• 400 MHz Pentium PC	• 400 MHz Pentium PC
• 128 MB memory	• 128 MB memory
• 10/100baseT network interface card	• 10/100baseT network interface card
• SCSI interface card	• SCSI interface card
• 9GB SCSI disk drive	• Two 23 GB SCSI drives

The SHADOW software was downloaded for free from the Naval Surface Warfare Center's web site [1]. Specific libraries and drivers required to run SHADOW were obtained from the web sites listed in Table 2. The sensor and analyser use tcpdump and libpcap to both capture and redisplay packets. Tcpslice splits or concatenates tcpdump binary files and is required by the analyser for post-processing. Secure Shell is required for the encrypted communication between the analyser and the sensor. Web server software may be required in an operational environment for access to the data by multiple analysts. Although not necessary in this case, the Apache web server was installed on the analyser to gain experience installing a complete SHADOW package.

Table 2: Software required for the operation of the SHADOW sensor (S) and analyser (A).

tcpdump 3.4	SA	Captures Internet packets and displays header information. Available on the Red Hat 6.2 distribution, or at ftp://ftp.ee.lbl.gov.
libpcap 0.4	SA	How UNIX gets network information from its kernel. A portable framework for low-level network monitoring, libpcap can provide network statistics collection, security monitoring and network debugging. Available on the Red Hat 6.2 distribution, or at ftp://ftp.ee.lbl.gov.
tcpslice 1.1a3	A	A program for extracting portions of packet-trace files generated using tcpdump's -w flag. It can also be used to glue together several such files. Available at ftp://ftp.ee.lbl.gov.
Secure SHell (SSH)	SA	Version 1.2.27 was installed since version 2 is not available free of charge for government use. http://www.ssh.org.
SHADOW	SA	The SHADOW scripts can be found at http://www.nswc.navy.mil/ISSEC/CID/step.tar.gz. Additional documentation is available in the parent directory.
Apache web server	A	The latest release of the Apache web server can be found at http://www.apache.org.

We chose Red Hat 6.2 as the platform for both, as they were more easily secured as outlined in [4]. The sensor and analyser communicated via SSH1. Reference [5] details the compilation and configuration procedure. Applications requiring upgrades for security reasons at the time of the installation were kernel source and headers, Netscape and gpm. These upgrades were obtained from ftp://ftp.cse.buffalo.edu/pub/Linux/redhat/ftp.redhat.com.

2.1.2 Deploying the SHADOW IDS

The SHADOW IDS was placed on a switch outside the DREnet firewall, at the border between the DREnet and the Internet (Figure 1). In particular, the sensor was connected to the Switch Port Analyser (SPAN) port of the switch. The SPAN port was configured to receive a copy of all traffic flowing through all of the other ports of the switch. This way, the sensor could capture every packet that passed or attempted to pass through the Internet-DREnet border.

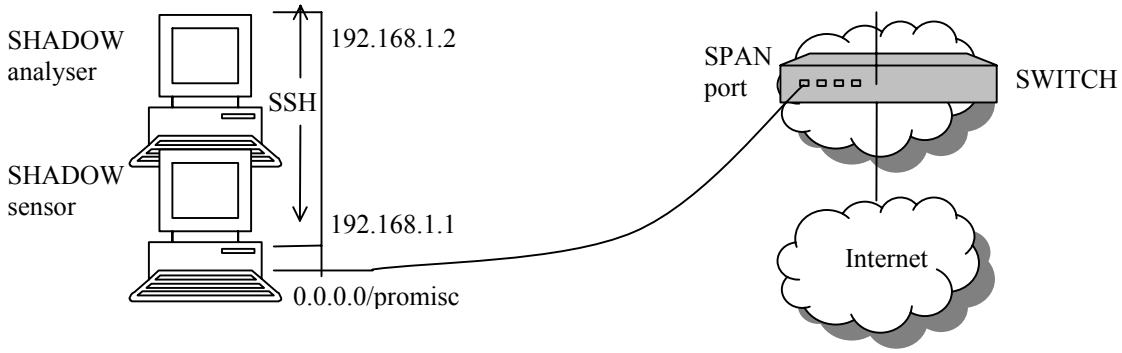

Figure 1. The location of the SHADOW IDS on the DREnet.

The sensor's NIC connected to the SPAN port was set to be in promiscuous mode (via '`ifconfig eth0 promisc`'[2]) and had no IP address assigned to it. By doing so, the NIC could read all packets that passed by it, yet remain inaccessible (no packet can be addressed to it if it has no IP). The second NIC in the sensor was connected to a hub shared with the analyser. These NICs were assigned the non-routable addresses 192.168.1.1 and 192.168.1.2, for the sensor and analyser respectively. By doing this, an island was created for the IDS, into which there was no access except from the analyser or sensor console.

Time synchronicity between the sensor and analyser is essential for proper operation of SHADOW. The sensor used rdate in a cron job to synchronize with the analyser.[3] The analyser pulled the hourly data from the sensor at the end of each hour (note the need for time synchronicity between the two machines) and ran the following scripts on the data:

1. fetchem.pl: grabs the log file from the sensor using scp, runs all the filters in the "filters" subdirectory on the log file and writes the output to an HTML formatted output.
2. sort_and_resolve.pl: sorts the HTML output of fetchem.pl by source IP address and resolves the addresses to names where possible.
3. find_scan.pl: searches the log file for evidence of scans.

These scripts are provided with the SHADOW distribution. Also provided in the SADOW distribution for the analyst's convenience are the following stand-alone scripts:

---

[2] `ifconfig eth0 -promisc` takes the card out of promiscuous mode in Red Hat 6.2.
[3] In operational environments, it is recommended to synchronize with a well-known time server in order to correlate events with other facilities.

1. pat_match.pl: search an hour for a specified pattern, in tcpdump format.
2. one_day_pat.pl: search each log in a given day for a specific pattern, specified in tcpdump format.
3. one_day_script.pl: search each log in a given day for a pattern defined in the specified script, also in tcpdump format (i.e. -F <scriptname>).

2.1.3 Writing SHADOW Filters

The SHADOW software includes a set of standard filters for the analyser and none for the sensor. In theory, the sensor filters should be kept to a minimum (preferably nothing) since filtering requires CPU time and reduces the rate at which the sensor can capture packets. In this experiment, traffic to the border routers was not logged. This was accomplished through the use of a filter on the sensor.

The data collection began at noon on August 16th, 2000. The standard analyser filters had to be tailored to the DREnet. The first step in determining a good set of filters for the DREnet was to determine which hosts are running services that are accessible from the Internet. A script was written in Perl (server-search.pl, Annexe B) to retrieve this information from the data. This was achieved with the following algorithm: if a SYN packet directed at one of our machines elicited a reply with a SYN-ACK packet, the conclusion was that this service is active and is allowed. Table 3 shows the results of our server search. Of course, this only shows the servers that were accessed during the two-week period.

Table 3: DREnet servers, determined by the presence of a completed TCP handshake.

PORT	SERVICE	NUMBER OF SERVERS
21	ftp	14 servers found
22	SSH	2 servers found
23	telnet	6 servers found
25	SMTP	16 servers found
53	DNS	4 servers found
80	www	16 servers found
102	Iso-tsap	2 servers found
110	Pop3	10 servers found
113	Auth	1 server found (firewall)
119	news	1 server found
143	Imap2	1 server found
389	Ldap	2 servers found
709	entrustmanager	2 servers found

Using this data, we were able to write filters for the analyser to log any traffic that is not explicitly allowed. In particular, the filtering rules designed for DREnet servers were different from those written for DREnet desktops. For servers accessible from the Internet, anything other than the allowed connections was reported. For the remainder of the machines on the DREnet, no connections were allowed from the outside. The filters were also written such that traffic that did not follow the rules of the TCP/IP protocol involved would be displayed to the analyst. The filters were designed to report on any anomalous behaviour.

The SHADOW filters also included generic filters [6] written by Vicki Irwin (one of the developers of SHADOW), which catch anomalies such as ICMP destination unreachable, TCP with anomalous flags and fragmentation.

Another filter had to be applied before all others to reduce traffic. DND's General Purpose network (GPnet) and the DREnet use each other as backup routes to the Internet via their respective Internet Service Providers. GPnet sometimes routes its traffic through the DREnet border to get to the Internet, thereby passing through the SHADOW IDS and adding a significant amount of traffic to our logs. Any traffic between GPnet and any IP not in the DREnet was filtered out.

All analyser filters were applied after the IDS was disconnected from the DREnet. The Perl script from the SHADOW distribution was modified to also indicate which filter triggered the output of the trace.

2.1.4 Storage

For the purposes of this experiment, an Ecrix VXA tape drive was used to store all of the logs on a 30 GB tape.

2.2 The Snort IDS

Snort is a rule-based IDS. It comes with a default set of rules that can be modified to suit the needs of the user. If a packet matches a rule in the ruleset, an alert is sent to `/var/log/snort/alerts` (by default). The format of the alert can be "fast" (one line per alert) or "full" (complete packet information). Subdirectories are created by default under `/var/log/snort` which are named after the IP from which a packet originated, where full headers for the packet are recorded. There are "preprocessors" in Snort, the most common of which are the portscan preprocessor and the http_decode preprocessor. The http_decode preprocessor decodes unicode sometimes used in URL requests, and if this reveals nefarious activity, an alert is logged to the `alerts` file. A portscan is defined in the portscan preprocessor as TCP connection attempts to more than *P* ports in *T* seconds or UDP packets sent to more than *P* ports in *T* seconds. Ports can be spread across any number of destination IP addresses, and may all be the same port if spread across multiple IPs. The alert sent to the `alerts` file simply states that a portscan occurred, while all packets involved in the scan are recorded in `/var/log/snort/portscan.log`.

Snort 1.7beta7 was obtained from a temporary ftp location on Martin Roesch's personal web page advertised on the snort-users mailing list. The beta version was used because of its ability to define the HOME_NET variable as a list of IP ranges (required for the DREnet). It was installed on a host running FreeBSD 3.4. The raw hourly log files were transferred from the SHADOW sensor via scp (secure copy), and each log file had to be processed by a program called "editcap" (available with the Ethereal distribution [7]), which translates Red Hat 6.2 tcpdump format to the standard tcpdump format. Each of these logs was then run through Snort with the default ruleset using the following command:

```
gunzip -c <file.gz> | snort -F gpnet.filter -dv -r - -A fast  -c snortfull.conf -N
```

The GPnet filter mentioned previously (in Section 2.1.3) was applied to each dataset. The alert method was “fast”, so that one line was written for each alert, and the default ruleset `snortfull.conf` was used, with minor changes to reflect our network. The -N option tells Snort not to create the subdirectories based on IP, since we lacked the disk space required for this option.

Due to the volume of alerts, each daily Snort alert file was split such that any ICMP alerts were diverted to a separate file (using 'egrep -v (ICMP|PING)') and any portscan alerts were diverted to another (using 'grep -v portscan'). The remaining Snort alerts were parsed using the freely-available Perl script `snort_sort.pl` [2], which groups identical events and can output the result to HTML format.

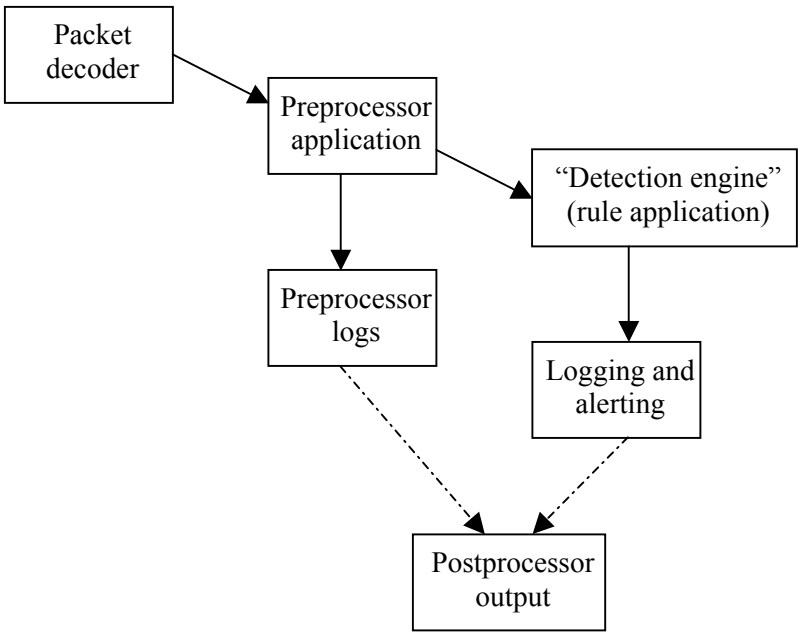

Figure 2. The Snort process. Packets first pass through the packet decoder, are run through the selected preprocessor plugins, then the rules are applied to generate alerts. Optional postprocessing may be applied to the logs.

3 Results

In this section, we present the results obtained from the analysis of the collected data, including the performance of the SHADOW IDS, statistics relating to the type of activity we saw between the Internet and DREnet, and potential misconfigurations that should be addressed.

3.1 Volume of traffic

For each weekday, approximately 1.2 GB of unfiltered packets were logged and stored (compressed using gzip). At the rate of 68 bytes per packet, this translates to roughly $18x10^6$ packets per day. At peak usage times (noon EST), we logged as many as 600 packets/sec. Given this volume of traffic, we were immediately faced with some of the problems that security analysts may have regarding the storage of large volumes of data, the logging rates and performance of IDSs, and the reduction of data for effective traffic analysis.

3.1.1 Performance of SHADOW

We cannot state with certainty whether the SHADOW sensor dropped packets or not. Red Hat 6.2 with the 2.2.16 kernel does not have the capability to allow us to determine if packets are being dropped during the logging process. However, during our analysis, we did not see any evidence of dropped packets caused by the logging speed of our SHADOW sensor. Hence, the hardware (including IDE rather than a SCSI interface card) and operating system of our sensor seems to be adequate for the volume of traffic at the Internet-DREnet gateway.

While analysing the SHADOW logs, we found that 2 seconds worth of packets were lost at the beginning of each hour. This was first discovered while running a script, unsolicited.pl (Annexe B) that searches the data for incomplete TCP handshakes. We found instances where a TCP conversation was clearly established but the TCP handshake was never initiated. Essentially, we were missing pieces of the beginning of a TCP conversation. These instances were only occurring at the beginning of each hour. Upon closer examination of the logs, we noticed a 2 second gap between the last packet logged in an hourly file and the first packet logged in the next hours' file. Hence, it would appear that some process is happening at the beginning of each hour that causes the sensor to drop 2 seconds of worth of packets.

3.1.2 SHADOW filters

The first set of SHADOW filters produced an average of 25 pages of traces per hour. There were many false positives caused by oversights when writing filters to alert on anomalous protocol behaviour. Also, ICMP traffic accounted for more than half of the traces. Hence, the filters were further refined. This reduced the traces to an average of 9 pages per hour, which is much more tolerable for an analyst.

3.2 Analysis of DREnet traffic

A complete analysis was performed on 12 hours of traffic on August 16th, from 12:00 to 23:00 using both Snort and SHADOW. The analysis process required the design and refinement of SHADOW filters, and investigation of alerts to eliminate false positives. Once the filter refinements were in place, the false positives became easier to identify.

The majority of the traces involved system scanning and false positives caused by misconfigurations of hosts or network devices. The DREnet was not subjected to a denial of service attack, however we did log traffic that appears to be the result of a denial of service on an entire external subnet. The content-based analysis (Snort) alerted on system penetration attempts such as attempts to retrieve telnet password files or attempts to access the root directory on the DREnet web server. However, none of the attempts were successful. There was just as much suspicious activity occurring during the daytime as there was at night.

Points of interest originating from the analysis of August 16th include:

1. System scanning and internal threats: We detected a series of scans originating from within the DREnet and targeting an external DND host. On further investigation, we concluded that this series of scans was most likely part of a planned trial, as first contact was made by the "victim". Still, this emphasizes the need to monitor for internal threats such as compromised hosts.
2. System scanning and external threats: The DREnet's IP address space was often used in decoy scans where the attacker attempts to hide his/her identity.
3. Misconfigured server: We found a DREnet server responding to TCP connection attempts with resets if the source was not one of our own. Responding in this manner could be used in a denial of service attack.
4. Misconfigured host: Some DREnet users have misconfigured computers (or laptops) at home that try to access DNS servers that are not accessible from the Internet.
5. Insecure practice: Telnet was used by a DREnet user to access his/her work computer from home. Telnet passwords are passed in the clear and could be sniffed. SSH is a more secure alternative.
6. Performance issue: We picked up a number of source quenches coming from a router at the border of one of our subnetworks, which mostly occur at around noon. This implies that there is too much traffic for this router to handle during peak usage. This is a performance issue that may warrant investigation.

For the remainder of the two-week period, some statistics were gleaned from the Snort alert logs using the Perl script snort_stat.pl [2]. Table C-1 of Annexe C contains the summary. The majority are scans for both header and content-based vulnerabilities. The statistics are overwhelmed by the number of alerts generated by scans. The scans, reported in Table C-2 of Annexe C, were obtained by tallying the types of scans from each day's portscan.log log file. A scan was defined in this case as 4 hits from the same source in 3 seconds. Neither of these statistics is necessarily accurate in the sense that it could be tallying false positives. However, they provide an estimate of the amount and types of activity that are present on the DREnet.

The script traffic-stats.pl (Annexe B) was written to attempt to determine what kind of traffic is most prevalent on the DREnet. Figure 3 shows the breakdown of the protocols used during

the period. TCP is the most common with UDP in a distant second. Figure 4 shows the breakdown of TCP traffic, most of which is comprised of web traffic. FTP, mail and news services make up the bulk of the remainder. Figure 5 shows the UDP traffic for the period. The spike between August 24 and August 25 is due to a videoconference between CRC and some universities. ICMP traffic is shown in Figure 6, with an echo request spike on the 26th corresponding to a large-scale ping sweep from the @HOME network.

As a post-processing measure, a Perl script was written to locate scans or related problems which require some stateful awareness, such as SYN-ACK scans, which can be identified by the lack of an initiating SYN. The script is attached in Annexe B, unsolicited.pl. The majority of the anomalies found with this method were third party effects of scans or misconfigured web servers.

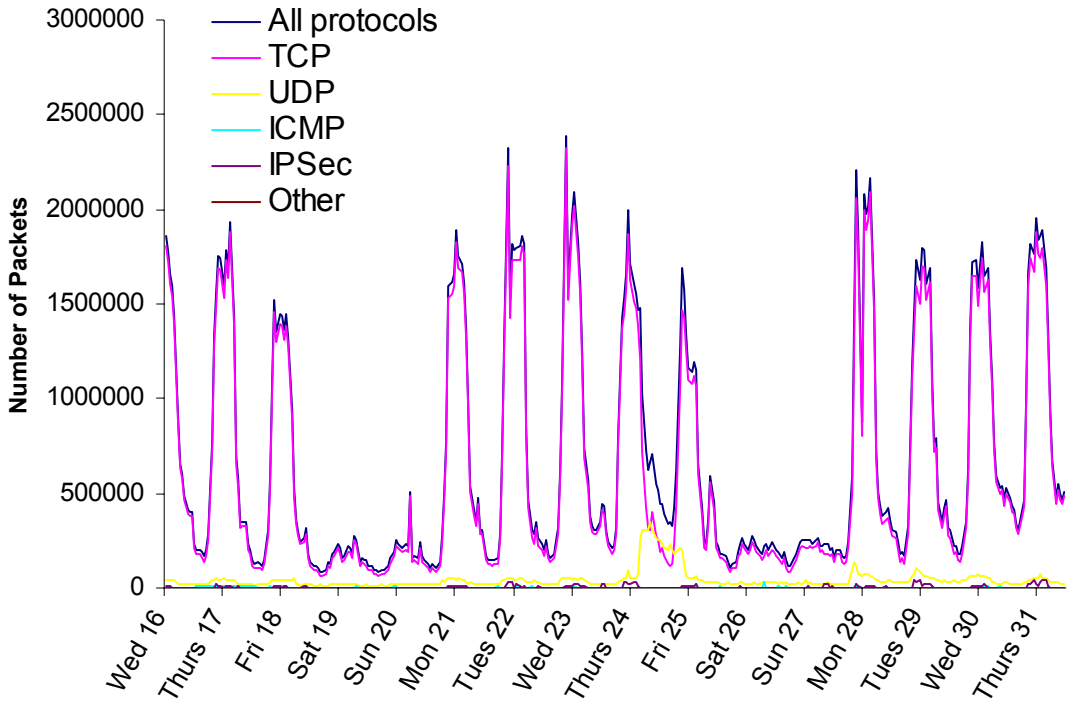

Figure 3. Breakdown of protocols for the 2-week period.

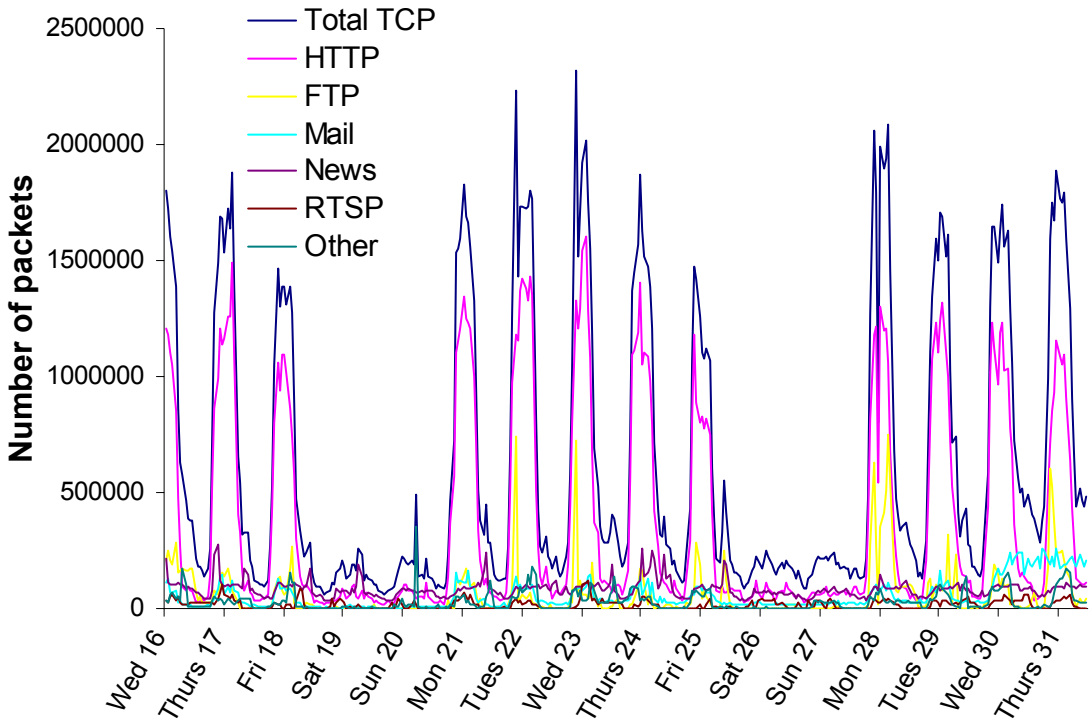

Figure 4. Breakdown of TCP traffic for the 2-week period.

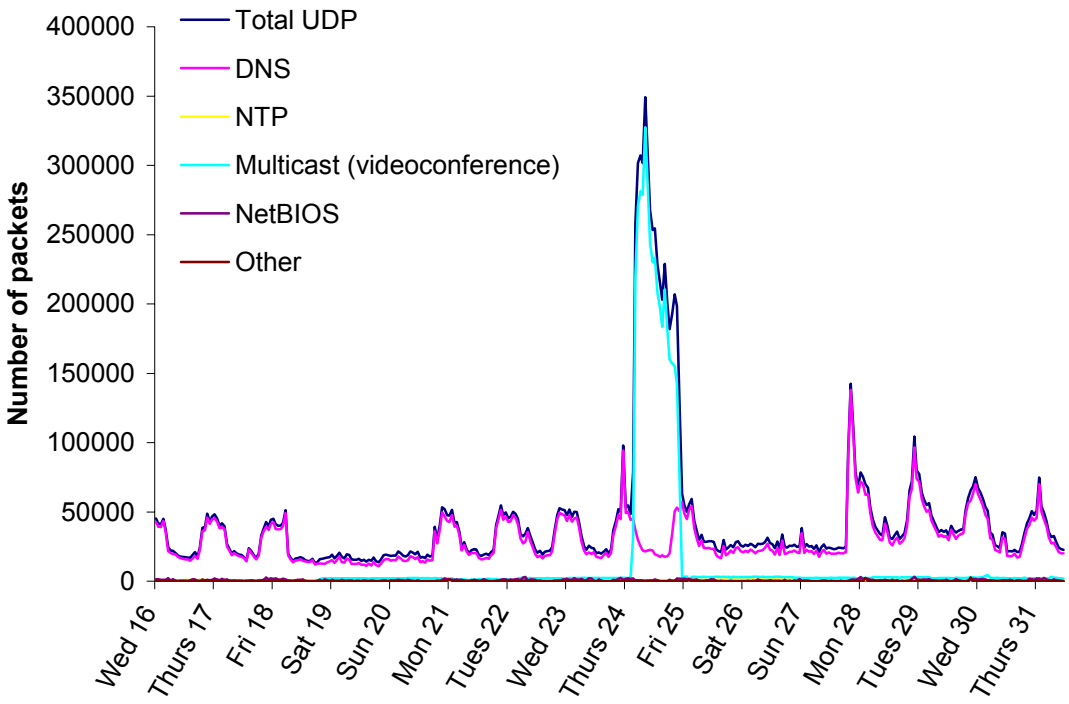

Figure 5. Breakdown of UDP traffic for the 2-week period.

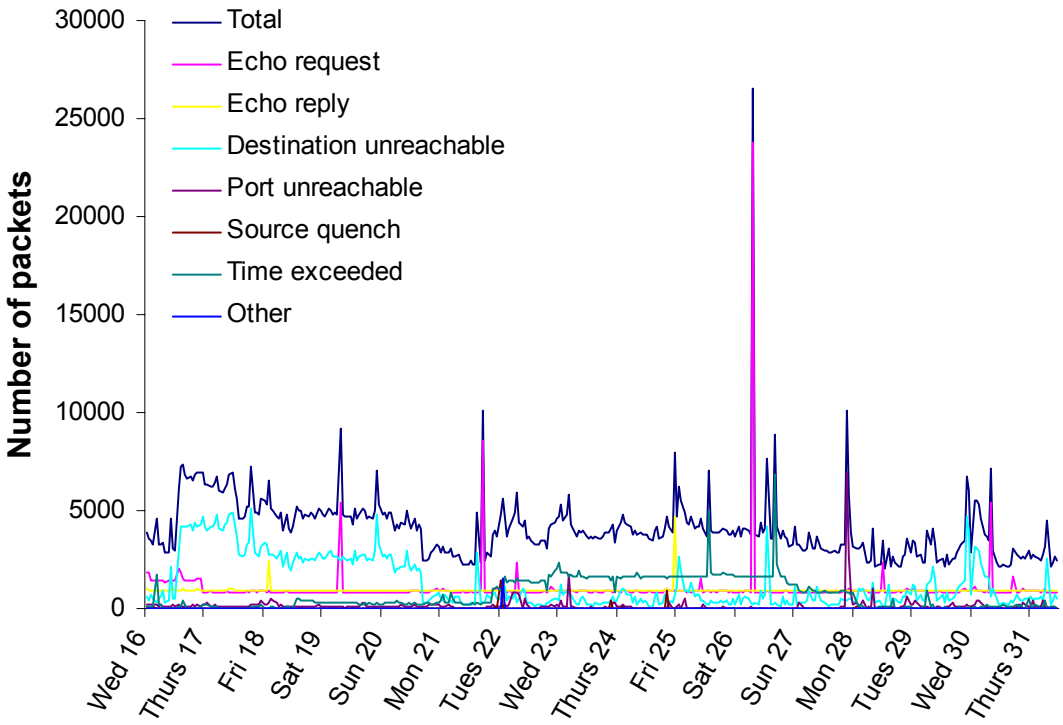

Figure 6. Breakdown of ICMP traffic for the 2-week period.

4 Discussion

In this section we discuss the lessons learned as a result of this study and make some comments on the ID systems used. The benefits and drawbacks of the use of an IDS are discussed in a final note.

4.1 Lessons learned

4.1.1 False positive reduction is a significant issue in an operational environment.

There is a trade-off in designing filters: if one designs filters to catch every possible attack, the logs get filled with false positives. However, if one refines filters too much or removes rules, there is a risk that some attacks will be missed (false negatives). It is usually left to the analyst to decide whether an alert or particular type of traffic is of interest.

The placement of the IDS makes a big difference in the rate of false positives. In our case, we placed the IDS in a position such that it would log all packets destined for our network range, even those eventually blocked by the border routers or the firewalls. An operational environment would likely not take such an extreme approach and would rather place an IDS inside the border routers and outside the firewall, or even just inside of the firewall. For example, in our Snort analysis, over 55% of the alerts were of the type ICMP. Most of these packets were dropped at the border and would not have been logged by an IDS placed inside the DREnet's border defences.

Ideally, one would place an IDS inside the firewall and one outside the firewall, giving the analyst only alerts which made it past the firewall, and still providing the resources for *a posteriori* trace analysis. For example, a Snort IDS could be placed inside the firewall and a SHADOW IDS outside the firewall. However, if the firewall had dynamic network address translation, correlation of traffic would be difficult. An alternative in this case would be to put a SHADOW IDS and a Snort IDS at each point, before and after the perimeter defences. This would optimize the alerting and trace analysis capabilities.

4.1.2 Network Map

One of the difficulties we encountered while analysing the data was not having a current picture of the network, *e.g.* what services were allowed to run and what the IP addresses of those servers were. We were provided with the DNS zone transfer file and the border router access control list, however we found this data to be incomplete when we ran server_search.pl (Annexe B) to identify servers and services on the DREnet. Some servers were running services of which we were not aware. We had no way of knowing if a new computer had been added. The procedure of looking up an IP on the DREnet was time-consuming. At times, we could not even resolve some IPs. A network map would have made the process more efficient.

4.1.3 Red Hat 6.2 tcpdump modification

Red Hat provides a modified version of tcpdump with its 6.2 release, which stores the binary logs and outputs text in a non-standard format. A lack of awareness of this modification led to the use of SHADOW without upgrading to the latest tcpdump [8]. This caused some inconvenience in that the SHADOW scripts had to be modified to deal with Red Hat's non-standard output format and the logs had to be filtered through editcap [7] before being processed with Snort.

4.1.4 Visualisation

The 2D visualisation of the traffic statistics allowed us to easily spot anomalies such as the videoconference and the ping sweep. This made us aware of the fact that visualisation methods may be a great aid to intrusion analysts if done properly. Projects investigating ways to incorporate visualisation techniques into ID systems, such as IRONMAN, are currently underway in the IO section.

4.1.5 Bi-directional traffic analysis

It is just as important to log traffic coming out of the network as traffic coming in. On August 16, we discovered a scan on a machine at a CF base coming from HQ. The conclusion is that this was a planned attack, since the conversation appears to have been initiated by the victim, however if this was real, one would like to be aware of it. Mainly, it is to watch for instances where hosts has been compromised. In some cases, it could be used to conduct further attacks.

4.1.6 Snort's default ruleset

Snort's content-based alerts caught some activity that would otherwise have gone unnoticed, although there were too many false positives with the default ruleset. Rule refinement is required.

4.1.7 Snort does not show ICMP codes in the "fast" alert format

Destination unreachables have a variety of codes, which Snort does not show in its "fast" alerts. In order to see the code, one would have to store a huge amount of data (using "full" alerts). Tcpdump, however, shows both the code and the message embedded within the ICMP packet by default. Destination unreachable alerts were separated and analysed by hand using tcpdump. Snort rules have been developed to deal with this, but were not implemented in the default ruleset as of Snort 1.7beta8.

4.1.8 Content-based alerts and privacy

At the beginning of this trial we decided to limit the amount of data collected to 68 bytes per packet, due to a concern for privacy of DREnet users. This amount was found to be enough to provide a large amount of content-based alerts, while remaining relatively unintrusive. The sacrifice for these alerts is small, since they involved attempts to compromise the security of the DREnet. If an analysis of attack methods is required, the limit to the amount of data collected per packet should be increased [9].

4.1.9 Packet loss

We suspect that SHADOW's loss of 2 seconds worth of packets (section 3.1.1) is due to the fact that SHADOW 1.6 runs a script to stop the logging process, followed by a script to start the next. These start and stop scripts should be overlapped by at least 3 seconds so that packets are not dropped.

There is some controversy over which Unix based OS should be used for the SHADOW sensor. Tcpdump 3.4 running on a 233 MHz Pentium II with Linux 6.1 was reported to drop packets, whereas the same version of tcpdump did not drop packets on a 120 MHz "586" running FreeBSD 3.1 [10]. However, we have a conflicting statement [9] that in another experiment, although FreeBSD did not report dropping packets, it was indeed dropping packets. As well, Red Hat 7.0 includes packet loss statistics in its kernel, however recent discussions on the Snort user's mailing list [11] state that while showing no packets are dropped, these statistics are bogus. This is a topic for future investigation.

4.1.10 Snort vs. SHADOW

Snort and SHADOW are two very different intrusion detection systems, and like most, each has good and bad qualities. Snort presents the analyst with alerts based on rules that can contain content analysis, while SHADOW presents the analyst with a tcpdump format page of anomalies. While SHADOW's tcpdump filters make it difficult to log packets based on content (since the content would have to be expressed in hex), Snort's rules make it difficult to alert on some header information such as fragmentation (although it does have a processor to reassemble fragments) and comparison of source and destination IPs.

One quality that they share is the ability/need to tune the IDS to provide the best quality of data for analysis. This is a good quality if you are a very hands-on type of analyst, and a bad quality if you prefer to "plug-n-play". The refinement of rules and filters is a time-consuming procedure, but is necessary to keep false positives to a manageable level.

Both Snort and SHADOW, being open-source products, have freely available support from users' community mailing lists. The Snort mailing list is far more active; hence Snort is continually improving due to the suggestions from the community. The authors reply to questions and comments promptly, and suggestions are often implemented the next day and are available through CVS.

SHADOW stores data in binary tcpdump format and provides post-processing scripts to search for patterns. Snort has plugins for SQL database storage of alerts, which may be useful in archiving and mining.

4.2 Weighing the Benefits and Drawbacks

It has become widely accepted that Intrusion Detection Systems are a valid and important component in a network's security architecture. The arguments against placing an IDS on one's network usually fall into the following categories:

1. Privacy issues (*e.g.* capture of passwords)

2. Resource issues (*e.g.* budget, including staffing, backup devices/tapes, PCs, software/yearly license, HR/training)

We have made a rough estimate of the costs involved in operating an IDS, shown in Table 4.

Table 4: Setup and operating costs for a hypothetical ID system.

PART	ESTIMATED COST (CDN$)
SANS Institute training	$3500.00
3 PCs (2 SHADOW, 1 Snort)	$6000.00
Backup device	$1000.00
Maintenance	1 PY
TOTAL COST	$10500.00 + 1 PY

These arguments must be weighed against the risks involved in leaving out this layer of security. ID systems provide the security administrator with:

1. Layers of security. Methods for firewall evasion are always being worked on, resulting in the need for secondary measures to react to situations as they arise. For example, a SYN-FIN scan cannot be blocked by most firewalls.
2. Advance warning of network scanning and malicious activity. This provides the security administrator with the ability to patch vulnerabilities before they are exploited.
3. Community activity. IDS data provides the opportunity to interact with organisations like SANS [12], which collect and share statistics on intrusions. This allows the community as a whole to better understand threats and to reveal new attack types for faster response.
4. Logs for trace analysis and elimination of false positives, which complement firewall logs rather nicely.
5. The ability to identify network problems or misconfigurations. For example, both users and administrators make mistakes in configuring and using systems. A misconfiguration could leave a vulnerability open or allow a network to be used as a pawn in a denial of service attack.

5 Conclusion

This experiment gave us hands-on experience with ID systems and helped us to be more aware of the issues faced by intrusion analysts. These issues include storage of data, false positive rates and number of alerts. Also, we found a need for up to date information of the state of the network and for "stateful inspection" of packets to determine whether proper protocols are being followed.

The data collected in this experiment will be useful in other research. For example, the data collected has already been put to use in training for and acquiring the SANS Institute GIAC Intrusion Analyst Certification [13] for 2 members of the section. There are already plans within the section to use the data in experiments involving data mining and ID visualisation techniques. Potential short projects include the study of methods of replaying the data in a lab environment, as well as a study of the performance of UNIX-based operating systems for packet collection.

Multi-sensor data fusion of ID systems for the reduction of false positives is a research activity within the ADA group. Network discovery and visual network mapping is a planned research area within the group, which will hopefully prove to be a useful and time-saving tool for the analyst.

Although it was not one of our goals to make recommendations regarding DREnet security, we found that the DREnet appears to be fairly well-protected by the firewall and border routers, in the sense that we did not see any successful exploits in the duration of the experiment. However, the content-based alerts produced by Snort did show attempts to compromise DREnet servers. For example, there were attempts to gain root access on machines available from the Internet, such as web servers and ftp servers (Annexe C, Table C-1). If all software used on these core machines is not updated quickly and regularly, one of these attempts might work in the future. A threat and risk assessment would aid in the decision of whether an IDS is an essential element for the DREnet.

6 References

1. Untitled Document (The SHADOW IDS home page). (Online) Naval Surface Warfare Center - Dahlgren Lab. http://www.nswc.navy.mil/ISSEC/CID/index.html (March 8, 2001).

2. Jim Forster. Snort- The Open Source Network Intrusion Detection System. (Online) Martin Roesch. http://www.snort.org (March 8, 2001).

3. J. Novak (2000). *SANS 2000 Course Book – Intrusion Analysis – Shadow Style*, Orlando, FL: The SANS Institute.

4. L. Brotzman (1999). *SANS 2000 Course Book - Linux Security Step-by-Step*, Orlando, FL: The SANS Institute.

5. S. Acheson (1999). *SANS 2000 Course Book – SSH: Introduction through Implementation*, Orlando, FL: The SANS Institute.

6. Vicki Irwin. Dec 21, 1999. Index of pub/virwin. (Online) ftp://ftp-eng.cisco.com/pub/virwin (September 2000).

7. G. Combs. The Ethereal Network Analyzer. (Online) http://www.ethereal.com (March 8, 2001).

8. J. Shaw. February 6, 2001. TCPDUMP Public Repository. (Online) http://www.tcpdump.org (March 8, 2001).

9. Guy Bruneau, DND CIRT (personal communication).

10. Milan Kuchta, Systolics Inc., Ottawa, ON (personal communication).

11. M. Johnson. April 5, 2001. [Snort-users] Linux packet loss statistics? (Online) http://www.geocrawler.com/mail/msg.php3?msg_id=5493644&list=4890 (June 4, 2001).

12. Jed Pickel. 2001. Global Intrusion Detection (Online) http://www.incident.org (June 21, 2001).

13. SANS GIAC Training and Certification Program (Online) The SANS Institute http://www.sans.org/giactc.htm (June 21, 2001).

7 Annexes

7.1 Annexe A: Proposal

Deploying a SHADOW IDS on the DREnet

August 1, 2000
(updated Aug 15, 2000)

Julie Lefebvre and Joanne Treurniet
Information Operations Section, Defence Research Establishment
Ottawa, 3701 Carling Avenue, Ottawa, ON K1A 0Z4

1. Purpose

Our immediate goal is to deploy a network intrusion detection system (IDS) on the DREnet to collect sets of data required for research activities of the Information Operations Section. In particular, we would like to deploy a SHADOW IDS to log network traffic (in-bound and out-bound) between the DREnet border router and the Internet for a period of three weeks. This would provide us with logs of the right type of network traffic (most importantly, unfiltered Internet activity directed to the DREnet) required for Information Operations (IO) defence scientists working on projects such as GIAC Certified Intrusion Analyst certification and IRONMAN. The rationale for using the DREnet rather than an IO Lab internal network is that the DREnet is a live, unclassified network under DRDC control that is not too large to analyse yet not too small as to be trivial. Intrusion attempts in this environment will be representative of intrusion attempts in the Internet at large.

The network logs collected will consist of packet headers and, in some cases, the first few bytes of the packet payload. This latter is partly an artifact of the SHADOW IDS operation and partly due to the fact that certain attack signatures occur in these bytes.

It is not our intention to monitor the Internet/DREnet traffic for content. We simply want an extended sample of "live" traffic to be used in ID research experiments.

2. Deployment of SHADOW IDS

To access network traffic between the DREnet and the Internet, we would like to build a demilitarized zone (DMZ) just outside the DREnet border router (Fig. 1). This involves adding a non-filtering router (Internet router) on the segment between the DREnet border router and the Internet (Teleglobe). While this additional router is effectively transparent (*i.e.* does not affect network traffic), it provides a segment from which we can tap in, with a hub, a SHADOW IDS and monitor network traffic exchanged between the DREnet border router and the Internet. The DREnet border router remains the first-line defence against outside attacks.

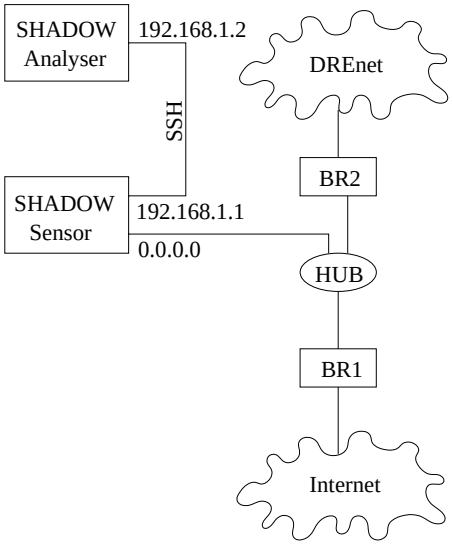

Figure 1: The proposed DMZ and the location of the SHADOW IDS on the DREnet. The SHADOW sensor and analyser communicate via OpenSSH only.

We chose the SHADOW IDS because it is a free, open-source, and flexible system that provides detailed logs of network traffic. SHADOW software is developed and supported by the Naval Surface Warfare Center, Dahlgren VA, USA.

The SHADOW IDS consists of a sensor (to collect network traffic) and analyser (to analyse logs and alert on suspicious activity). The SHADOW IDS sensor logs traffic (packet headers and, in some cases, the first few bytes of the packet payload) and stores it on an hourly basis. These logs are pulled by the analyser for analysis. The SHADOW IDS sensor host has two NIC cards. The first NIC is to connect to the hub (tap) and is set in promiscuous mode (IP 0.0.0.0) to avoid detection. The second NIC is used to network the SHADOW IDS sensor with the analyser. Both sensor and analysis hosts will be used exclusively for SHADOW. Neither of these hosts will be connected to the internal DREnet. This way, the SHADOW IDS system, which isn't protected by the DREnet routers and firewalls, sits by itself on the network. Any compromise to the SHADOW IDS sensor and/or analyser will not affect the DREnet.

Both SHADOW IDS hosts have been secured according to guidelines specified in Joanne's document *Securing Your Linux Desktop at DREO* (draft available on request) and the SANS guide *Intrusion Detection - Shadow Style*. The only daemon running on the SHADOW IDS sensor is OpenSSH and it only allows access to the analyser (so they can securely communicate with each other). The only service running on the SHADOW IDS analyser is the Apache web server, which is used to provide a web-style interface to view the logs. Access is limited to console logins (or via secure shell on the sensor). We have ensured that neither machine uses DNS. The Pentasafe 10-point Checkup tool will be used to perform a vulnerability assessment on both hosts.

3. Material requirements

The SHADOW IDS hosts will ideally be located in the 2nd floor IO Lab at DREO, which would entail the installation of a fibre optic cable from the tap location to the IO Lab. If this is not feasible in the short term, the sensor and analyser can be physically located near the routers.

The raw data collected by the sensor will need to be backed up. We will require a storage device (preferably a removable CD-RW, for future portability and for the larger storage size of a CD) in order to back up data and transfer it to other machines. Although the collected data itself is unclassified, the logs will be protected in a locked storage cabinet when not in use for IO research.

4. Timeline

We hope to finish the data collection by September 1, 2000. We will require a few days to fine-tune the IDS (*i.e.* adjust the filters, especially the sensor to reduce the volume of data) and to learn how to deal with the volume of data. Approximately two weeks of data collection will follow. Table 1 shows the plan.

Table 1: Timeline of the project.

Task	Date
Proposal submitted	July 18, 2000
Proposal revised	August 1, 2000
Proposal approval	August 1, 2000
Material acquisition and installation	August 8 – August 11, 2000
Test period	August 14 – August 16, 2000
Data collection	Noon August 16 – noon September 1, 2000

7.2 Annexe B: Analysis Scripts

7.2.1 Server-search.pl

```
#!/usr/bin/perl
#
# server-search.pl - Joanne Treurniet
#
# script to analyse a tcpdump file for a SYN followed by a SYN-ACK response
# by pairing requests and responses, used to identify what is behaving like
# a server on the DREnet.
#
# CYCLES THROUGH ALL HOURS OF ALL DAYS, sorts output by IP, and
# only check the 1-2 minutes following a SYN.
#
$HOME = "/home/jtreurniet";
$SITE = "GENERIC";
$ENV{PATH} = "/bin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin";
$ANALYZER_DIR = "$HOME/DREnetAnalysis/Raw";

# START DAY LOOP HERE
@days = qw(26);
for $day (@days) {
print STDERR "Starting day $day...\n";

$subdir = "Aug".$day;

# START HOUR LOOP HERE
@hours = qw(17);
for $hour (@hours) {
print STDERR "Starting hour $hour...\n";

$zipped_file="$ANALYZER_DIR/$subdir/tcp.200008$day$hour.gz";

chdir("$ANALYZER_DIR/$subdir");

$cmd1 = "gunzip -c $zipped_file" . " | "
    . "tcpdump -S -n  -r - 'tcp and (((dst net 131.136 or 131.137 or 131.132 or
131.133 or 131.134 or 131.135) and tcp[2:2]<1024 and tcp[13]&0x3f=0x02) or ((src net
131.136 or 131.137 or 131.132 or 131.133 or 131.134 or 131.135) and tcp[0:2]<1024 and
tcp[13]&0x3f=0x12))'";

open(RAWFILE,"$cmd1|");
print STDERR "Reading SYNs and SYN-ACKs...\n";
print STDERR "$cmd1\n";
$is = 0;
while(<RAWFILE>)
{
    next if /gre-proto/;
    next if /trunc/;
    if ($_ ne $arr[$is-1]) {
        $arr[$is] = $_;
        $is++;
    }
}
close(RAWFILE);

print STDERR "Processing $is elements ...\n";
for ($i=0;$i<$is;$i++) {
    # look for handshakes.  The SA must follow the S time.
    $match = -1;
```

```
    $line = $arr[$i];
    @fields = split(/\s+/, $line);
    if ($fields[5] eq "S" and $fields[7] eq "win") {
        # It's a SYN, search for a corresponding SYN-ACK
        @time = split(/\:/,$fields[0]);
        $min1 = $time[1];
        @srcoctets = split(/\./,$fields[2]);
        $csrcip = "$srcoctets[0].$srcoctets[1].$srcoctets[2].$srcoctets[3]";
        @dstoctets = split(/\./,$fields[4]);
        $cdstip = "$dstoctets[0].$dstoctets[1].$dstoctets[2].$dstoctets[3]";
        $csrcport = $srcoctets[4];
        $cdstport = $dstoctets[4];
        $cdstport =~ tr/://d;
        for ($j=$i+1;$j<$is;$j++) {
            $line = $arr[$j];
            @fields = split(/\s+/, $line);
            @time = split(/\:/,$fields[0]);
            $min2 = $time[1];
            # Only continue the search for 1-2 minutes.
            if (($min2-$min1)>2) {
                last;
            } else {
                @srcoctets = split(/\./,$fields[2]);
                $ssrcip = "$srcoctets[0].$srcoctets[1].$srcoctets[2].$srcoctets[3]";
                @dstoctets = split(/\./,$fields[4]);
                $sdstip = "$dstoctets[0].$dstoctets[1].$dstoctets[2].$dstoctets[3]";
                $ssrcport = $srcoctets[4];
                $sdstport = $dstoctets[4];
                $sdstport =~ tr/://d;
                # if the client and server form a pair,
                if (($csrcip eq $sdstip) and
                    ($cdstip eq $ssrcip) and
                    ($csrcport eq $sdstport) and
                    ($cdstport eq $ssrcport)) {
                    # We have a match.
                    $match = $j;
                    last;
                }
            }
        }
    } else {
        $match = -2;
    }
    if ($match eq -2) {
#       print "Skipped SYN-ACK $i $arr[$i]\n";
    } elsif ($match eq -1) {
#       print "No match for $i $arr[$i]\n";
    } else {
        # There was a match - add it to arr2
        $size=$#{$arr2[$cdstport]};
        $arr2[$cdstport][$size+1]=$cdstip;
    }
}

# END HOUR LOOP HERE
}

$size=$#arr2 + 1;
# Make a hash out of arr2 to obtain the unique elements, and output.
for ($i=0;$i<$size;$i++) {
    %seen=();
    $size2=$#{$arr2[$i]}+1;
    if ($size2 gt 0) {
        for ($j=0;$j<$size2;$j++) {
            $seen{$arr2[$i][$j]}=1;
        }
        @uniqueIP = keys %seen;
```

```
        @sortedIP = sort by_IP @uniqueIP;
        # Nice formatted output: 4 IPs per line
        $size3=$#sortedIP+1;
        if ($size3 > 4) {
            if ($size3 % 4 == 0) {
                $nrow=$size3/4;
            } else {
                $nrow=int($size3/4)+1;
            }
            for ($k=0; $k<$nrow; $k++) {
                print STDERR "$i $sortedIP[$k*4] $sortedIP[$k*4+1] $sortedIP[$k*4+2]
$sortedIP[$k*4+3]\n";
            }
        } else {
            print STDERR "$i @sortedIP\n";
        }
    }
}

# END DAY LOOP HERE
}

# FINAL OUTPUT TO FILE:
$size=$#arr2 + 1;
# Make a hash out of arr2 to obtain the unique elements, and output.
for ($i=0;$i<$size;$i++) {
    %seen=();
    $size2=$#{$arr2[$i]}+1;
    if ($size2 gt 0) {
        for ($j=0;$j<$size2;$j++) {
            $seen{$arr2[$i][$j]}=1;
        }
        @uniqueIP = keys %seen;
        @sortedIP = sort by_IP @uniqueIP;
        # Nice formatted output: 4 IPs per line
        $size3=$#sortedIP+1;
        if ($size3 > 4) {
            if ($size3 % 4 == 0) {
                $nrow=$size3/4;
            } else {
                $nrow=int($size3/4)+1;
            }
            for ($k=0; $k<$nrow; $k++) {
                print "$i $sortedIP[$k*4] $sortedIP[$k*4+1] $sortedIP[$k*4+2]
$sortedIP[$k*4+3]\n";
            }
        } else {
            print "$i @sortedIP\n";
        }
    }
}

sub by_IP
{
    @fieldsa = split(/\s+/ , $a);
    @fieldsb = split(/\s+/ , $b);
    @avec = split(/\./ , $fieldsa[2]);
    @bvec = split(/\./ , $fieldsb[2]);

    $avec[0] <=> $bvec[0]
        or
    $avec[1] <=> $bvec[1]
        or
    $avec[2] <=> $bvec[2]
        or
    $avec[3] <=> $bvec[3]
        or
```

```
    $fieldsa[0] cmp $fieldsb[0]
}

exit;
```

7.2.2 unsolicited.pl

```
#!/usr/bin/perl
#
# unsolicited8.pl - Joanne Treurniet
# Used to find unsolicited SYN-ACKs or RSTs or FINs.
#
# A SYN-ACK must be preceeded by a SYN on the same port
# RSTs that are not in response to anything constitue a RST scan.
# => a preceeding established connection (S bit not set) on the same port
# FINs that are not ending a conversation constitute a FIN scan.
# => a preceeding established connection (S bit not set) on the same port
#
$HOME = "/home/jtreurniet";
$ENV{PATH} = "/bin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin";
$ANALYZER_DIR = "$HOME/DREnetAnalysis/Raw";

if($ARGV[0] eq undef)
{
   print "Usage: unsolicited8.pl <day of the month of August> <hour>.\n";
   exit;
}

$day = $ARGV[0];
$hour = $ARGV[1];
$prevhour = $hour - 1;
# prepend a zero to the hour if it's one digit only:
if ($prevhour < 10) { $prevhour = "0$prevhour"};
$subdir = "Aug".$day;

$zipped_file="$ANALYZER_DIR/$subdir/tcp.200008$day$hour.gz";

chdir("$ANALYZER_DIR/$subdir");

$filename = "$ENV{PWD}/unsol-$day$prevhour.dat";

# Check for the presence of a file for the previous hour (unsol-$day$hour.dat)
if (! -e $filename) {
    warn "Cannot read $ENV{PWD}/unsol-$day$prevhour.dat: $!";
} else {
    open(LASTHOUR, $filename);
    @all = <LASTHOUR>;
    close(LASTHOUR);
    foreach $dataline (@all) {
        $line = $dataline;
        # Check the time: if it's too old ($hour-2) don't read it in.
        @time = split(/\:/,$line);
        $hour1 = $time[0];
        if ($time[0] >= $hour - 2 ) {
            # put the data in the arrays
            @fields = split(/\s+/, $line);

            # get the source and dest IPs
            @srcoctets = split(/\./,$fields[2]);
            $srcip = "$srcoctets[0].$srcoctets[1].$srcoctets[2].$srcoctets[3]";
            @dstoctets = split(/\./,$fields[4]);
            $dstip = "$dstoctets[0].$dstoctets[1].$dstoctets[2].$dstoctets[3]";
            $srcport = $srcoctets[4];
            $dstport = $dstoctets[4];
```

```
            $dstport =~ tr/://d;
            # if the first 5 chars of $srcip is "131.13", use the dstIP
            # to separate.  Otherwise, use the srcIP.
            # also use the port>1023 to separate the second hash.
            if ($srcip=~m/^131\.136/ or $srcip=~m/^131\.135/ or $srcip=~m/^131\.132/) {
                if ($srcport>$dstport) {
                    push @{ $HoHoL{"$dstip-$srcip"}{"$srcport-$dstport"} },$line;
                } else {
                    push @{ $HoHoL{"$dstip-$srcip"}{"$dstport-$srcport"} },$line;
                }
            } else {
                if ($srcport>$dstport) {
                    push @{ $HoHoL{"$srcip-$dstip"}{"$srcport-$dstport"} },$line;
                } else {
                    push @{ $HoHoL{"$srcip-$dstip"}{"$dstport-$srcport"} },$line;
                }
            }
        } else {
#           print "$hour1: $line\n";
        }
    }
}


# Grab all SYN only, SYN-ACK and anything with RST set and anything with FIN set.
$cmd1 = "gunzip -c $zipped_file" . " | "
    . "tcpdump -S -n -r - 'tcp and (tcp[13]&0x3f=0x02 or tcp[13]&0x3f=0x12 or
tcp[13]&0x04!=0 or tcp[13]&0x01!=0) and (not net 131.137 or (src net 131.137 and (dst
net 131.136 or 131.135 or 131.132)) or (dst net 131.137 and (src net 131.136 or
131.135 or 131.132)))'";

open(RAWFILE,"$cmd1|");
print STDERR "Reading SYNs, SYN-ACKs and RSTs and FINs...\n";
print STDERR "$cmd1\n";

while($line=<RAWFILE>)
{
    next if /gre-proto/;
    next if /trunc/;
    @fields = split(/\s+/, $line);
    # get the source and dest IPs
    @srcoctets = split(/\./,$fields[2]);
    $srcip = "$srcoctets[0].$srcoctets[1].$srcoctets[2].$srcoctets[3]";
    @dstoctets = split(/\./,$fields[4]);
    $dstip = "$dstoctets[0].$dstoctets[1].$dstoctets[2].$dstoctets[3]";
    $srcport = $srcoctets[4];
    $dstport = $dstoctets[4];
    $dstport =~ tr/://d;
    # if the first 5 chars of $srcip is "131.13", use the dstIP to
    # separate.  Otherwise, use the srcIP.
    # also use the port>1023 to separate the second hash.
    if ($srcip=~m/^131\.136/ or $srcip=~m/^131\.135/ or $srcip=~m/^131\.132/) {
        if ($srcport>$dstport) {
            push @{ $HoHoL{"$dstip-$srcip"}{"$srcport-$dstport"} },$line;
        } else {
            push @{ $HoHoL{"$dstip-$srcip"}{"$dstport-$srcport"} },$line;
        }
    } else {
        if ($srcport>$dstport) {
            push @{ $HoHoL{"$srcip-$dstip"}{"$srcport-$dstport"} },$line;
        } else {
            push @{ $HoHoL{"$srcip-$dstip"}{"$dstport-$srcport"} },$line;
        }
    }
}
close(RAWFILE);
```

```
print STDERR "Searching for anomalies...\n";

# For each outside host we found,
foreach $outsideIP ( keys %HoHoL ) {

# check for TCP replays.

    foreach $port (keys %{$HoHoL{$outsideIP}}) {

    for ($i=0; $i<=$#{ $HoHoL{$outsideIP}{$port}}; $i++) {
        $line = $HoHoL{$outsideIP}{$port}[$i];
        @time = split(/\:/,$line);
        $min1 = $time[1];
        @fieldsa = split(/\s+/, $line);
        for ($j=$i+1; $j<=$#{ $HoHoL{$outsideIP}{$port}}; $j++) {
            $line = $HoHoL{$outsideIP}{$port}[$j];
            @time = split(/\:/,$line);
            $min2 = $time[1];
            if (abs($min2-$min1)>10) {
                last;
            } else {
                @fieldsb = split(/\s+/, $line);
                if (($fieldsa[2] eq $fieldsb[2]) and
                    ($fieldsa[4] eq $fieldsb[4]) and
                    ($fieldsa[5] eq $fieldsb[5]) and
                    ($fieldsa[6] eq $fieldsb[6]) and
                    ($fieldsa[7] eq $fieldsb[7]) and
                    ($fieldsa[8] eq $fieldsb[8])) {
# Use the "splice" function to remove the repeated element:
                    splice (@{ $HoHoL{$outsideIP}{$port}}, $j, 1);
                    last;
                }
            }
        }
    }


# For each packet in the array for $outsideIP:

    for ($i=0; $i<=$#{ $HoHoL{$outsideIP}{$port}}; $i++) {
        $line = $HoHoL{$outsideIP}{$port}[$i];
        $match=0;
        @fields = split(/\s+/, $line);

# Search for SYN-ACK scans:

        if ($fields[5] eq "S" and $fields[7] eq "ack") {
            # It's a SYN-ACK, search for a preceeding SYN
            $line = $HoHoL{$outsideIP}{$port}[$i - 1];
            @fields = split(/\s+/, $line);
            if ($fields[5] eq "S" and $fields[7] eq "win") {
                $match=1;
#  If the SYN-ACK is the last in the array, the conversation is
#  unfinished and the line should be added to the @unfinished array.
#  also if the SYN_ACK is followed by a single FIN, the conversation
#  still hasn't ended.
                if ($i == $#{ $HoHoL{$outsideIP}{$port}}) {
                    push @unfinished, $HoHoL{$outsideIP}{$port}[$i - 1];
                    push @unfinished, $HoHoL{$outsideIP}{$port}[$i];
                } elsif ($i == $#{ $HoHoL{$outsideIP}{$port}} - 1) {
                    @fields = split(/\s+/, $HoHoL{$outsideIP}{$port}[$i + 1]);
                    if ($fields[5] eq "FP" or $fields[5] eq "F") {
                        push @unfinished, $HoHoL{$outsideIP}{$port}[$i - 1];
                        push @unfinished, $HoHoL{$outsideIP}{$port}[$i];
                        push @unfinished, $HoHoL{$outsideIP}{$port}[$i + 1];
                    }
                }
```

```
                last;
            }
            if ($match!=1) {push @synackarr, $HoHoL{$outsideIP}{$port}[$i];}

#  Search for Reset scans:

        } elsif ($fields[5] eq "R") {
            # It's a RST, look for a preceeding anything.
            # if it's the only one, log it.
            # if $i=0, log (it's the first occurrence and a fin shouldn't come first).
            if ($#{ $HoHoL{$outsideIP}{$port} } <1 or $i eq 0) {
                push @rstarr, $HoHoL{$outsideIP}{$port}[$i];
            }

# Search for FIN scans:

        } elsif ($fields[5] eq "F" or $fields[5] eq "FP") {
            # It's a FIN, look for a preceeding anything.
            # if it's the only one, log
            # if $i=0, log (it's the first occurrence and a fin shouldn't come first).
            if ($#{ $HoHoL{$outsideIP}{$port} } <1 or $i eq 0) {
                push @finarr, $HoHoL{$outsideIP}{$port}[$i];
            }
        }
    }
}
}

# Print out the contents of the arrays, first the Unsolicited SYN-ACKs,
# Unsolicited RSTs, then the Unsolicited FINs.

print STDERR "Outputting...\n";

print "</pre><hr><h3>Unsolicited SYN-ACKs</h3><pre>\n";
@newsynackarr = sort by_IP @synackarr;
for ($j=0;$j<=$#synackarr;$j++) {
    print $newsynackarr[$j];
}

print "</pre><hr><h3>Unsolicited RSTs</h3><pre>\n";
@newrstarr = sort by_IP @rstarr;
for ($j=0;$j<=$#rstarr;$j++) {
    print $newrstarr[$j];
}

print "<hr><h3>Unsolicited FINs</h3>\n<pre>";
@newfinarr = sort @finarr;
for ($i=0;$i<=$#newfinarr;$i++) {
    print $newfinarr[$i];
}
print "</pre>\n";

# Check for the presence of a file for the previous hour (unsol-$day$hour.dat)
open(THISHOUR,">$ENV{PWD}/unsol-$day$hour.dat") || die "Cannot open $ENV{PWD}/unsol-
$day$hour.dat: $!";
@newunfinished = sort @unfinished;
for ($i=0;$i<=$#newunfinished;$i++) {
    print THISHOUR $newunfinished[$i];
}
close(THISHOUR);

sub by_IP
{
    @fieldsa = split(/\s+/ , $a);
    @fieldsb = split(/\s+/ , $b);
    @avec = split(/\./ , $fieldsa[2]);
    @bvec = split(/\./ , $fieldsb[2]);
```

```
    $avec[0] <=> $bvec[0]
        or
    $avec[1] <=> $bvec[1]
        or
    $avec[2] <=> $bvec[2]
        or
    $avec[3] <=> $bvec[3]
        or
    $fieldsa[0] cmp $fieldsb[0]
}

exit;
```

7.2.3 Traffic-stats.pl

```
#!/usr/bin/perl
#
# traffic-stats.pl - Joanne Treurniet
#
# script to generate statistics from a series of tcpdump files.
#
# CYCLES THROUGH ALL HOURS OF ALL DAYS
#
$HOME = "/home/jtreurniet";
$SITE = "GENERIC";
$ENV{PATH} = "/bin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin";
$ANALYZER_DIR = "$HOME/DREnetAnalysis/Raw";

if($ARGV[0] eq undef)
{
   print "Usage: traffic-stats.pl <start day> <start hour> <end day> <end hour>.\n";
   exit;
}

$startday = $ARGV[0];
$starthour = $ARGV[1];
$endday = $ARGV[2];
$endhour = $ARGV[3];

print "day hour ntcp nudp nicmp nipsec nother ntotal\n";


open(TCPFILE,">tcp.data") || die;
print TCPFILE
"day\thour\tnhttp\tnftp\tnmail\tntcp\tnnntp\tnrtsp\tnauth\tndns\tntpo\tnrsync\tntcp_ot
\n";

open(UDPFILE,">udp.data") || die;
print UDPFILE "day\thour\tnudp_dns\tnntp\tnmulti\tnnetbio\tnudp_ot\tnudp\n";

open(ICMPFILE,">icmp.data") || die;
print ICMPFILE
"day\thour\tnreques\tnreply\tndu\tnpu\tnquench\tntimex\tnicmp_o\tnicmp\n";

for ($day=$startday; $day<=$endday; $day++) {
    $subdir = "Aug".$day;
    for ($hour=$starthour; $hour<=$endhour; $hour++) {
        $zipped_file="$ANALYZER_DIR/$subdir/tcp.200008$day$hour.gz";
        chdir("$ANALYZER_DIR/$subdir");

# Initilaize counters
$ntcp=0;
        $nhttp=0;
```

```
        $nmail=0;
        $nftp=0;
        $nnntp=0;
        $nrtsp=0;
        $nauth=0;
        $ndns=0;
        $ntpo=0;
        $nrsync=0;
        $ntcp_other=0;
$nudp=0;
        $nudp_dns=0;
        $nntp=0;
        $nmulti=0;
        $nnetbios=0;
        $nudp_other=0;
$nicmp=0;
        $nreply=0;
        $nrequest=0;
        $ndu=0;
        $npu=0;
        $nquench=0;
        $ntimex=0;
        $nicmp_other=0;
$nipsec=0;
$nother=0;

# Read all TCP
$cmd1 = "gunzip -c $zipped_file" . " | "
    . "tcpdump -n -S  -r - tcp";
open(RAWFILE,"$cmd1|");
print STDERR "Reading file $zipped_file...\n";
print STDERR "$cmd1\n";
while(<RAWFILE>)
{
    next if /gre-proto/;
    next if /trunc/;
    $ntcp++;
    # Split this number into the interesting protocols
    @fields = split(/\s+/, $_);
    @srcoctets = split(/\./,$fields[2]);
    @dstoctets = split(/\./,$fields[4]);
    $srcport = $srcoctets[4];
    $dstport = $dstoctets[4];
    $dstport =~ tr/:://d;
    if ($srcport eq 80 or $dstport eq 80 or $srcport eq 8080 or $dstport eq 8080 or
        $srcport eq 443 or $dstport eq 443 or $srcport eq 8443 or $dstport eq 8443) {
        $nhttp++;
    } elsif ($srcport eq 25 or $dstport eq 25 or $srcport eq 110 or
             $dstport eq 110 or $srcport eq 143 or $dstport eq 143) {
        $nmail++;
    } elsif ($srcport eq 20 or $dstport eq 20 or $srcport eq 21 or $dstport eq 21) {
        $nftp++;
    } elsif ($srcport eq 119 or $dstport eq 119) {
        $nnntp++;
    } elsif ($srcport eq 554 or $dstport eq 554) {
        $nrtsp++;
    } elsif ($srcport eq 113 or $dstport eq 113) {
        $nauth++;
    } elsif ($srcport eq 53 or $dstport eq 53) {
        $ndns++;
    } elsif ($srcport eq 102 or $dstport eq 102) {
        $ntpo++;
    } elsif ($srcport eq 873 or $dstport eq 873) {
        $nrsync++;
    } else {
        $ntcp_other++;
    }
```

```
}
close(RAWFILE);

# Read all UDP
$cmd1 = "gunzip -c $zipped_file" . " | "
    . "tcpdump -n -S  -r - udp and not port 500";
open(RAWFILE,"$cmd1|");
print STDERR "Reading file $zipped_file...\n";
print STDERR "$cmd1\n";
while(<RAWFILE>)
{
    next if /gre-proto/;
    next if /trunc/;
    $nudp++;
    @fields = split(/\s+/, $_);
    @srcoctets = split(/\./,$fields[2]);
    @dstoctets = split(/\./,$fields[4]);
    $srcport = $srcoctets[4];
    $dstport = $dstoctets[4];
    $dstport =~ tr/://d;
    if ($srcport eq 53 or $dstport eq 53) {
        $nudp_dns++;
    } elsif (($srcoctets[0] ge 224 and $srcoctets[0] le 239) or
             ($dstoctets[0] ge 224 and $dstoctets[0] le 239)) {
        $nmulti++;
    } elsif ($srcport eq 123 or $dstport eq 123) {
        $nntp++;
    } elsif ($srcport eq 137 or $dstport eq "137NBT") {
        $nnetbios++;
    } else {
        $nudp_other++;
    }
}
close(RAWFILE);

# Read all icmp
$cmd1 = "gunzip -c $zipped_file" . " | "
    . "tcpdump -n -S  -r - icmp";
open(RAWFILE,"$cmd1|");
print STDERR "Reading file $zipped_file...\n";
print STDERR "$cmd1\n";
while(<RAWFILE>)
{
    next if /gre-proto/;
    next if /trunc/;
    $nicmp++;
    @fields = split(/\s+/, $_);
    if ($fields[7] eq "reply") {
        $nreply++;
    } elsif ($fields[7] eq "request") {
        $nrequest++;
    } elsif ($fields[8] eq "unreachable") {
        $ndu++;
    } elsif ($fields[10] eq "unreachable") {
        $npu++;
    } elsif ($fields[6] eq "time") {
        $ntimex++;
    } elsif ($fields[6] eq "source") {
        $nquench++;
    } else {
        $nicmp_other++;
    }
}
close(RAWFILE);

# Read all ipsec
$cmd1 = "gunzip -c $zipped_file" . " | "
```

```
    . "tcpdump -n -S  -r - 'ip[9]=50 or (udp and port 500)'";
open(RAWFILE,"$cmd1|");
print STDERR "Reading file $zipped_file...\n";
print STDERR "$cmd1\n";
while(<RAWFILE>)
{
    next if /gre-proto/;
    next if /trunc/;
    $nipsec++;
}
close(RAWFILE);

# Read all others
$cmd1 = "gunzip -c $zipped_file" . " | "
    . "tcpdump -n -S  -r - 'not tcp and not udp and not icmp and ip[9]!=50'";
open(RAWFILE,"$cmd1|");
print STDERR "Reading file $zipped_file...\n";
print STDERR "$cmd1\n";
while(<RAWFILE>)
{
    next if /gre-proto/;
    next if /trunc/;
    $nother++;
    @fields = split(/\s+/, $_);
    print "Other protocol: $fields[5]\n";
}
close(RAWFILE);

$ntotal = $ntcp+$nudp+$nicmp+$nipsec+$nother;

print "ALL: $day $hour $ntcp $nudp $nicmp $nipsec $nother $ntotal\n";
print TCPFILE
"$day\t$hour\t$nhttp\t$nftp\t$nmail\t$ntcp\t$nnntp\t$nrtsp\t$nauth\t$ndns\t$ntpo\t$nrs
ync\t$ntcp_other\t$ntcp\n";
print UDPFILE
"$day\t$hour\t$nudp_dns\t$nntp\t$nmulti\t$nnetbios\t$nudp_other\t$nudp\n";
print ICMPFILE
"$day\t$hour\t$nrequest\t$nreply\t$ndu\t$npu\t$nquench\t$ntimex\t$nicmp_other\t$nicmp\
n";

}
}
close(TCPFILE);
close(UDPFILE);
close(ICMPFILE);

exit;
```

7.3 Annexe C: Summary of Snort Alerts

Table C-1: Snort alerts and their rate of occurrence over the period.

% OF ATTACKS	# OF ATTACKS	SNORT ALERT NAME
30.37	1822	IDS115 - MISC-Traceroute-UDP
20.27	1216	MISC-Attempted Sun RPC high port access
10.19	611	MISC-WinGate-8080-Attempt
6.33	380	default Backdoor access!
5.12	307	Possible SubSeven access
2.45	147	WEB-prefix-get //
1.62	97	IDS133 - RPC - portmap-request-rusers
1.18	71	MISC-WinGate-1080-Attempt
1.05	63	FrontPage-shtml.exe
1.03	62	OVERFLOW-NOOP-X86
0.88	53	FrontPage-shtml.dll
0.8	48	Napster 7777 Data
0.73	44	IDS266 - CAN-1999-0261 – SMTP Chameleon Overflow
0.65	39	IDS021 - RPC - portmap-request-nisd
0.58	35	Napster 8888 Data
0.33	20	IDS015 - RPC - portmap-request-status
0.23	14	IDS219 - WEB-CGI-Perl access attempt
0.22	13	IIS-scripts-browse
0.2	12	WEB-CGI-Wguest CGI access attempt
0.2	12	WEB-CGI-Rguest CGI access attempt
0.18	11	IDS008 - TELNET - daemon-active
0.17	10	IIS-*.idc
0.15	9	Whack-a-mole
0.15	9	Netbus/GabanBus

0.13	8	MISC-DNS-version-query
0.13	8	IDS025 - RPC - portmap-request-selection_svc
0.12	7	IIS-srch.asp
0.12	7	WEB-CGI-ksh shell
0.12	7	WEB-CGI-bash shell
0.12	7	CVE-1999-0191 - IIS-newdsn
0.12	7	WEB-CGI-csh shell
0.1	6	WEB-CGI-Aglimpse CGI access attempt
0.1	6	IDS224 - CVE-1999-0045 - NPH CGI access
0.1	6	CVE-1999-0449 - IIS-codebrowser Exair
0.1	6	IDS138 - CVE-1999-0183 - TFTP rootdirectory
0.1	6	IDS258 - Web cgi get32.exe
0.1	6	Possible EvilFTP access
0.1	6	IIS-iisadmpwd
0.1	6	BIND Shell
0.1	6	Possible GirlFriend access
0.1	6	IIS-_vti_inf
0.1	6	IDS128 - CVE-1999-0067 - CGI phf attempt
0.1	6	WEB-CGI-Count.cgi probe!
0.1	6	WEB-CGI-Webdist CGI access attempt
0.1	6	WEB-~root
0.1	6	WEB-CGI-tsch shell
0.08	5	WEB-CGI-Upload CGI access attempt
0.08	5	IDS010 – RPC - portmap-request-rstatd
0.08	5	IDS012 – RPC - portmap-request-ypserv
0.08	5	VNC Active on Network
0.08	5	IDS013 – RPC - portmap-request-mountd
0.08	5	IDS218 - CVE-1999-0070 - TEST-CGI probe

0.08	5	CAN-1999-0229 - IIS WEB-..\..
0.08	5	IDS237 – Web-Frontpage .htw
0.07	4	Possible NetSphere access
0.07	4	WEB-PageService
0.07	4	Possible NetSphere FTP acces
0.07	4	CAN-1999-0736 - IIS-showcode
0.05	3	Napster Client Data
0.05	3	IDS184 - DDoS - TFN client command
0.05	3	IDS006 – MISC-Source Port Traffic 20 TCP
0.05	3	IDS213 – FTP-Password Retrieval
0.03	2	IDS127 - TELNET - Login Incorrect
0.03	2	IDS145 - SCAN-Cybercop-OS-Probe sfp
0.03	2	FTP-nopassword
0.03	2	IDS239 – MISC-PCAnywhere Startup
0.03	2	IDS112 - DDoS - mstream handler to
0.03	2	IDS149 - SCAN-Cybercop OS Probe pa12
0.02	1	Possible Queso Fingerprint attempt
0.02	1	IDS249 - SMTP Relaying Denied
0.02	1	SCAN - Whisker Stealth Mode 4- HEAD
0.02	1	IDS183 - DDoS - TFN client command
0.02	1	IDS137 – CVE-1999-0183 - TFTP parent directory
0.02	1	Back Orifice
0.02	1	IDS027 - SCAN-FIN
0.02	1	WEB-CGI-Wrap CGI access attempt
0.02	1	WEB-CGI-redirectt
0.02	1	IDS197 - DDoS - Trin00

Table C-2: Number of scans detected by Snort during the period. Note that packets with unacceptable TCP flags set which did not target multiple machines or multiple ports were not counted as they are often innocuous.

SCAN TYPE	PORTS	# OF OCCURRENCES
SYN	21, 53, 80, 98, 110, 111, 139, 635, 1524, 6776, 27374	57
SYN-FIN	53, 109, 111, 1524, 2222, 9704	8
UDP	137	1
NULL		1
Multi-port to 1 target	Multiple	5
Multi-method to 1 target, 1 port	23	1
Multi-method to 1 target, multi-port	Multiple	1

List of symbols/abbreviations/acronyms/initialisms

ACHQ	Air Command Head Quarters
ADA	Attack Detection and Analysis
CERT	"CERT" is a registered service mark of Carnegie Mellon University. Although "CERT" is not an acronym, CERT/CC was originally the computer emergency response team.
CF	Canadian Forces
CIRT	Computer Incident Response Team
CRC	Communications Research Centre
CVS	Concurrent Versions System
DMCS	Directorate Maritime Control Systems
DMZ	De-Militarized Zone
DND	Department of National Defence
DNS	Domain Name Service
DRDC	Defence Research and Development Canada
DREnet	Defence Research Establishment network
DREO	Defence Research Establishment Ottawa
FTP	File Transfer Protocol
GIAC	Global Information Assurance Certification
GIAC	Global Incident Analysis Center
GPnet	General Purpose network
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
ID	Intrusion Detection

IDE	Integrated Drive Electronics
IDS	Intrusion Detection System
IO	Information Operations
IP	Internet Protocol
IPSec	IP Security protocol
IRONMAN	InteRactive Ontology for Network MANagement
ISP	Internet Service Provider
LSEC	Land Software Engineering Centre
NAT	Network Address Translation
NDHQ	National Defence Head Quarters
NIC	Network Interface Card
NTP	Network Time Protocol
PC	Personal Computer
RPC	Remote Procedure Call
RTSP	Real-Time Streaming Protocol
SANS	System Administration, Networking, and Security
Scp	Secure copy
SCSI	Small Computer System Interface
SHADOW	Secondary Heuristic Analysis for Defensive Online Warfare
SPAN	Switch Port Analyser
SQL	Structured Query Language
SSH	Secure SHell
TCP	Transmission Control Protocol
TPOC	Technical Point of Contact

UDP	User Datagram Protocol
URL	Uniform Resource Locator
VPN	Virtual Private Network

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Defence Research Establishment Ottawa
Department of National Defence
Ottawa, ON Canada K1A 0Z4

2. SECURITY CLASSIFICATION (overall security classification of the document, including special warning terms if applicable)

UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C or U) in parentheses after the title.)

DREnet Traffic Analysis: Lessons learned in the deployment of an IDS on the DREnet (U)

4. AUTHORS (Last name, first name, middle initial)

Lefebvre, Julie; Treurniet, Joanne

5. DATE OF PUBLICATION (month and year of publication of document)

November 2001

6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.)

54

6b. NO. OF REFS (total cited in document)

13

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Technical Memorandum

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.)

DREO/IO Section

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant)

15bf26

9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written)

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)

DREO TM 2001-100

10b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor)

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(x) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

Full unlimited announcement

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

To better understand the needs of intrusion analysts, the SHADOW Intrusion Detection System (IDS) was deployed on the DREnet. All network traffic at the DREnet-Internet border in a 16-day period was collected and analysed using SHADOW, and again analysed by replaying through the Snort IDS. In this report, Snort and SHADOW are compared and the benefits of deploying an ID System are discussed. Some interesting statistics for the period are presented, as well as a full analysis of a half-day of activity, including scans, effects of denial of service attacks, false alarms and misconfigurations. SHADOW filters, customized for use on the DREnet, and scripts developed for the analysis process are included.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

DREnet; intrusion detection; traffic analysis; SHADOW; Snort