Defence Research and Development Canada | Recherche et développement pour la défense Canada

DEFENCE DÉFENSE

Fast packet recovery

Technical challenges and solutions

Grant Vandenberghe

Defence R&D Canada – Ottawa
TECHNICAL MEMORANDUM
DRDC Ottawa TM 2008-209
October 2008

Canada

Fast packet recovery

Technical challenges and solutions

Grant Vandenberghe
DRDC Ottawa

Defence R&D Canada – Ottawa

Technical Memorandum
DRDC Ottawa TM 2008-209
October 2008

Principal Author

Original signed by G Vandenberghe

G Vandenberghe

Computer Scientist

Approved by

Original signed by Julie Lefebvre

Julie Lefebvre

NIO Section Head

Approved for release by

Original signed by Pierre Lavoie

Pierre Lavoie

Chief Scientist

Abstract

Network security analysts review packets associated with network security events on a daily basis. The packet data that is being reviewed is collected using a distributed real time data acquisition system. The volume of data that is collected is very large making it challenging to store, summarize, access, and manage. This study demonstrates a new technique for storing packet data which uses a selectively recoverable compressed file format that is compatible with the GZIP compression algorithm. This means that compressed files can be readily exchanged and decompressed using standard tools or selected sections of data can be recovered using a modified GZIP program. To summarize the content of the compressed files, an IP based summarization tool is also demonstrated. This summarization tool is faster and generates more compact data files than commonly used flow based traffic summarization tools. Finally, to simplify data collection for the distributed packet logging system, a centralized user interface is demonstrated that can find and recover the packets requested by the analyst.

Résumé

Les analystes de la sécurité réseau examinent quotidiennement les paquets associés aux événements relatifs à la sécurité. Les données relatives aux paquets qui sont examinés sont recueillies au moyen d’un système réparti d’acquisition des données en temps réel. La quantité de données recueillies est extrêmement grande, ce qui rend difficile leur stockage, leur récapitulation, leur gestion ainsi que l’accès à ces données. Cette étude montre une nouvelle technique de stockage des données sur les paquets qui fait appel à un format de fichier comprimé récupérable compatible avec l’algorithme de compression GZIP. Cela signifie qu’il est facile d’échanger et de comprimer ces fichiers comprimés au moyen d’outils standard et qu’on peut récupérer des sections de données sélectionnées au moyen d’un programme GZIP modifié. Nous faisons aussi la démonstration d’un outil récapitulatif IP afin de récapituler le contenu des fichiers comprimés. Cet outil récapitulatif est plus rapide et il génère des fichiers de données plus compacts que les outils récapitulatifs de trafic basés sur la circulation utilisés le plus souvent. Finalement, pour simplifier la collecte de données au moyen du système réparti d’enregistrement de paquets, nous faisons la démonstration d’une interface utilisateur centralisée qui peut trouver et récupérer les paquets demandés par un analyste.

This page intentionally left blank.

Executive summary

Fast packet recovery: Technical challenges and solutions

Grant Vandenberghe; DRDC Ottawa TM [2008-208]; Defence R&D Canada – Ottawa; October 2008.

Background: When network based security events are detected these events need to be validated. To do this verification, the security analysts often need to refer back to the original packet data surrounding the attack. The data itself may reside on a single packet logger or distributed across several data collectors (when parallel paths exist). It is the responsibility of the packet logger to store a copy of every packet crossing a network link. Many packet logging systems either use indexing or compression to improve speed and performance respectively. At present however it is difficult to employ both techniques at the same time. Once the data is compressed the ability to randomly access the data is lost. Systems that do employ compression and indexing often need to decompress the entire file prior to searching the data which is time consuming when large files are involved.

The document describes the work completed as part of the Fast Packet Recovery project whose purpose is to improve the packet recovery speed from an existing packet logger called the Shadow IDS. The revised system allows packet data to be selectively decompressed and adds packet indexing capabilities.

Principal Results: The Fast Packet Recovery project significantly increases the rate of packet recovery by altering the GZIP compression algorithm so that the resulting file becomes selectively de-compressible. The ability to recover a single second rather that the complete hour dramatically improves the rate at which packets can be recovered. An indexing system based on a single IP end point has been developed which is shown to be 9 times smaller and more than 10 times faster than SANCP which is a popular flow-based open source indexing tool. The data recovery tool developed in this study can remotely filter and recover packets from a distributed packet collection system. By combining the remote filtering, indexing, and selective decompression; data can be recovered 10 to 100 times faster than was previously possible using the Shadow IDS.

Significance of Results: The revised compression algorithm allows data to be recovered much faster, saving time and allowing more events to be reviewed during each analyst shift. The techniques for selectively decompressing data could be applied to a number of different fields where the incoming data stream has a pre-sorted index (such as time in the case of a real-time data set).

Future Work: DRDC hopes to release the changes to the Shadow IDS back to the open source community to facilitate better critical infrastructure protection.

Sommaire

Fast packet recovery: Technical challenges and solutions

Grant Vandenberghe; RDDC Ottawa TM 2008-209; R & D pour la défense Canada – Ottawa; Octobre 2008

Contexte: Lorsque des événements relatifs à la sécurité sont détectés, il est nécessaire de valider ces événements. Pour effectuer cette vérification, les analystes en sécurité doivent souvent consulter les données des paquets originaux circulant pendant la période d'attaque. Les données elles-mêmes peuvent résider sur un seul enregistreur de paquets ou elles peuvent être réparties entre divers enregistreurs de données (lorsqu'il existe des voix parallèles). Cet enregistreur de données a pour responsabilité de stocker une copie de chacun des paquets passant par une liaison réseau. De nombreux enregistreurs de paquets utilisent l'indexation ou la compression afin d'améliorer respectivement la vitesse ou les performances. À l'heure actuelle, il est difficile de faire appel aux deux techniques simultanément. Lorsque les données sont comprimées, on perd la faculté d'accéder aléatoirement aux données. Les systèmes qui utilisent la compression et l'indexation doivent souvent décomprimer le fichier entier avant de chercher les données, ce qui exige du temps lorsque de gros fichiers sont traités.

Le document décrit les travaux effectués dans le cadre du projet de récupération rapide de paquets, dont l'objet est d'améliorer la vitesse de récupération des paquets provenant d'un enregistreur de paquets existant, Shadow IDS. Le système révisé permet de décomprimer sélectivement des données et il ajoute des fonctions d'indexation de paquets.

Principaux résultats: Le projet **Récupération rapide de paquets** accélère nettement la vitesse de récupération des paquets en modifiant l'algorithme de compression GZIP de manière que le fichier résultant puisse être décomprimé sélectivement. La possibilité de pouvoir récupérer une seule minute de données au lieu d'une heure complète accélère grandement la vitesse de récupération des paquets. Un système d'indexation basé sur un seul point d'extrémité IP a été développé. Il est 9 fois plus petit et plus de 10 fois plus rapide que SANCP, un logiciel à code source ouvert d'indexation populaire basé sur la circulation de paquets. L'outil de récupération de données développé au cours de cette étude peut filtrer et récupérer à distance des paquets à partir d'un système d'enregistrement de paquets réparti. En combinant le filtrage à distance, l'indexation et la compression sélective, il est possible de récupérer les données de 10 à 100 fois plus rapidement que ce qui était possible au moyen de Shadow IDS.

Importance des résultats: L'algorithme de compression révisé permet de récupérer les données beaucoup plus rapidement, ce qui fait gagner du temps et permet d'examiner un plus grand nombre d'événements pendant chaque quart de travail des analystes. La technique de décompression sélective des données pourrait être appliquée à divers domaines dans lesquels le flux de données entrantes comporte un index trié préalablement (comme le temps dans le cas d'un jeu de données en temps réel).

Travaux futurs: RDDC espère publier les modifications apportées à Shadow IDS au sein de la communauté du code source ouvert afin de faciliter une meilleure protection des infrastructures essentielles.

Table of contents

List of figures

List of tables

1 Packet Analysis Technology Overview

This study investigates improved packet logging techniques. It begins with a survey of packet loggers, their architecture, and capabilities.

1.1 Packet Analysis Background

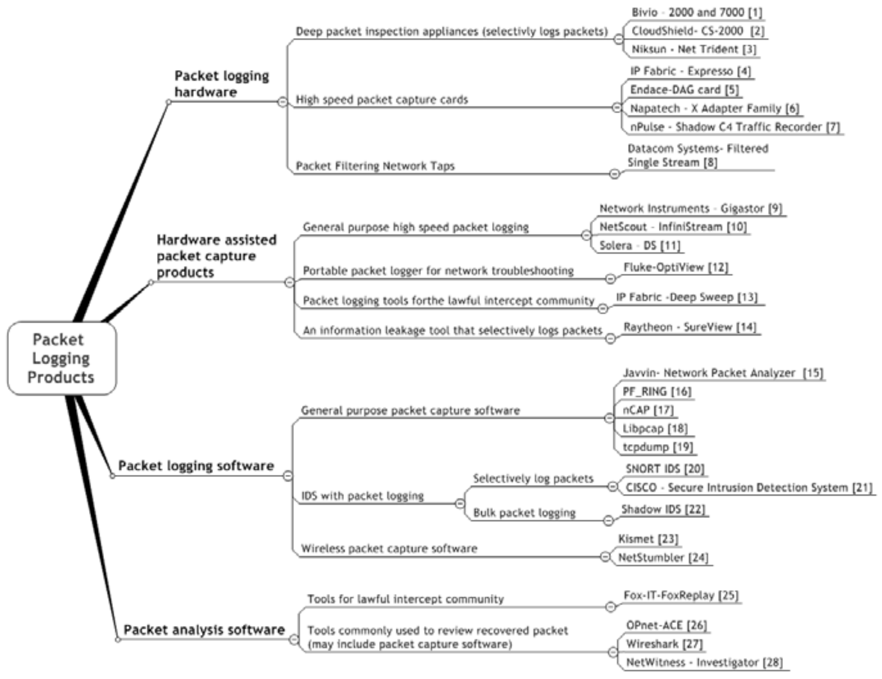
Packet loggers have been commercially available for a number of years. There are two distinct types of packet logging solutions: network-based and host-based. Host-based loggers monitor the packets entering and leaving the local computer. Network-based loggers capture packets for a group of networking devices. Network based packet loggers are often placed near the backbone and record high volumes of packet data. Both network and host based systems use similar hardware and software however each has significantly different data volume and management issues associated with it.

A survey of packet logging hardware, and software is shown in Figure 1. There are a number of highly specialized, high end appliances and network interface cards for receiving and processing packet data at high speed [1-12]. Most of this hardware is intended to operate in the 200Mbps to 10Gbps range. Whereas software based solutions operate at rates below 200 Mbps; however there are software solutions that run on common-off-the-shelf (COTS) hardware that are more cost efficient. Most packet logging solutions try to carve a niche for themselves that distinguish them from their competition as listed below:

- Cloudshield – CS2000 [2] includes a hardware based database,
- Niksum [3] offers products that combines packet logging and Intrusion Detection System (IDS) capabilities.
- IP Fabric - Expresso [4] includes the ability to process packets in parallel in hardware,
- Endace-DAG [5] write packets directly to the computer memory using DMA,
- Network Intruments-NextGen 2 [9] has 24GB of on board memory.
- Products identified in [9-12] are intended for flexible general purpose packet logging
- IP Fabric – Deep Seep [13] include standards-based software interfaces for data collection for the lawful wire tapping community[1],
- Raytheon-SureView [14] is directed towards the information leakage community,
- Products identified in [20-22] are intrusion detection systems that include selective or full packet logging capability

[1] Intercept tools that are intended for full packet capture have extra features that allow for the reconstruction of conversations.

- There are other product identified in [23-24] that are intended for wireless networks[2],
- Fox IT – Fox Replay [25] performs conversation reconstruction for the lawful wiretapping community and
- OPNET – ACE [26] is for intended for application characterization[3].



Packet Logging Products
Packet logging hardware
Deep packet inspection appliances (selectivly logs packets)
Bivio - 2000 and 7000 [1]
CloudShield- CS-2000 [2]
Niksun - Net Trident [3]
High speed packet capture cards
IP Fabric - Expresso [4]
Endace-DAG card [5]
Napatech - X Adapter Family [6]
nPulse - Shadow C4 Traffic Recorder [7]
Packet Filtering Network Taps
Datacom Systems- Filtered Single Stream [8]
Hardware assisted packet capture products
General purpose high speed packet logging
Network Instruments - Gigastor [9]
NetScout - InfiniStream [10]
Solera - DS [11]
Portable packet logger for network troubleshooting
Fluke-OptiView [12]
Packet logging tools forthe lawful intercept community
IP Fabric -Deep Sweep [13]
An information leakage tool that selectively logs packets
Raytheon - SureView [14]
Packet logging software
General purpose packet capture software
Javvin- Network Packet Analyzer [15]
PF_RING [16]
nCAP [17]
Libpcap [18]
tcpdump [19]
IDS with packet logging
Selectively log packets
SNORT IDS [20]
CISCO - Secure Intrusion Detection System [21]
Bulk packet logging
Shadow IDS [22]
Wireless packet capture software
Kismet [23]
NetStumbler [24]
Packet analysis software
Tools for lawful intercept community
Fox-IT-FoxReplay [25]
Tools commonly used to review recovered packet (may include packet capture software)
OPnet-ACE [26]
Wireshark [27]
NetWitness - Investigator [28]

Figure 1: Packet logging solutions

Network based packet loggers have an additional component that distinguish them from their host based counterpart. They require that a copy of the packets crossing the network link or entering the switch is sent to the packet logger. Frequently either a span port on a switch or a network tap is used to make this copy. A network tap is an inline device that can intercept and record all packets crossing a network link in both directions as shown in Figure 2. These network taps act as a data diode allowing a copy of the data crossing the monitoring link to be sent to the packet logger but blocks any attempts for the packet logger to send traffic. Some network taps are now available to filter the data being received by the packet logger as well [29]. The ability to filter the data is useful for situations where you want to either selectively log packet or subdivide a large data stream into more manageable portions for packet logging.

[2] Wireless tools often include tools to break WEP encryption and have reporting features more attuned to the need of a wireless security administrator (signal strength and so forth).
[3] Performance tuning tools have extra reports that characterize the operation of protocols, and emphasize server response time.

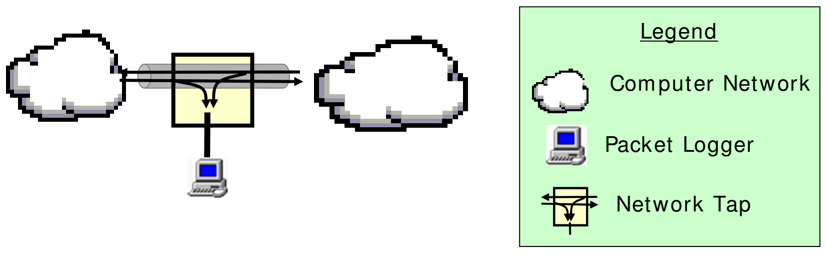


Figure 2: Packet logger with network tap.
In this diagram the network tap allows the packet logger to listen to the packets crossing the network link in both directions but does not allow the packet logger to transmit data.

The packet logger processes and stores data received in a variety of ways. The packet loggers can either store a complete copy of all the packets (bulk full packet logging) [9], a copy of some of the packets (selective packet logging) [20-21], a fraction of the packet (partial packet capture) [21] or a summary description of the packet activity [30]. Continuous full packet logging within an enterprise is often not used because of the amount of storage space required and because of privacy concerns. In its place selective full packet capture is frequently used to perform intercepts of targeted individuals or groups. In selective full packet capture a trigger event causes the packets crossing the network to be captured for later review. In the case of law enforcement professionals this trigger event would be defined through a search warrant. For a network security analyst, an event from an Intrusion Detection System (IDS) would be the trigger. This packet recovery approach works well when a trigger event can be ascertained ahead of time. Unfortunately, one of the problems with selective packet capture is that the events leading up to the attack or unforeseen triggers events can result in critical information being lost.

To illustrate and verify this point a SNORT IDS [20] was configured to selectively capture packets and store them in a database. Several packet traces of successful computer attacks were obtained using data from [31]. The attack traffic was mixed with pre-recorded DRDC traffic, and replayed to the SNORT IDS. The results showed that several of the attacks managed to evade the IDS and in one case the SNORT IDS triggered late (capturing the residual effects of the attack rather than the attack itself). In both of these situations critical packets were being missed making the forensic analysis of the attack difficult if not impossible.

If there is not enough space to store the entire packet in bulk and not enough information to pick the packets then many users choose to do partial packet capture on all packets crossing the network. This means that a fragment of each packet crossing the network is captured. These packet fragments are often very helpful in identifying the general activity surrounding an event and tend to be less invasive from a privacy perspective. The main issue with partial packet capture is that the capture files are still very large.

The last option involving the least amount of data storage involves storing a summary of the packet data. This tends to be least desirable from a security practitioners perspective because there is not enough data to investigate the attack in detail.

An organization's preferred packet logging system will depend on desirable product features, the organization's security policy, and the demands of the particular network on which the packet logger is placed. The chosen packet logging solution also needs to meet the financial resources of the organization. Many of the commercial solutions are quite expensive both in terms of purchasing the system, the human resources involved in deploying one across a geographically distributed organization, and finally to maintain the solution over the long term. Many of the commercial solutions use specialized network cards with a large buffer that can push data to the hard disk at a very high rate. However, having a traffic logger that supports data collection at gigabit line rates at a remote site that has a low speed WAN link is overkill. Supporting multiple incompatible systems is also undesirable.

1.2 Network Packet Collection Architecture

In order to be effective, the placement of the packet loggers on a network need to be properly coordinated to ensure proper coverage. There are a variety of common data collection architectures that a security architect can choose from as shown in Figure 3. The most appropriate architecture choice will vary based on cost versus perceived benefit, the size of the network, and volume of data being recovered. Some security architects choose to deploy a single traffic collector at the core of the network or near the Internet gateway. There are a number of single box packet logging appliances that are well suited to this type of application [15].

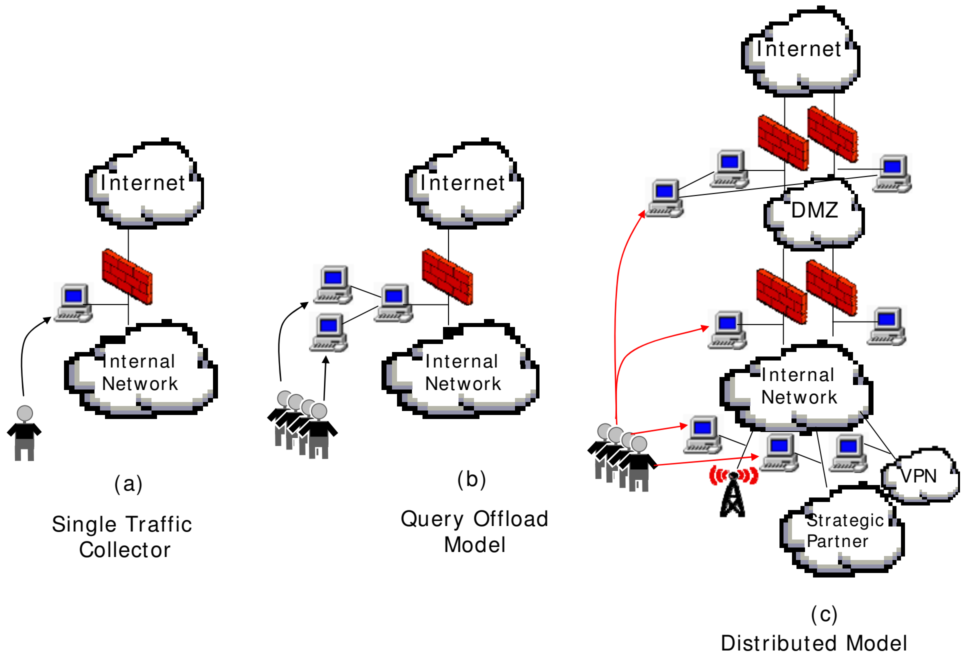


Figure 3: Packet logging placement.

One of the issues with a single box appliance approach is that a single box is being used to simultaneously address data queries and record real time data. An alternative to the single box appliance is a query offload model. In this approach, packet logging and query functions are placed on separate machine or hardware resources[4]. For very intense configurations there may be several query/analysis servers processing packet data[5] [9,22]. By distributing the workload between several servers, CPU, and I/O bottlenecks can be alleviated.

Monitoring a large network from a single point is impractical and insufficient. Often there is more than one security zone, wireless access point, business/coalition partner or Internet connection that needs to be monitored. Packet activity might be either split into parallel paths or obfuscated along a serial path as shown in Figure 4. In some cases the two approaches may be combined[6]. To deal with these situations multiple packet loggers are needed. The packet loggers which are often geographically distributed need to be able to merge the split traffic streams back together before analysis can begin.

[4] In some commercial products the networking card acts like an autonomous data collector while the server handles the queries. The networking card has a big enough buffer that it does not matter if for short periods of time, the server becomes overloaded.

[5]. Multiple long term searches can place a significant workload on any server. As an example when any new network vulnerability is discovered historical network data should be reviewed to determine if the vulnerability was being exploited.

[6] The load balancing systems can sometimes act as an intermediary on the communication path that can hide the address of the true endpoint and provide parallel pathways through the network.

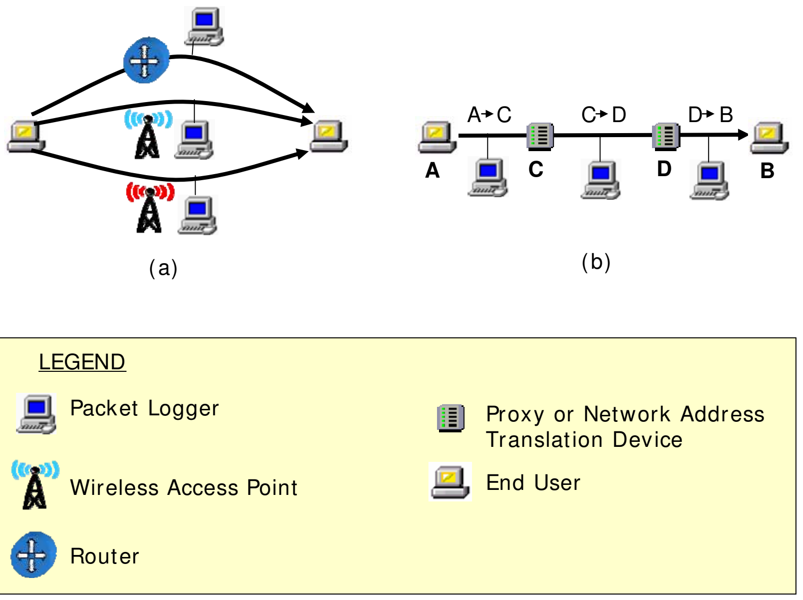


Figure 4: Parallel versus serial pathways through the network.

1.3 Packet Logging Hardware and Software Architecture Options

This section describes the internal hardware and software architecture options used in a packet logger. Figure 5 shows generic overview of how the software is typically structured to interact with the hardware. In general there is a driver layer that interfaces with the network interface card (NIC). The operating system and/or the packet intercept library receive the data from the driver and forward the data to the application layer. Security and performance specialists have done a considerable amount of research into streamlining the data acquisition process. Walravens [32] has examined the performance impact on the way the interrupts are sent between the NIC and the driver. Ethtool [33] is a tool to tune a NIC driver's ring buffer size. Researchers in high performance networking have published revised setting for the LINUX receive buffer sizes [34, 35]. Other researchers have created packet interception function and libraries [16-18]. In general there is a balance between performance and functionality. Lindump [36] is reported to be noticeably faster than the libpcap library but lacks complex filtering and requires a dedicated NIC because of the way the software intercepts the packets [37].

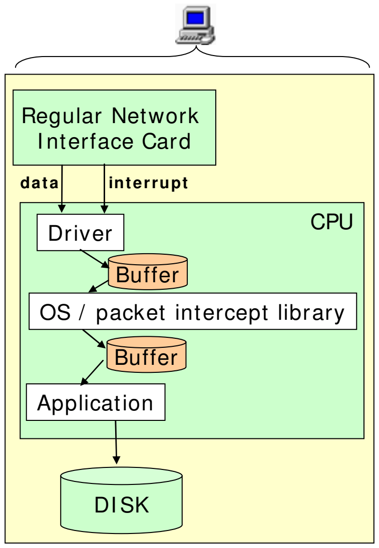


Figure 5: Generic packet capture using COTS hardware.

COTS products based on tcpdump have their limitations in that when they become overloaded they tend to drop packets. To minimize packet losses some researchers have developed parallel packet collection architectures to achieve higher data collection rates [37, 39-41].

An alternative to the COTS approach involves the use of specialized packet acquisition cards. There are a number of vendors that sell high end packet loggers that take a variety of technical approaches as shown in Figure 6. There are several commercial vendors that offer specialty hardware based capture cards. These cards can vary in capability however most include a larger on-board memory buffer and support packet filters at the hardware level. Some cards like Endace-DAG supports direct memory access and some allows the cards to push incoming packets directly to the computers main memory as shown in Figure 6(a). The Endace architecture can be used in a multiprocessor architecture where several equally endowed CPUs inspect packets and selectively save packets in parallel [42].

Network Instruments-NextGen2 [24] card has a considerably different design as shown in Figure 6(b). These cards have a programmable FPGA chip which allows some filtering to be done in hardware logic. The card also has a very large onboard memory buffer that is comparable in size to a small hard disk (24GB).

IP Fabric – Expresso card [43] takes a third approach that is shown in Figure 6(c). This card has multiple network processors that can process packets in parallel or along a serial pipeline. This allows the card to selectively log the packets in a very high speed fashion.

The hardware architecture solutions presented thus far have relied upon changes to the packet logger itself to achieve measurable performance improvement. In some cases where the volume of data exceeds the packet logger's capacity, the data may need to be subdivided before it reaches the packet logger. Figure 6(d) show an example of a smart network tap that distributes the workload across several packet loggers in parallel [5].

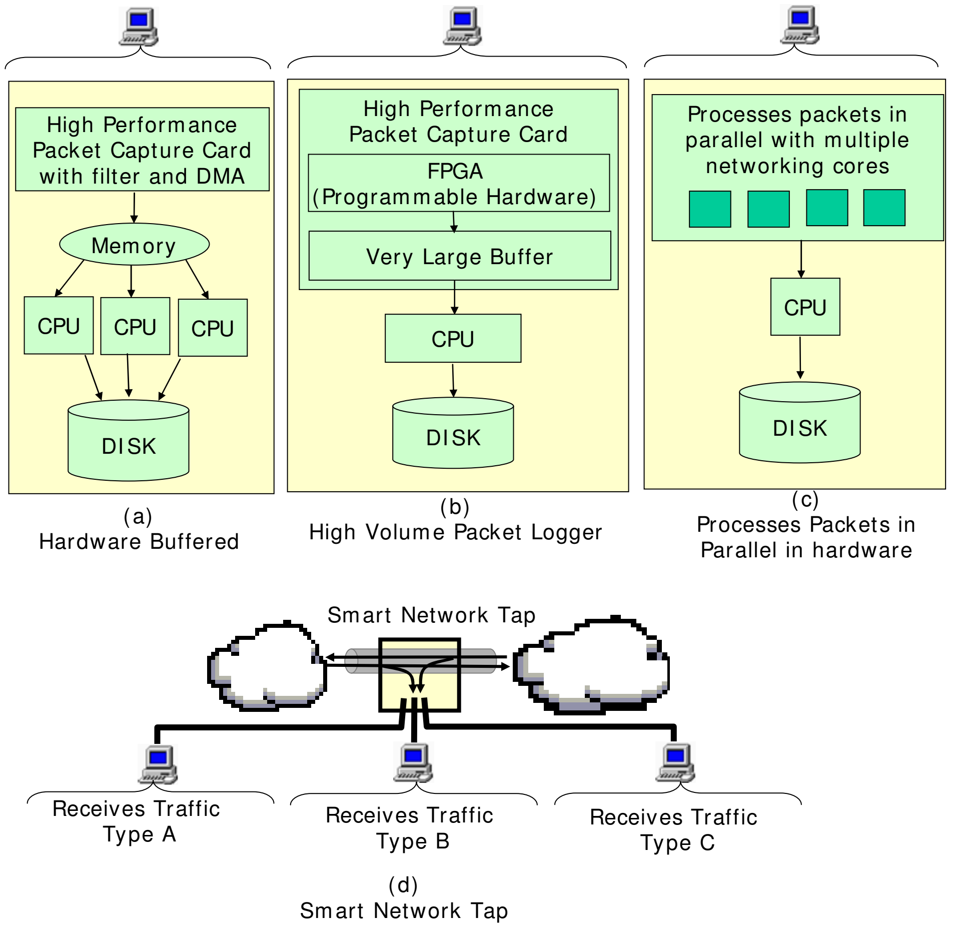

High Performance Packet Capture Card with filter and DMA
Memory
CPU
CPU
CPU
DISK
(a) Hardware Buffered
High Performance Packet Capture Card
FPGA (Programmable Hardware)
Very Large Buffer
CPU
DISK
(b) High Volume Packet Logger
Processes packets in parallel with multiple networking cores
CPU
DISK
(c) Processes Packets in Parallel in hardware
Smart Network Tap
Receives Traffic Type A
Receives Traffic Type B
Receives Traffic Type C
(d) Smart Network Tap

Figure 6: Packet logging hardware architecture options.

1.4 Packet Logging Data Formats

The packets captured by a packet logger can be stored in a number of different data formats:

- Flat files,
- Compressed files,
- Indexed files or a database.

In general there is the perception that storing data in a database is preferable because of the ability to access the data quickly. However a simple test involving triggering SNORT to redirect all packets to a database showed that SNORT could save about a million packets per hour into a database. Unfortunately this result is far too slow for the intended application. Since the SNORT IDS is intended to be used as an IDS not a packet logger a second test was performed to ascertain whether a dedicated database application would be more efficient. The test results showed that the dedicated application could log 9 million packets per hour into a database. Although this result was much faster than the SNORT approach, it falls short of the estimated 60 million packets/hour load. Although it is foreseeable that other kinds of databases such as hardware based [2], in-memory [44] or clustered approaches [45] would be fast enough, they would add a significant amount of cost and complexity that would be difficult to justify. The database approach also requires an exportation step to convert the database record into the PCAP or ASN.1 formats.

Flat file formats are commonly used in data logging application because they are simple and quick. Packet loggers that use the libpcap library [18] such as tcpdump [19] automatically store data to disk in the PCAP format. Lab testing showed that tcpdump is able to meet the 60 million packets/hour design requirement[7]. Unfortunately the PCAP format does not provide an index system to make the data searchable. In addition, since each packet can vary, length jumping to the middle of a capture window is also difficult.

Enterprise packet logging consumes a great deal of disk space and it is not uncommon for a rack mounted server to run out of space in the space of a couple days[8]. This length of time could be significantly extended by compressing the packet data. Table 1 shows the results of a simple file size test for a large traffic file. A compressed file is about half the size of an uncompressed flat file and a quarter the size of a database.

[7] Testing on a Xeon (64-bit) dual quad core system showed that tcpdump could keep up with the 330 million packets per hour produced by the traffic generator. Testing on an older Pentium 4 (32-bit) system showed that packets will start to be lost at about half of this rate.

[8] The exact time is dependent on the size of the disk, the packet capture size, and the network load.

Table 1: Storage format versus file size

File Storage Approach	File Size
Compressed File	692 MB
Uncompressed File	1461 MB
Database	3142 MB

The main problem with storing information in a compressed file is that compressed file formats like GZIP [46-47] do not support random file access. However, if it is possible to modify a compressed file to make it searchable a significant performance benefits could be realized.

In summary each storage system has its weaknesses. Database systems tend to be too slow to keep up with the real time demands of packet logging. Compressed files do not support random access and are difficult to search efficiently. There has been some research into blending techniques. Sybase offers a database that offers page at a time compression using GZIP [48]. Goldstein [49] demonstrates how a page level compression technique for low cardinality fixed with B-tree/R-tree indices. A comparison of file, page, record, and attribute compression was demonstrated in [50]. Multiple attribute compression was examined in [51]. Although these papers claimed to make some improvements to database speed the magnitude was perceived to be insufficient for this application. The approach used here involved taking something that is fast enough and slowly adding features to it rather than taking something that is too slow and trying to find ways to increase the speed.

1.5 Packet Logging Data Exchange Standards

There are two data management issues that are associated with the packet logging. One issue surrounds how data is stored locally on the packet logger which was the subject of the previous section. The other issue is how the data is exchanged with other servers on a distributed basis. Data interoperability is important because it allows third party tools and external parties to review the data.

There are a number of standards that are used by law enforcement agencies (LEA) to standardize the exchange of wiretapping data. These standards cover all forms of electronic communication including telephone and packet data. Although this study is intended for enterprise packet logging, maintaining some degree of joint interoperability is important. By having the data stored in a form that is suitable for handoff to another LEA without conversion would also be useful.

Although each country has its own laws surrounding interception of computer communications [52,53] two of the most common standards are ETSI TS 102 232 [54] and Communication Assistance for Lawful Enforcement Act (CALEA) [55]. CALEA is the standard used in the United States while the ETSI standard is specific to Europe [56]. Please refer to Annex A for more details description of these standards. At the time of this writing Canada is still in the process of enacting legislation [57] which will define the standard this country will follow.

The PCAP file format used in CALEA was chosen for the current work; however, the summary data collected by this technique does not follow the CMII data format. At present, the packet logger does not have access to the data needed to attribute packets to their end user. Also the CMII data format more closely resembles a flow based summarization approach which differs from the approach taken in the study. An example of a product that offers both PCAP and CMII export can be found from [62].

1.6 Shadow IDS as a Packet Logger

It has been established that there are a wide range of packet loggers, each with its own set of technical features. For the targeted application in this study, the following design requirements exist:

- the solution must be capable of collecting the projected packet volume preferably using COTS hardware,
- the data recovered needs to be compressed to maximize data retention time
- the data needs to be stored in a format that would facilitate interoperability,
- the data needs to be readily searchable on at least 2 indices (time and IP address),
- the packet logger needs to be extremely easy to manage,
- the packet logger needs to be geographically distributed,
- the system needs to handle packets which may cross the network via multiple paths, and
- the data must be recoverable at a central analyst station.

None of the packet loggers reviewed met the entire list of requirements; however there was one open source solution that was close. The Shadow IDS [22] is a distributed packet logger that stores packets in a compressed PCAP format on an hourly basis; this simplifies data management activities. The Shadow IDS also includes a series of perl scripts that review the hourly traffic logs and looks for anomalous traffic behaviour (this is why it is referred to as an IDS rather than a packet logger).

The Shadow IDS is quite old and there are concerns about whether the Shadow IDS is fast enough to log packets for the intended application. Tests were performed on the Pentium 4 and XEON class servers to determine the peak packet capture speed. As shown in Table 2, when an (32-bit) Pentium 4 system is used as a packet logger the performance of the default Shadow IDS is inferior to tcpdump. The original Shadow IDS uses the GZIP compression algorithm in its default compression configuration. Further testing showed that if the fastest compression switch is turned on, the overall performance of the Pentium 4 system could be significantly improved. The trade off for fast compression is a slight reduction of the compression efficiency.

Table 2: Packet loss rates associated with real time data collection 3.4GHz Pentium 4

File Storage Approach	Total Packets Dropped (100 million packets @ 80Mbps)
Shadow with GZIP in default compression mode	76000
Standard tcpdump (creates a PCAP flat file)	52000
Shadow with GZIP in fast compression mode	27000

Some additional performance testing of the Shadow IDS on a current generation (2.6Ghz 64-bit dual quad) XEON processor has been conducted. As expected, the performance of the XEON system is significantly better than the older Pentium 4 systems. As shown in Table 3 the system could handle about 244 Mbps of traffic with tcpdump and 200Mbps with GZIP in fast compression mode. In the case of the XEON architecture there is a sharp and sudden transition between complete packet capture and random volumes of packet loss. As a result the testing methodology was changed to measure the peak volume of data stored before packet losses started to appear. The results indicate that dumping packet data directly to disk is faster than any attempt to compress the packets before it is stored on disk. The performance results also showed that the Shadow IDS met with the performance requirements of the targeted application.

Table 3: Packet loss rates associated with real time data collection 2.6 GHz Xeon dual quad

File Storage Approach	Top Speed	File Size
Shadow with GZIP in default compression mode	82 Mbps	44.5 MB
Standard tcpdump (creates a PCAP flat file)	244 Mbps	98.7 MB
Shadow with GZIP in fast compression mode	200 Mbps	48.3 MB

Although the Shadow IDS was fast enough to capture the peak volumes of traffic for the targeted application, there are a number of issues with it. The scripts that manage the Shadow IDS's data acquisition capabilities stop and restart the packet capture scripts on an hourly basis which results in several seconds of packets being lost each hour. The Shadow IDS needed to be modified so that the data can be continuously stored across a rollover window.

The Shadow IDS also provides several reports that describe gross network usage statistics and top talkers on the network. However these hourly reports provide minimal data granularity and are taking excessively long periods of time to be processed using PERL scripts which are being run on an hourly or daily basis. Adding the ability to perform some real time capture statistics would be beneficial.

The Shadow IDS can move data to a central point for analysis and display the data on a web interface but this approach assumes that it is efficient to move the data to a central location. There needed to be a means added to recover the data on a distributed basis. This would allow the analysts to perform his work through a single console window quickly and efficiently.

Finally, one of the most serious deficiencies of the Shadow IDS is that the compressed files are not readily accessible without decompressing large volumes of data. When this decompression activity is performed by multiple analysts the net effect is very slow response times. To address this issue, a searchable compressed file technique that allows analysts to quickly recover the packets from a compressed file was implemented.

1.7 Summary and Project Goal

As discussed in this section there are a number of different packet loggers available. For this particular application the Shadow IDS running on COTS hardware is sufficiently fast to process the volume of network traffic occurring on the target network. The main outstanding issue is that the packets obtained following this approach are not being stored in a format that supports immediate access. Normally a database would have been ideal data storage choice however initial lab testing showed that a simple database design was not fast enough to keep up with the incoming packets stream and would bloat the file size. This meant that either a flat file or compressed file format approach would need to be used to store the data. Creation of an indexed compressed file would allow for significantly larger amounts of packets to be retained on the target sensor.

The goal of this study was to create a searchable compressed file. Specifically, the standard GZIP file compression format will be modified so that the file format remains compatible with the standard GZIP program, but will also be stored in a form that makes it selectively de-compressible. A further goal was to facilitate data summarization and centralized data recovery.

This paper is organized as follows. Section 2 describes the changes made to the compression program and improvements made to the packet logger's performance. Section 3 describes the changes made to summarize the IP address activity and broaden the range of data queries. Section 4 describes the changes made to ensure that no packets will be dropped during the hourly rollover of the data files. Section 5 describes the user interface that allows selective recovery from a remotely distributed array of packet loggers.

2 Compressed Packet Data Collection

This section describes the process that has been followed to create a searchable compressed file. Rather than writing a new compression algorithm from scratch the decision was made to modify the existing GZIP compression algorithm [63-64]. Overall the changes have been done in a way that extends GZIP's original capabilities while still maintaining strict compatibility with the original GZIP program.

One of the questions that arise is why the general purpose GZIP compression algorithm is being used to compress packet data. Is GZIP the best choice for compressing data? Also, are there any other compression algorithms available that can perform selective data recovery? During the initial information survey of compression programs it was determined that there are many compression programs but none were found that would perform selective data recovery. One excellent resource on the web is the maximum compression website [66] which has a list of 200 compression programs listed and includes a performance comparison of the different tools for a range of file types. For this particular application a compression algorithm was needed that had the following characteristics:

- Fast data compression
- Linux compatible
- Reasonable compression efficiency[9]
- Reasonable user base

Using the above performance criteria and the performance results from the Maximum Compression website, the tools were narrowed to a select few. The ranking showed that the open source GZIP program had the fastest compression engine that was commonly available in Linux[10].

Normally a file compressed using the GZIP compression algorithm cannot be selectively decompressed[11] [65]. This is because the information gathered earlier in the program is reused later to compress the data. These backwards references and two layers of compression make it difficult to jump directly to the middle of a file and begin decompression. This section demonstrates how the GZIP compression algorithm can be modified to support selective decompression. It also examines the performance implications of making these changes.

[9] There was a tradeoff being made between compression efficiency and performance. In the maximum compression test the GZIP program reduced the overall file size by 63% in 28 seconds. The most efficient compression engine reduced the file size by 80% but took 12 hours.
[10] Techincally GZIP was 16th fastest program however all of the faster programs were written for Windows. All of these faster alternatives were not well known, and concerns over interoperability and long term availability made them inferior choices at this time.
[11] As specified in RFC 1952, "The data format defined by this specification does not attempt to: - Provide random access to compressed data …"

2.1 How GZIP Compresses Data

The GZIP program compresses files using two types of lossless compression algorithms. The program uses LZ77 [67] to replace duplicate strings with a pointer and length indicator (see Figure 7). The program also uses Huffman codes [68] which are a byte coding system that uses shorter byte codes for frequently used characters (see Figure 8).

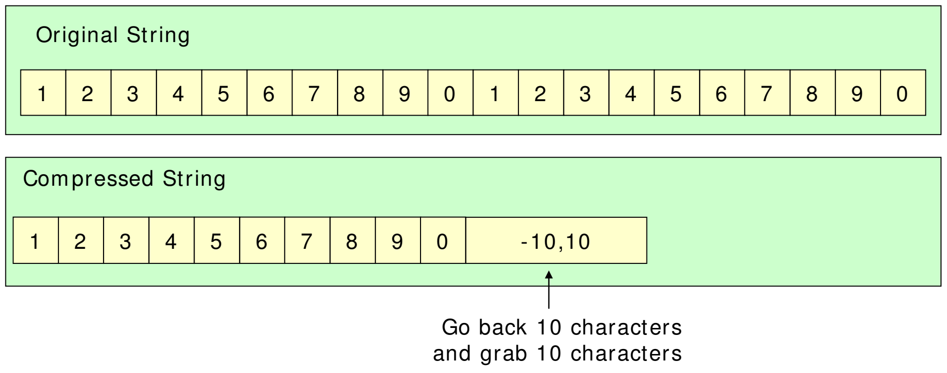


Figure 7: LZ77 compression.

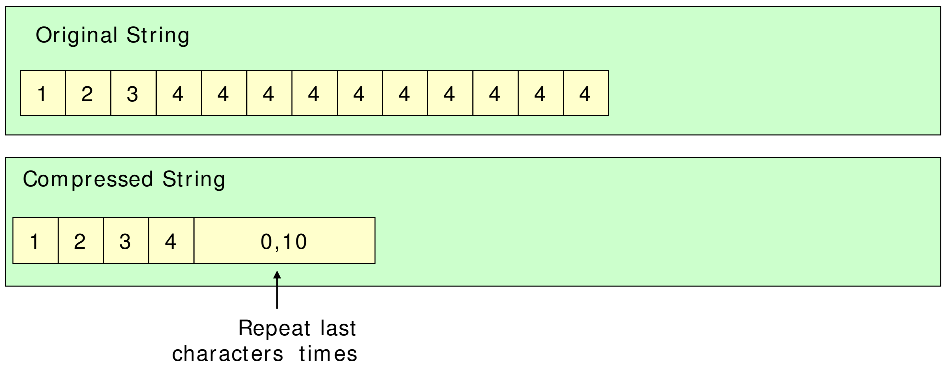


Figure 8: Variable length bytes encoding.
Huffman codes take advantage of the frequency of byte usage to compress data To save space frequently used characters are allocated shorter codes

The GZIP compression program uses a two pass system. During the first pass the algorithm measures the frequency of character usage and finds repeated character strings of length 3 or greater. The pattern search is limited to the length of the 32Kbyte window[12]. On top of the string matching, GZIP also looks for and eliminates repeated characters (like a series of 0's in a data stream) and replaces them with a single byte and quantity indicator.

During the second pass the GZIP compression program replaces the fixed sized characters codes with a character code that is based on a variable bit size (a Huffman code). In reality two Huffman trees are used. One Huffman trees has dual use; it contains both the original character codes as well as the match length indicators[13]. The second Huffman tree is used for the pointer distances.

The GZIP program stores data to disk in a series of compressed blocks. Each compression block can be a maximum of 64KB in size but can be terminated early using a special termination indicator. This GZIP program can vary its compression approach based on the type of data encountered. There are three types of compression blocks supported:

1. RAW DATA BLOCK TYPE: Blocks of incompressible data are stored in its raw form[14].

2. PRE-SHARED HUFFMAN BLOCK TYPE: Text based documents often has similar byte code usage statistics. [Characters like "a" and "e" are more frequently used than characters like "x" and "z".] Rather than repeatedly storing relatively similar Huffman tables prior to each compressed block, GZIP has a pre-shared Huffman tree that can be used by default to save space.

3. DYNAMIC HUFFMAN BLOCK TYPE: Some data is compressible but not all data looks like text. Data that is compressible but has a unique byte code distribution may not be efficiently stored using the pre-shared Huffman table. Based on the results of an efficiency calculation, GZIP will create a custom Huffman table and store it at the beginning of the compressed block.

As shown in Figure 9 each compression block is stored surrounded by a start and end. The start byte indicates the compression block type while the end byte contains a special end byte to mark the end of a compression block. Each GZIP file has an overall file header and trailer block. The header of the file includes a magic number and some global file information attributes, like the original file name and time stamp of the original file. The file trailer contains a CRC calculation to verify that the original data is decrypted properly.

[12] GZIP limits the search window size to ensure that the replacement codes are a reasonable length.
[13] During the decompression phase when GZIP finds a character code that maps to a length indicator it knows that a string substitution has been made and that the subsequent byte code is a distance pointer. If the character code maps to a literal it knows that the original byte can be reinserted.
[14] Encryption typically avoids storing repeated byte patterns and all byte codes are used on an even distribution. This means that encrypted data blocks resemble random byte patterns which are incompressible.

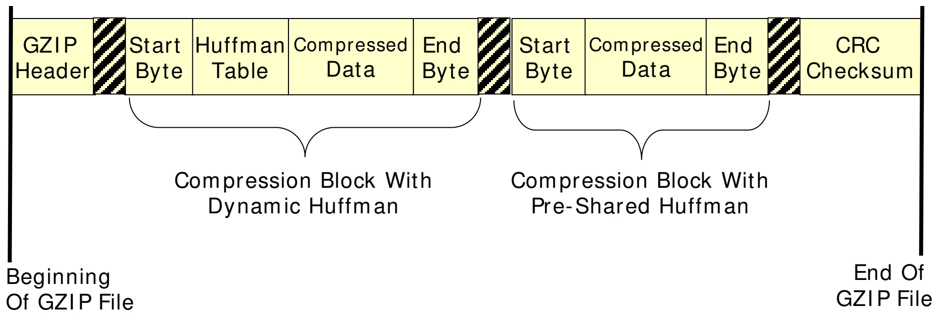


Figure 9: Example of a GZIP File

2.2 Summary of Changes Made to the Compression Algorithm

There are several obstacles to making the GZIP searchable namely:

a. All string matches need to be self contained in a given compressed block: In a standard compressed file string matches can occur within the previous 32 Kbytes of data. This may lead to a situation where one compressed block points to the previous compressed block which points to a previous compressed block and so on. The matching algorithm would need to be changed such that all matches would need to stay within the confines of the current compressed block.

b. Each compressed block would need to begin at a meaningful location within the underlying PCAP data file. This means that each compressed block needs to begin with the beginning of a packet record

c. The corollary to the previous statement is that each compression block must terminate at the end of a packet record. Since each compression block contains a series of variable length packets there needs to be a means of controlling when the block terminates.

d. The index used for the searchable compressed files must pre-exist in the PCAP data file in a pre-sorted form. In the case of a packet logger the timestamp acts as a natural index for the compressed file[15].

e. As the file is being compressed the indexing information needs to be saved for use later by the decompression algorithm. For a PCAP file the time index associated with each compression block needs to be stored in a companion “PIDX” file.

[15] The ability to index files based on time is useful because most security investigations are triggered by a security event which includes a time stamp.

f. Every PCAP file that is being compressed has a 24-byte header. This header needs to be stored in a separate compressed block at the beginning of the compressed file.

Based on the above issues, the following changes were made to the GZIP compression algorithm:

a. A command line option was added to indicate that a PCAP file is being compressed.

b. When the compression program is started a companion time index file (the PIDX file) is opened and before the program exits it needs to be closed.

c. During compression, the incoming data stream needs to be parsed before it is passed to the compression logic. This parsing logic interprets the incoming data stream, estimates the space left in the compression block, and sends a signal to trigger the closure of the current compression block. The parser is designed to terminate the first compression block at the end of the PCAP file header block. Subsequently, the parser ensures that the each compressed block is closed at the end of a packet record. The parser will also close a compressed block at the beginning of each second.

d. The string matching algorithm is limited to operating within the current compression block. This is done by resetting a series of pointers and hash tables at the end of each compression block eliminating its knowledge of the previous compression block.

e. The original GZIP program includes a compression efficiency test that will restart a compression block if performance is poor. This restart logic is disabled.

f. In order for the parsing logic to control the end of a compression block, it must withhold data from the compression engine. In some cases, it will appear to the compression logic as though the end of the file is reached (which triggers the compression block to be closed). However the side effect is that extra logic needs to be added to distinguish between the end of a truncated compression block and the end of a file.

g. Once the first compression phase is able to find available patterns and count byte code usage, it then passes the data to the second compression phase. During the second compression phase when the data is converted into variable length byte codes the total number of bits written to disk are tracked. The companion time index file uses this information to store the location of each compressed block in the file.

h. The time index is written to disk at the end of the second compression phase. The time index record contains the start time of the compression block and the bit location of the block. In the case of the very last compression block in the file, a special final record is added to the time index file to indicate the ending time span covered by the file.

The changes listed above modify the way in which string matches are made during the compression phase and changes when each compressed blocks terminate. However these changes do not break the GZIP file format. The modified compressed output file is still in a valid GZIP format and can be decompressed with either the standard GZIP program or the revised program.

2.3 Summary of Changes Made to the Decompression Algorithm

The original GZIP program could only decompress a file by starting at the beginning of the file and working forward to the end. The changes to the compression algorithm discussed in the previous section resulted in a program that writes a companion file that holds the location of each compressed block. During the decompression phase, the revised program will use the indexed file created during the compression phase to allow the program to jump directly to the region of interest, decompress a variable number of compressed blocks and then exit. A summary of the changes made to the decompression logic are described below:

a. An extra command line option is added to allow the user to specify the time range to be decompressed. The user enters both a start and end time in the YYYY/MM/DD@hh:mm:ss format. Both date strings need to be converted into an integer value describing the seconds from epoch so that they match the format of the PIDX file.

b. Prior to the start of the decompression the index file is searched and the bit location of the associated compression blocks is found. A counter containing the number of blocks to decompress is also set.

c. The program begins to decompress the file. The GZIP header is read and the first compression block is read which contains the PCAP file header block[16]. The standard decompression algorithm is intercepted at this point.

d. When the end of the PCAP header block is reached the disk buffer is flushed and the program jumps ahead in the file to the beginning of the targeted compressed block

e. The decompression function is re-engaged and executes until the requested number of compressed blocks are read.

f. Once the end of the last block is reached the program prematurely aborts closing the compressed file circumventing the CRC check, and releasing allocated variables on exit.

[16] The recovered data file must always be prefaced with the PCAP header to make it readable to an external program like tcpdump.

2.4 Performance Results

The revised compressing algorithm, PZIP (packet zip) is slightly faster than GZIP in both the compression and decompression phase for the Pentium 4 architecture. Sample results for an actual network traffic sample are shown in Table 4. The table shows the amount of time that is required to compress an hour of packet traffic and then decompress a select time range surrounding a network event. During the decompression phase a second rather than a complete hour is being decompressed. As shown the performance gains were significant. The performance gains in the compression phase come as a result of the restrictions to the match window length. Since all matching strings must occur within the current compressed block the total time spent finding suitable match is reduced. However, this also means that sub-optimal matches are being made which increases the total file size slightly as shown in and Table 5. In addition to support, the time indexing feature a companion time index file must be stored with the file adding to the storage space requirements. Overall the total loss of compression efficiency is only a few percent.

Table 4: Algorithm Performance Pentium 4 Architecture

ALGORITHM	COMPRESSION	DECOMPRESSION
Standard GZIP Compression	39sec	6 sec
PZIP Compression (Modified GZIP)	32sec	<1sec

Table 5: Algorithm Compression Efficiency (header snaplen 64)

ALGORITHM	FILE SIZE
Original File	241MB
Standard GZIP Compression	111MB
PZIP Compression (Modified GZIP)	114MB + 0.037MB (Size of time index file)

With respect to the XEON architecture the performance results were fairly similar as is shown in Table 6 and Table 7. The time associated with a query for the large PZIP file was still very fast. Tests showed that an ICMP traffic query from a second in the middle of the 14.7GB compressed file could be recovered in 0.182 seconds (whereas decompressing the full file takes several minutes).

Table 6: Disk Based File Compression Performance Xeon Architecture

ALGORITHM	COMPRESSION SPEED	DECOMPRESSION SPEED
Original File	N/A	N/A
Standard GZIP Compression	77 sec	15.9 sec
Fast GZIP (-1 option)	44.6 sec	15.9 sec
PZIP Regular (Modified GZIP)	67 sec	15.4 sec
PZIP Fast (-1 option)	41.4 sec	15.8 sec

Table 7: Algorithm Compression Efficiency (header snaplen 150)

ALGORITHM	FILE SIZE
Original File	24.6 GB
Standard GZIP Compression	13.3 GB
Fast PZIP Compression (-1 option)	14.7 GB + 0.003GB (Size of time index file)

2.5 Future Directions

Another way to boost performance of a packet logger involves better leveraging new multiprocessor CPU developments. Packet loggers that compress data need to find and eliminate repeated byte patterns in the incoming data stream. Researchers have demonstrated that the cell processor can be used to boost pattern matching performance of an IDS [69][17].As shown in Figure 6(c) the cell processor model uses a hierarchal multiprocessor model where a powerful central CPU offloads pattern matching work onto a series of slave CPU's.

[17] Although this study is applied to the pattern matching capabilities of the Snort IDS the process of matching patterns could be realistically be reused for compression.

3 Summarizing Network Activity

The ability to selectively decompress a file based on a time range as presented in the previous section is a useful; however it is a somewhat incomplete solution. When security analysts begin investigating a security event they will know that an event has occurred at time “Y” but will not know when it started and when it finished. Rather than forcing an analyst to guess a time range, there needs to be a way of identifying when an IP address is active.

The IP indexing tool discussed in this section is able to both capture and report IP based activity data. As a stand-alone package the tool can be run in parallel to the packet logger to characterize network activity. It is also capable of capturing and presenting gross network statistics that are useful in assessing some types of security events. In subsequent sections this summarization capability will be combined with the improved compression to create a unified data collector for traffic monitoring.

3.1 Techniques for Summarizing Packet Activity

There are several technical approaches available for summarizing packet activity as shown in Figure 10. One form of packet summarization involves describing the total amount of activity present on the network. There are several tools that are capable of measuring total network activity. A popular open source tool called NTOP is able to summarize network activity and bandwidth usage at a single point on a network [70] and a survey of similar tools is available in [73]. In large enterprises a Network Management System (NMS) is often used to monitor the state of the network and hold detailed link usage statistics [74-76]. Link usage has some limited value to a security analyst to help verify worm or denial of service activity.

Another approach to summarizing IP activity involves combining the low level activity with some other data source. As an example Packet Sentry is a product that associates packet activity with end user actions [77]. It does this by listening to packet exchange data and augmenting the information with data received from the Windows LDAP servers. This allows the product to identify who is manipulating databases and file shares.

Another approach involves watching the behaviour of the network over time and using changes in activity pattern to detect security threats. Network Behavoral Analysis Systems like: Mazu-Profiler, Q1 Labs – Qradar, and Lanscope - StealthWatch fall into this category [78-80]. These tools are most valuable in detecting worms and some types of zero day events[18].

[18] Some of these behavioural analysis systems also leverage flow information however they have a much broader scope.

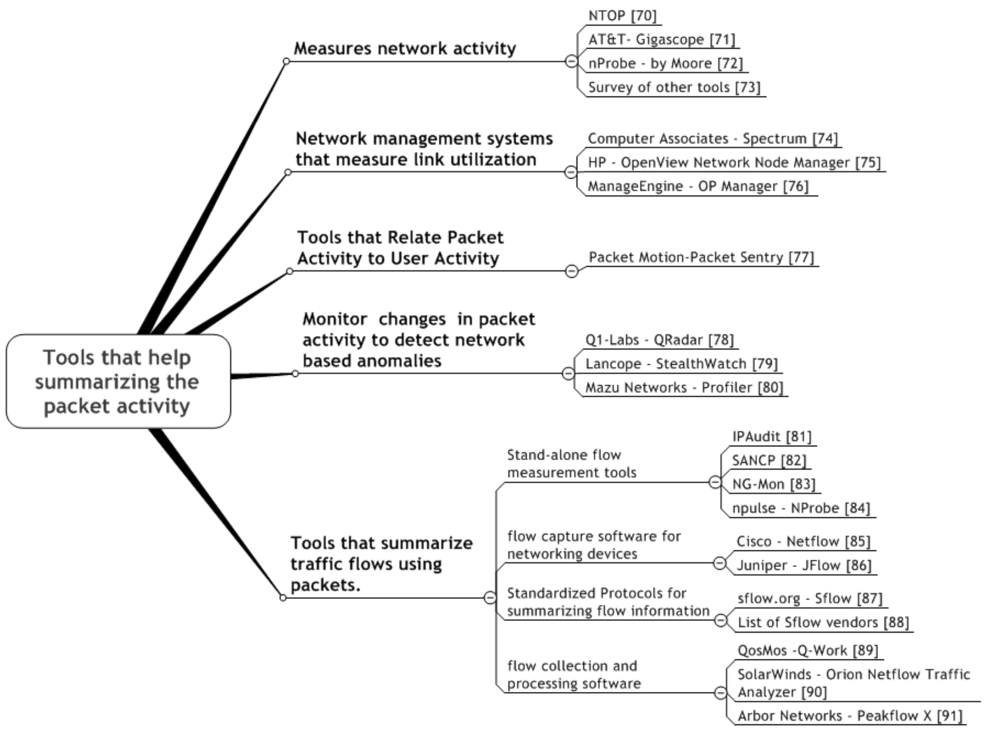

Tools that help summarizing the packet activity
Measures network activity
NTOP [70]
AT&T- Gigascope [71]
nProbe - by Moore [72]
Survey of other tools [73]
Network management systems that measure link utilization
Computer Associates - Spectrum [74]
HP - OpenView Network Node Manager [75]
ManageEngine - OP Manager [76]
Tools that Relate Packet Activity to User Activity
Packet Motion-Packet Sentry [77]
Monitor changes in packet activity to detect network based anomalies
Q1-Labs - QRadar [78]
Lancope - StealthWatch [79]
Mazu Networks - Profiler [80]
Tools that summarize traffic flows using packets.
Stand-alone flow measurement tools
IPAudit [81]
SANCP [82]
NG-Mon [83]
npulse - NProbe [84]
flow capture software for networking devices
Cisco - Netflow [85]
Juniper - JFlow [86]
Standardized Protocols for summarizing flow information
sflow.org - Sflow [87]
List of Sflow vendors [88]
flow collection and processing software
QosMos -Q-Work [89]
SolarWinds - Orion Netflow Traffic Analyzer [90]
Arbor Networks - Peakflow X [91]

Figure 10: Summarizing the networks activity and state.

The last approach involves creating a summarized view of the packet activity. There are several ways to summarize packet activity. One approach involves summarizing packet data on a session or flow basis. Data summarized using this approach typically results in reports that resembles a telephone long distance bill. There are two common flow collection architectures used within organizations: stand alone and distributed flow collection which are shown in Figure 11.

Stand alone applications are useful for knowing the activity at a single point in the network and are often used at critical network locations such as Internet gateways or the network backbone. There are a number of stand alone open source software tools like IPAudit [81] and SANCP [82] that are intended to be used in lower bandwidth stand alone applications. For higher bandwidth applications NGMon [83] and nProbe [84] are potential options.

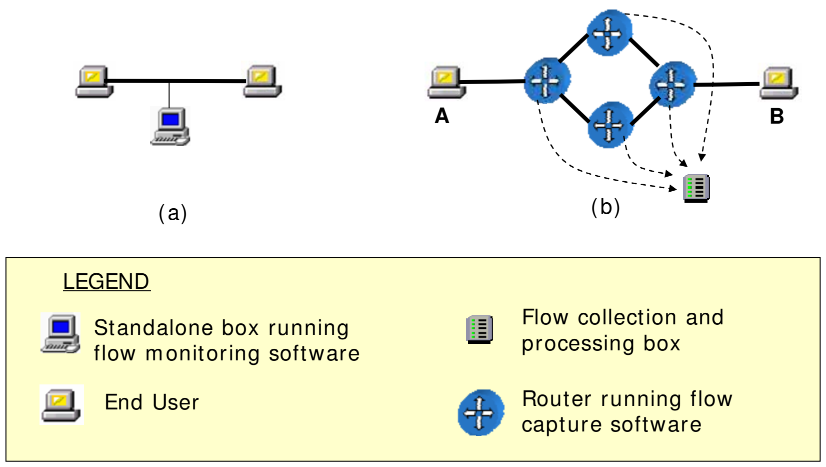

A
B
(a)
(b)
LEGEND
Standalone box running flow monitoring software
Flow collection and processing box
End User
Router running flow capture software

Figure 11: Flow Monitoring Architectures.

(a) Stand-alone flow collection (b)Distributed flow collection

In terms of distributed flow monitoring there are a number of approaches. One of the most popular approaches involves using the networking infrastructure to collect flow data. Cisco-Netflow [85], and Juniper-JFlow [86] are examples of flow software for a particular vendor's networking devices. There is also a more vendor neutral flow approach backed by Hewlett Packard called Sflow [87]. There are a number of vendors which have implemented interfaces for the Sflow standard which are listed in [91].

Each of these flow formats is fairly similar to one another on a feature and functionality basis. To understand the general advantages/disadvantages of the flow based approach one of these formats will be analysed in depth. Cisco-Netflow is configurable with 83 recoverable data types [92]. It handles in excess of 8900 packets per second and can export about 1000 records/sec to a data collector [93]. However the Netflow approach does add a significant load on the router's CPU. This load may be reduced by sampling 1 out of every 100 packets or 1 out of 1000 packets. Another significant prerequsite of the Netflow approach is that it assumes that the security analyst has access or influence over the router's configuration which is not always true in an organization

The data that is collected on the router needs to be moved to a post processing device for analysis and presentation. There are several products available for processing flow information such as [89-91].

The IP_INDEX tool presented in this study operates slightly differently from the flow measurement tools described above. The tool characterizes endpoint activity - it captures how much data entered and left a given IP address, the number of IP addresses the end node communicated with and the number of sessions observed on the link. The tool was created to review backbone traffic where one of the addresses is typically that of the NAT translation box. Although this approach is less informative for some types of security investigations the intent here was to create a summarization tool that would act as a gateway into the raw packet data itself. Relative to other solutions, the tool has a very compact record size which saves disk space and makes the data more portable.

In addition to the IP summarization capabilities the tool also collects a number of gross network statistics. It tracks the top talkers/listeners on the network and measures the total volume of traffic exchanged on a link. This information can be stored at fixed intervals and recovered later for review or presentation in graph form. Although IP_INDEX tool is a stand-alone tool it can be queried in a distributed fashion through the packet recovery tools discussed in Section 5.

Some readers may question what the benefit of having a "rough outline of an IP's activities" is. Depending on the type of event, an outline can identify whether an event requires further investigation. As an example, if a scan is reported, the report generated by the tool will describe whether the scan succeeded in recovering any information (bytes received by attacker = 0). Perhaps instead, there is some type of suspicious activity detected. By reviewing past activity associated with this IP, insight might be gained as to whether this is a new user or a regular user of the network. If the user is new to the network and his packets are suspicious, then it may be prudent to review the activity with a higher level of scrutiny.

3.2 IP Index

The IP_INDEX program tracks the amount of activity associated with each IP address transmitting/receiving data on the network. Most critically the program retains a time span describing when the IP is active. The time span is used as a reference for querying the raw packet data. In addition to the time span several other measurements are stored to help describe the activity surrounding an IP address including:

1. Total number of conversations or sessions the IP address was involved in.

2. The number of bytes transmitted and received

3. The number of packets transmitted and received

4. The number of unique IP addresses that packets were exchanged with.

5. The commonly used ports that were used. (There are 32 user settable ports/protocols that can be set).

Each data record describing an IP address is 16 bytes long and is described in detail in Table 8. Due to the tight information packing, some of the data is highly summarized. Although the tool is less precise and more feature rich than SANCP, NTOP, or Netflow it is only a front end for the raw data. If the analyst needs more information or precise details the information can be obtained through analysis of the raw packets.

Table 8 IP indexing features

DATA TYPE	SIZE (BYTES)	IMPLEMENTATION NOTES
IP address	4	The source and target IP address in a packet form two records.
Start Time	1	This field indicates the time the activity started for this IP address for the collection window. The actual value is rounded down to the nearest second[1]. To save space the time is relative to the start time of this collection cycle.
End Time	1	This field indicates the time the activity ended for this IP address during this collection window. The actual value is rounded up to the nearest second.
Bytes Transmitted	1.5	The total number of bytes transmitted by this IP address during the collection window. This field uses a variable round off technique. The field is 12 bits wide the first 2 bits holds a scaling indicator (Bytes,kB,MB,GB). The remaining 10 bytes contain an integer of the value at that scaling level. As an example a rarely used traffic source might reported as transmitting 900 bytes. Whereas a major traffic source might be reported as transmitting 2GB. (In terms of doing assessments it is the relative magnitude that is most useful – knowing a machine transmitted 2GB.
Bytes Received	1.5	The total number of bytes transmitted by this IP address during the collection window. This field uses a variable round off technique similar to that described for the bytes transmitted.
Packet Transmitted	1.5	The total number of packets transmitted by this IP address during the collection window. This field uses a variable round off technique similar to that described for the bytes transmitted.
Packet Received	1.5	The total number of packets received by this IP address during the collection window. This field uses a variable round off technique similar to that described for the bytes transmitted.
Communication Sessions	1.5	This field is a VERY approximate measure of session activity. This field counts the number of TCP SYN-ACK packets and UDP packets that appear after a 5 second period of silence[2]. The width of this field is 12 wide and the bits are formatted in the same manner as the bytes transmitted field.
Host to Host Connections	0.5	This field describes the number of different targets with which a computer communicated. The width of this field is very small and activity >7 is reported as >=7 session for a collection window.
Port Activity of Common Ports	2	When the IP index program is set up the user indicates a list of common ports/protocols to track. Each flag is assigned a prime number. Each unique prime for a port is multiplied together to form a product. If there are a large number of ports active on a device then the product will exceed a 32767 (2^16-1) a overflow value (0) is stored and a message "MANY PORTS IN USE" will appear in subsequent ports. If the packets that are exchanged exclude any user specified ports then the initial value of (1) is stored. More details associated with the packet can be recovered from the raw data.
RECORD SIZE	16	

[1]Note the start time and end times are not session based. The intent here is provides guidance as to where activity is occurring not exact quantities. Tools like SANCP give much more precise information but it comes at the expense of file size and real time performance.

[2]Obviously this is a inaccurate measure of session activity because TCP on a slow link will issue often issue multiple SYN-ACK packets and UDP/ICMP session activity could be clustered packets.

When the IP_INDEX program is acquiring packet data, the information received is placed in a table in memory. Each row of this table describes a unique IP address. Over time, the table fills up as new IP addresses are encountered. The table is flushed after a user specified interval or when the user defined maximum table size is reached. Laboratory experiments have shown that smaller tables fill up quickly but provide better activity granularity. Since each table contains repeated information larger tables are saved less frequently to disk and consume less space. The best values are highly network dependent. The program was applied on two different enterprise networks and the best table size varied significantly between the two. Interestingly the smaller network required the larger table size because of the nature of their work activities and network structure. The point being made is that the default table size and flush rate need to be verified before the tool is used on a production basis.

Although each 16-byte record IP address record is quite small, there are many addresses used on a large network. The total volume of disk space consumed by these files is significant. In the test environment the IP index files were approximately 12% of the size of the associated compressed packet capture file. This figure is highly situation dependent and will vary with IP index table size, network architecture and packet capture size. These files however are highly compressible and future releases could examine different options for compressing the index.

In addition to the recording the activity surrounding an IP address the IP_INDEX program also measures some gross network statistics. A summary of the statistics captured are provided below:

1. The top 100 packet transmitters and receivers

2. The top 100 byte transmitters and receivers

3. The total bytes transmitted on each commonly used port

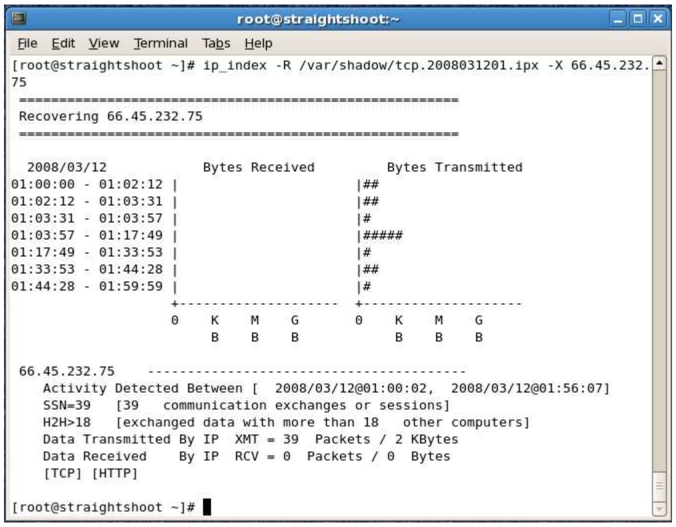
4. The total bytes received on each commonly used port.

5. Total bytes captured overall

The IP_INDEX program offers numerous command line options that allow the tool itself to be used in a variety of ways. The program can read packets directly from the network interface card or from a pre-existing PCAP files. The IPX output files generated by the tool can be stored in a single continuous file or broken up on an hourly basis which allows it to be nicely integrated with the hourly packet logs from the Shadow IDS.

The program was intended to be used on a real time basis. The program is multithreaded such that the data capture, table creation, and table storage activities fall on separate threads forming a simple assembly line and allowing the program to operate efficiently in real time.

The IP_INDEX tool generates reports in a simple text based output. A sample report for a single IP address is shown in Figure 12. The graph uses a text based bar graph to show the volume of network activity. Since the tool is IP based rather than session based the tool is limited in its capacity to isolate session based data. The tool however will allow the user to identify when two IP addresses are simultaneously active. In addition the tool also allows the user to query on a subnet basis.



```
[root@straightshoot ~]# ip_index -R /var/shadow/tcp.2008031201.ipx -X 66.45.232.
75
 =======================================================
 Recovering 66.45.232.75
 =======================================================

  2008/03/12           Bytes Received         Bytes Transmitted
01:00:00 - 01:02:12 |                      |##
01:02:12 - 01:03:31 |                      |##
01:03:31 - 01:03:57 |                      |#
01:03:57 - 01:17:49 |                      |#####
01:17:49 - 01:33:53 |                      |#
01:33:53 - 01:44:28 |                      |##
01:44:28 - 01:59:59 |                      |#
                    +--------------------  +--------------------
                    0    K    M    G       0    K    M    G
                         B    B    B            B    B    B

 66.45.232.75    ----------------------------------------
    Activity Detected Between [  2008/03/12@01:00:02,  2008/03/12@01:56:07]
    SSN=39   [39   communication exchanges or sessions]
    H2H>18   [exchanged data with more than 18   other computers]
    Data Transmitted By IP  XMT = 39  Packets / 2 KBytes
    Data Received    By IP  RCV = 0  Packets / 0  Bytes
    [TCP] [HTTP]

[root@straightshoot ~]# █
```

Figure 12: Sample text output from IP Index program.

3.3 Performance Results

In order to evaluate the IP_INDEX tool two different types of tests were performed. In the first test tcpreplay [94] was used as a packet generator to provide a sustained traffic load. The load was increased until a point was reached where packets began to get dropped. In this case the IP index could handle 150 Mbps of traffic without losing data on a 64 byte XEON 2.5GHz dual quad server.

Table 9 Maximum Program Throughput

PROGRAM	MAXIMUM THROUGHPUT
IP Index	160Mpps

Benchmaking was performed using a Dell 2900 Dual Quad Core Xeon (2.5 GHz)

In the second set of tests the IP_INDEX tool was compared to SANCP which is the closest comparable flow measurement tool. The results from the second test are shown in Table 10. As can be seen the text based files generated by SANCP were much less space efficient than the packed binary files. In addition the IP indexing software runs much faster than SANCP. The performance difference is attributed to the use of multithreaded code and indexing based on IP address rather than flow.

Table 10 Comparing Performance of IP_INDEX versus SANCP

ALGORITHM	FILE SIZE	TIME
Original PCAP File (INPUT FILE)	2.7 GB	N/A
SANCP Index Size	213 MB	308 sec
IP_INDEX	25MB	27 sec

Benchmaking was performed using a Dell 2900 Dual Quad Core Xeon (2.5 GHz)

4 Combining Packet Logging and Traffic Summarization into a Single Real Time Hourly Packet Logger

At this point in the discussion a tool to index compressed packet capture files has been presented as well as a tool to summarize IP address activity. The ability to actually log the packets on a real time basis still needs to be addressed. The original Shadow packet logger uses a combination of tcpdump and gzip to log packets to disk. The data was exchanged between these two applications using the Linux file redirection capabilities. Although this approach is workable there are several problems associated with it namely:

- Running two separate applications is slower than using a single combined application,
- At the end of the hourly collection window the packet logging process would need to be restarted. During the rollover no packets would be collected and a gap in the data collection would exist.
- During the current hourly collection window the data was unrecoverable. This means that security analysts could not investigate events until the beginning of the next hourly collection window.

In this section a new tool is presented that combines the packet logging, compression, and IP summarization into a single tool called TCPZIP.

4.1 TCPZIP

The architecture diagram of the TCPZIP program is shown in Figure 13. Tcpdump, PZIP and IP_INDEX exist within the program as separate threads and are linked together through a series of circular buffers[19]. The TCPZIP block at the top of the diagram parses the user's input and sets up the other threads. The tcpdump thread collects packet from the network interface and pushes the packets into a circular buffer. The pzip thread reads the packets from the circular buffer, processes them and then stores the compressed packet data and time index information into two output buffers. The IP index thread shares the same input buffer as pzip but sends its result to a different output buffer.

TCPZIP is intended for capturing packets from either an existing PCAP file or from the network interface card and saving it to disk in a compressed form. Two compression modes are supported the standard GZIP format and the PZIP time-indexed format. The program supports the use of Berkeley packet filters which means that the program can be used for selective packet capture. The program which was designed as a dedicated packet logger cannot print packets like tcpdump and unlike GZIP it does not decompress the files it creates.

[19] A circular buffer is a type of first-in first-out (FIFO) buffer that has a fixed length. The read and write pointers indexing the buffer shift as data gets written into and read out of the buffer. The buffer is called circular because the when either pointer reaches the end of the buffer it is reset to the beginning of the buffer.

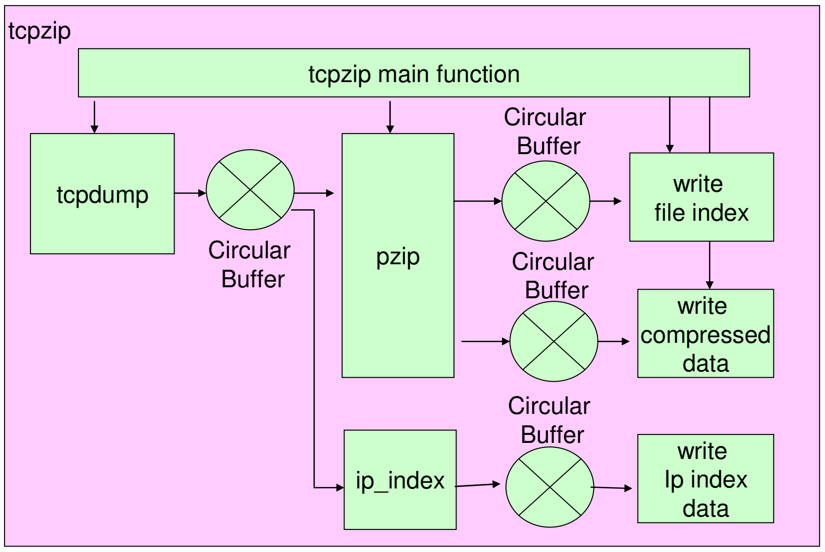


Figure 13: TCPZIP program architecure.

The program can log packets in several different modes namely:

- Continuous and Unlimited Capture - The program will collect packets until a stop signal is generated. A stop signal can be generated by sensing a SIGINT signal (generated by typing control-C) or a SIGTERM signal (generated with the kill(1) command).
- Limited Duration Capture – The program can be used to collect a fixed number of packets and then stop automatically.
- Hourly Capture – The program stores packets in an hourly capture file. At the end of each hour the PCAP file is closed and a new file is started with an updated PCAP header block. During each hourly rollover the PZIP and IP indexing threads are restarted but the tcpdump thread is left running. During the thread restart the tcpdump thread continues to store the packets into the circular buffer. The benefit here is that packets are no longer being lost over the hourly collection window.

4.2 Performance Results

The throughput results from TCPZIP are shown in Table 11. As shown in the table when either IP_indexing is run or PZIP compression is run, the performance is fairly close to that of tcpdump. However when both IP_indexing and compression are simultaneously performed the performance of TCPZIP drops significantly to about 60 Mbps. The reason for this dramatic drop is not well understood, at this time. As a consequence this means that in high volume applications two servers should be used in place of one if sustained throughput rates exceed 60Mbps. The first server would be used to compress the data while the second server would generate the IP_INDEX file.

Table 11 Disk Based File Compression Throughput Performance Xeon Architecture

ALGORITHM	MAXIMUM THROUGHPUT
tcpdump	160 Mbps
TCPZIP with IP indexing **enabled**	60 Mbps
TCPZIP with IP indexing **disabled**	160 Mbps
IP_INDEX	150 Mbps

Sample disk based file compression results are shown in Table 12. As can be seen the results are comparable to the original GZIP program however it has the added benefit of the time and IP indexing capabilities.

Table 12 Disk Based File Compression Performance Xeon Architecture

ALGORITHM	FILE SIZE	COMPRESSION SPEED	DECOMPRESSION SPEED
Original File	834MB	N/A	N/A
Standard GZIP Compression	479MB	77 sec	15.9 sec
Fast GZIP Compression	555MB	44.6 sec	15.9 sec
TCPZIP Compression (With IP Indexing)	557MB [555MB + 1.7MB (IP index) + 0.1 MB (time index)]	46.1 sec	N/A (Refer to pzip)
TCPZIP Compression (No IP Indexing)	555MB [555MB + 0.1 MB (time index)]	43.2 sec	N/A (Refer to pzip)

5 Distributed Packet Data Recovery

This study has demonstrated an improved packet logger using off the shelf computing hardware. It could be deployed as a single data collector or as part of a distributed group of sensors across an enterprise network. The analysts that are querying the distributed system are often remotely located and communicate with the sensor through bandwidth-constrained links. When a security event is detected, the analyst could directly log into the appropriate packet logger and recover the associated data. The issue with this solution is that it is not scalable or convenient. It is inconvenient because distributed events require that the analysts log into multiple packet loggers and work with multiple traces. It is not scalable because there is going to be a significant number of analysts and software based agents querying the traffic data[20]. The combination of query volume and filtering demands will place a significant workload on any single machine. Each data query that is issued needs to be executed in a way that minimizes the load on the packet logger.

Currently, the Shadow IDS includes scripts that help distribute the packet analysis workload across multiple computers as shown in Figure 14. Real time packet collection is performed on a box called the Shadow Sensor while data queries and analysis is performed on a Shadow Analyst station. There can be multiple analyst stations each of which obtains its data by securely copying data directly from the sensor or another analyst station. Unfortunately, this data transfer can only be performed after the file is closed at the end of the hourly data collection period. Recovered data can only be held for a finite period of time because of disk space limitations. Some sensor/analyst stations are intended to retain data much longer than others. As a result, any improved packet recovery system for the Shadow IDS needs to direct queries to a resource that covers the requested time period. To keep things simple, queries involving recent events should be handled by a short term analyzer while longer term investigations would be handled by a server with longer data retention capabilities.

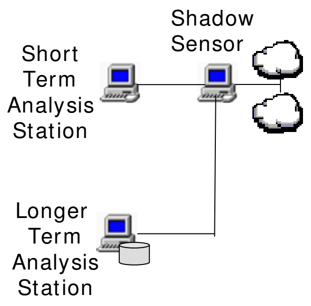


Figure 14: Shadow IDS Tiered Architecture
The Shadow IDS performs hourly packet captures using a packet logger called a "sensor". After the hourly rollover the data gathered by the sensor is copied to one or more analysis stations for review.

[20] Collected data may need to be reviewed when new software vulnerabilities are announced, when non-real time scripts are processed. If a compromised computer is discovered and a long term forensic analysis is required the analysis can be performed without bogging down the rest of the analysis work.

In summary, the new fast packet recovery software discussed in previous sections offers new and enhanced features that allow it to be operated across a large distributed network. What the software lacks is a centralized operator interface that recovers the data for the analyst. In this section a new tool will be discussed to allow the security analyst to simply enter the data associated with a security event and have the data recovered. The tool is designed to automatically recover current data directly from the packet logger itself, recent data from a short term analysis station, and longer term data directly from the data archive. When data is split across multiple data sources, the data will be sought out from each source separately and merged into a single consolidated output. The recovered data will be downloaded in a form that is suitable for advanced packet analysis using tools like tcpdump or wireshark.

5.1 Packet Fetch

Packet Fetch is the name of the tool written to remotely recover packet data from the packet logger and analysis stations. The program is able to receive input from the user either directly from the command line or through a simple text based user interface. Based on the user input the program connects to the appropriate server, filters the data at the remote end and then transfers the data to the analyst's workstation.

The packet fetch software is able to interact with one or more data sources and supports several packet logging configurations as shown in Figure 15. A summary of these packet recovery configurations are listed below.

a. 1 standalone Shadow Sensor[21] (support excludes current data collection window)

b. 1 standalone TCPZIP sensor

c. 1 TCPZIP sensor that offloads its data to 1 or more analysis stations each hour.

d. 2 Shadow sensors or TCPZIP sensors in a load balanced configuration or in a serial chain

e. 2 Shadow sensors or TCPZIP sensors that offloads its data to 2 or more analysis stations each hour[22]

.

[21] This option allows analysts to slowly transition from the old Shadow software to the new system incrementally over time.

[22] The software can connect to 4 different servers to obtain information. The key here is that the software chooses the most appropriate 4 servers based on the time range requested.

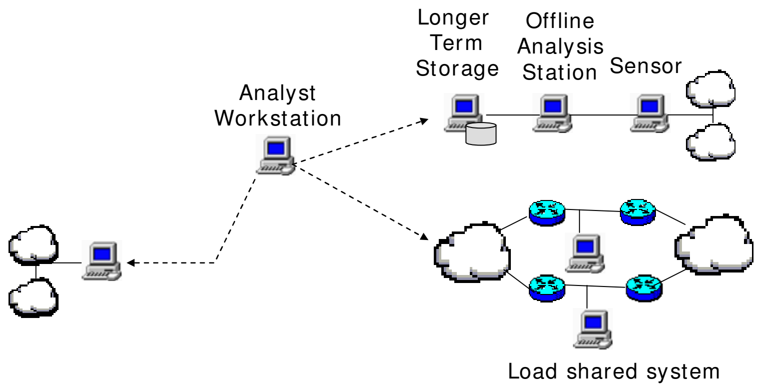


Figure 15: Example of Supported Packet Fetch Configurations.

At the beginning of a typical investigation the security analyst receives an IDS alert that requires detailed investigation. Although the analyst knows the addresses, time, and location of the event the analyst still needs to determine when the suspicious activity started and ended. The packet fetch program can be used to leverage the capabilities of the IP indexing tool to determine the bounds of the packet query. In reviewing the results of this query, the analyst will have the opportunity to review the activity surrounding the attacker's (or victim's) IP address. The report that is generated will summarize the type of activity that was occurring around the event and give the analyst a sense of what time range needs to be recovered and the volume of data that is to be expected[23].

Based on the IP index report the analyst can determine a suitable time range over which to recover packets. The analyst can recover all packets in this range however in most cases the analyst will set a Berkeley packet filter to limit the packets recovered. Based on the time range requested and the IDS location, the Packet Fetch software determines where the data is currently at rest and automatically retrieves the data from the least processing intensive point in the packet logging infrastructure. For example the program will load late breaking events directly from the packet logger. It will download the recent data events from the main shadow analysis box. Finally historical data pulls are made from the long term storage array. In a load balanced situation if two streams of data need to be merged the data from both sources are recovered.

The packets that are recovered using Packet Fetch are always saved to disk under the name supplied by the user and if requested the program will display the results on the screen in tcpdump format. The recovered PCAP file can be subsequently imported into a more advanced packet analysis tool for detailed analysis.

[23] This summary report may also contain enough information to close the security events without downloading any packets. If the analyst is investigating a known worm attack but the victim of the attack ignores the malicious packets then the incident may be closed without further investigation.

The packet fetch program is written in C. Input supplied by the user is used by the program to write a custom Expect [95] script. When the analyst submits a query the Expect script is saved to disk and executed. If the analyst wants to repeat the query later, all the analyst needs to do is run the script again. If the analyst wants another analyst to look at the packets the analyst can send the script rather than the entire packet trace. The script acts as part of the forensic record of actions taken during the course of an investigation. The script contains the exact list of commands that was used to recover the data and a timestamp is embedded in the script's filename.

5.2 Performance Results

A summary of the performance results for the Packet Fetch program is shown in Table 13. The Packet Fetch program is able to recover packets much faster than scripts that download a file and filter the results locally. It is also much easier to use and faster than requiring an analyst to manually log into multiple machines and entering the commands directly.

Table 13 Data recovery results on an 11GB compressed file (XEON dual quad core server operating under a 40Mbps sustained traffic capture load)

ALGORITHM	RECOVERY TIME
Search compressed file using pzip / tcpdump through packet fetch (4 minute pull)	24 sec
Search **uncompressed** file using tcpdump	306 sec
Search compressed file using gzip / tcpdump through packet fetch	355 sec
Time to transfer 1 hour packet capture file to analysis station	2358 sec

6 Discussion and Conclusion

The original Shadow IDS is an open source packet logger that has some intrusion detection scripts embedded within it. The packet logging portion of the tool is able to save compressed hourly packet capture files in a GZIP format which made it unique in the open source community. The hourly compressed files made the data easier to handle but data recovery was slow and inefficient. This study has modified the standard GZIP program to allow data to be selectively recovered from a compressed file. The modifications have reduced compression efficiency slightly but have enhanced the rate at which data is recovered. Since 1 second worth of captured traffic can now be recovered without the need to recover the entire hour, the performance gain is enormous. In addition, the changes made to the original algorithm do not compromise the GZIP file format, meaning that the revised file can still be decompressed using the standard GZIP program.

The IP_INDEX tool developed in this study provides an endpoint summary of IP activity. This summary provides the analyst with a means of surveying network activity in a way that is different from that of a flow based approach. The small well packed records are more space efficient and can be processed faster than flow based data. The indexed files themselves are structured to allow quick internal IP address searches which allows for longer term activity searches to be efficiently performed.

The TCPZIP tool developed as part of this study eliminated the loss of packet data that occurred during the hourly rollover for the original Shadow IDS. The TCPZIP application also provides packet logging, IP summarization, and bandwidth usage measurements into a single application. Unfortunately, when the TCPZIP program is run with both the PZIP and IP_INDEX features enabled, the overall performance is slower than expected. However when these features are run on separate computers, the overall performance increases to a level comparable to tcpdump.

The Packet Fetch program developed as part of this study allows analysts to quickly and efficiently recover remotely stored data from across the network. Embedded within the program is an awareness of how information is distributed across the network; this allows Packet Fetch to obtain the data from a storage source in a way that will minimally impact the overall system performance. The program is able to selectively recover data, saving bandwidth and time. The program is also capable of merging multiple data streams into a single PCAP file for analysis. In addition to recovering packets the program is able to extract IP activity reports, network usage and list the top talk/listeners on the network.

In summary, the Fast Packet Recovery project allows the analyst to find and recover the packets related to a security investigation quickly and easily. The next phase of this project will focus on the analysis side of the packet recovery. Packet data in its raw form is difficult to interpret and most commercial packet loggers offer some type of packet analysis software. Since all packet loggers are able to capture packets, most vendors try to distinguish themselves using the packet analysis software. At present, the Shadow IDS has only a weak traffic analysis capability. Currently, the recovered packets must be examined using tcpdump or wireshark. During the next phase of the fast packet recovery project the development of more advanced packet analysis software will be pursued.

References

[1] Bivio (2008) Product Overview http://www.bivio.net/products/ (Accessed date 23 April 2008)

[2] CloudShield (2008) Cloud Shield http://www.cloudshield.com/ (Accessed date 23 April 2008)

[3] Niksun (2008) NIKSUN NetTrident 2005 http://www.niksun.com/Products_Trident.htm (Accessed date 23 April 2008)

[4] IP Fabrics (2008) Double Expresso Datasheet, http://www.ipfabrics.com/pdf/Double_Espresso_Datasheet.pdf (Access date 22 June 2008)

[5] Endace (2008) Endace http://www.endace.com/ (Accessed date 23 April 2008)

[6] Napatech (2008) Products http://www.napatech.com/Products (Accessed date 23 April 2008).

[7] Npulse Networks (2008) Npulse: The Network Sensor Company http://www.npulsenetworks.com/ (Accessed date 23 April 2008)

[8] Datacom Systems (2008) Packet Filetering Products http://www.datacomsystems.ca/products/?c=4 (Accessed date 23 April 2008)

[9] Network Instruments (2008) GigaStor http://www.networkinstruments.com/products/gigabit/GigaStorProbe.html (Accessed date 23 April 2008)

[10] NetScout (2008) Continuous Packet Capture nGenius® InfiniStream™ http://www.netscout.com/products/infinistream.asp (Accessed date 23 April 2008)

[11] Solera Networks (2008) Capture Appliances, http://www.soleranetworks.com/products/capture-appliances.php (Accessed date 23 April 2008)

[12] Fluke (2008) Optiview Network Analyzer http://www.flukenetworks.com/fnet/en-ca/products/OptiView+Series+III+Integrated+Network+Analyzer/ (Accessed 20 June 2008)

[13] IP Fabrics (2008) IP Fabrics System and Software Products http://www.ipfabrics.com/products/index.php (Access date 24 June 2008)

[14] Raytheon (2008) SureView http://www.raytheon.com/businesses/rtnwcm/groups/iis/documents/content/rtn_iis_sureview_datasheet.pdf (Accessed date 20 June 2008)

[15] Javvin (2008) Javvin- Network Packet Analyzer CAPSA 6.7 http://www.javvin.com/product/Packet/PacketAnalyzerDataSheet.pdf (Accessed date 23 April 2008)

[16] Deri, L., and Carbone, R. (2008) PF_Ring https://svn.ntop.org/svn/ntop/trunk/PF_RING/ (Accessed date 23 April 2008)

[17] Deri, L. (2005) Ncap: wire-speed packet capture and transmission, IEEE End-to-End Monitoring Techniques and Services, 2007. E2EMON '05 http://ieeexplore.ieee.org/iel5/10461/33206/01564468.pdf?tp=&arnumber=1564468&isnumber=33206 (Accessed date 23 April 2008)

[18] Jacobson, V. et al, (2007) Libpcap, http://www.tcpdump.org/(Access date 2 April 2008)

[19] Jacobson, V, Leres C, McCanne, S (2007) tcpdump, http://www.tcpdump.org/ (Access date 2 April 2008).

[20] Roesch, M. (1998) SNORT, http://www.snort.org/. (Access date 1 April 2008)

[21] Cisco (2008) Cisco Secure Intrusion Prevention System 6.0, http://www.cisco.com/univercd/cc/td/doc/product/iaabu/csids/csids13/index.htm

[22] United Stated Navy Surface Warfare Center (2003) Shadow IDS http://www.whitehats.ca/downloads/forensics/ (Accessed date 23 April 2008)

[23] Kershar, M. (2007) Kismet http://www.kismetwireless.net/ (Accessed date 23 April 2008)

[24] Milner, M. (2004) Netstunbler.com http://www.netstumbler.com/ (Accessed date 23 April 2008)

[25] Fox-IT (2008) FoxReplay http://www.foxreplay.com/ (Accessed date 23 April 2008)

[26] Opnet (2008) Application Performance Management, http://www.opnet.com/solutions/application_performance/ (Accessed date 23 April 2008)

[27] Combs, G. et al, (2008) wireshark, http://www.wireshark.org/ (Access date 2 April 2008).

[28] NetWitness (2008) NetWitness – Total Network Knowledge –Investigator http://www.netwitness.com/products/investigator.aspx (Access date 16 June 2008)

[29] Datacom Systems (2008) The Filtered SINGLE*stream*™ Link Aggregation Tap http://www.datacomsystems.com/products/filtered-aggregation-tap.asp, (Accessed date 20 June 2008)

[30] Curry, J. (2008) SANCP http://www.metre.net/sancp.html (Access date 2 April 2008)

[31] Massicotte, F. (2007) Intrusion detection system (IDS) testing with a packet stimulator system, Communication Research Centre Canada CRC-TN-2007-003 http://www.crc.ca/en/html/library/home/publications/pubdb/pub-details?publication_id=98022 (Accessed date 23 April 2008)

[32] Walravens, C. and Gaidioz, B. (2006) Receive Descriptor Recycling for Small Packet High Speed Ethernet Traffic, IEEE Electrotechnical Conference MELECON '06, pages 1252-1256 http://ieeexplore.ieee.org/xpl/RecentCon.jsp?punumber=11001 (Access date 26 June 2008)

[33] Miller, D. (2005) ethtool, http://sourceforge.net/projects/gkernel/

[34] Manley, J., Filiba, T. Werthimer, D. (2008) CASPER Memo 21: 10GbE Network card benchmarking, http://casper.berkeley.edu/memos/nics.pdf (Access date 26 June 2008)

[35] Lawrence Berkeley National Laboratory (2004) TCP Tuning Guide http://dsd.lbl.gov/TCP-tuning /linux.html (Accessed date 23 April 2008)

[36] Nicola, B. (2008) lindump, http://awgn.antifork.org/codes/lindump.c (Access date 22 June 2008)

[37] Anderson, E. and Arlitt, M. (2006) Full Packet Capture and Offline Analysis 1 and 10 Gb/s Networks, Hewlett Packard http://www.hpl.hp.com/techreports/2006/HPL-2006-156.html (Access date 22 June 2008)

[38] Donelly, S. (2006) DAG Capture Performance, Endace http://www.endace.com/assets/docs/accelerated/DAGPacketCapturePerformance.pdf (Access date 20 June 2008)

[39] Mano, C. et al (2006) CLog: Low Cost Gigabit Full Packet Logging, Journal Of Communications, 1, [7], 17-23

[40] Viken, B. and Heegaard, P. (2002) Performance Evaluation of a Low-cost Network Monitor Proceedings of 17th Nordic Teletraffic Seminar, Fornebu, Norway http://www.telenor.com/rd/pub/rep02/R_51_2002.pdf (Access date 22 June 2008)

[41] Intel (2006) Supra–linear Packet Processing Performance with Intel Multicore Processors http://www.intel.com/technology/advanced_comm/311566.htm (Access date 24 June 2008)

[42] Graham, I. (2007) Achieving Zero-loss multi-Gigabit IDS, Endace http://www.touchbriefings.com/pdf/2259/graham.pdf (Access date 20 June 2008)

[43] IP Fabrics (2006) IP Fabrics DeepSweep Redefines the Playing Field in High Performance Netqwork Surveillance, http://www.ipfabrics.com/pdf/DeepSweep%20Performance.pdf (Access date 26 June 2008)

[44] mcObject (2007) In-memory Database Systems (IMDSs Beyond the Terabyte Size Boundary, http://www.leadingtek.com.cn/developer-zone/content/whitepaper/TerabyteBenchmark.pdf (Access date 27 June 2008)

[45] mySQL (2008) MySql Cluster, http://www.mysql.com/products/database/cluster/ (Access date 3 July 2008)

[46] Deutsch P. (1996) RFC 1952 GZIP File Format Specification Version 4.3 http://rfc.net/rfc1952.html (Access date 21 Nov 2006)

[47] Deutsch P. (1996) RFC 1951 Deflate Compressed Data Format Specification Version 1.3 http://rfc.net/rfc1951.html (Access date 21 Nov 2006)

[48] Sybase (2008) Column Compression Sample, http://wwwsybase.com/detail?id=1053701 (Access date 26 June 2008)

[49] Goldstein, J., Ramakrishnan, R., and Shaft, U. (1997) Compressing Relations and Indexes Technical Report no. CS Department, University of Wisconsin-Madison (Access date 26 June 2008)

[50] Ray, G, Haritsa, J. and Seshadri, S. (1995) Database Compression: A Performance Enhancement Tool, Proc. of 7th Intl. Conf. on Management of Data http://dsl.serc.iisc.ernet.in/publications/conference/compcomad95.ps.gz (Access date 3 July 2008)

[51] Baid, A. and Krishnan S. (2005) Binary Encoded Attribute-Pairing Technique for Database Compression, http://pages.cs.wisc.edu/~swetha/764Project.pdf (Access date 3 July 2008)

[52] Holben R. (2006) Juniper CALEA(LI)/Monitoring Solution Architectures http://www.uknof.org.uk/uknof5/Holben-Lawful_Intercept.ppt (Access date 26 June 2008)

[53] SS8 (2007) The Ready Guide to Intercept Legislation 2.0, http://www.ss8.com/ready-guide.ph (Access date 27 June 2008)

[54] ETSI (2004) ETSI TS 102 232 V1.1.1 (2004-02) Telecommunications security: Lawful Interception (LI) Handover specification for IP delivery http://www.gliif.org/ (Access date 24 June 2008)

[55] Federal Bureau of Investigation (2007) Communications Assistance for Law Enforcement Act of 1994 (CALEA) http://www.askcalea.net/calea/ (Access date 27 June 2008)

[56] Branch, P (2003) Lawful Interception of IP Traffic http://atnac2003.atcrc.com/ORALS/branch.pdf (Access date 24 June 2008)

[57] Valiquet D. (2005) Bill C-74: Modernization of Investigative Techniques Act, http://www.parl.gc.ca/common/Bills_ls.asp?lang=E&ls=c74&source=library_prb&Parl=38&Ses=1 (Access Date 27 June 2008)

[58] Lawful Access – Another Bill Expected? http://goliath.ecnext.com/coms2/gi_0199-5996500/Lawful-Access-Another-Bill-Expected.html (Access Date 27 June 2008)

[59] Golman, A (2007) A Description of Lawful Intercept and Calea, http://www.isp-planet.com/politics/2007/ss8_calea_basics.html (Access date 26 June 2008)

[60] Keeling G. (2006) Higher Education CALEA Panel Update, http://www.sdbor.edu/administration/information_technology/resources/documents/WW-CALEApptpresentations.pdf

[61] Wireless Internet Service Providers Association (2007) CALEA Compliance Guidance for WISPs, http://www.wispa.org/CALEA_compliance_for_WISPs.pdf (Access date 27 June 2008)

[62] Solera Networks (2008) Calea and Lawful Intercept, http://www.soleranetworks.com/solutions/calea-compliance.php (Access date 24 June 2008)

[63] ITU-T (2002) Abstract Syntax Notation 1, ISO 8824, X.208, X.680 http://www.itu.int/ITU-T/studygroups/com17/languages/X.680-0207.pdf (Access date 3 July 2008)

[64] Deutsch, P. (1996) RFC 1951: Deflate Compressed Data Format Specification 1.3 http://rfc.net/rfc1951.html (Access date 21 Nov 2006)

[65] Deutsch, P. (1996) RFC 1952: GZIP file format specification version 4.3 http://madhaus.utcs.toronto.ca/links/ftp/doc/rfc/rfc1952.txt (Access date 20 Nov 2006)

[66] maximumcompression.com (2008) Maximum compression http://www.maximumcompression.com/ (Access date 26 June 2008)

[67] Ziv, J., and Lempel, A. (1977) A Universal Algorithm for Sequential Data Compression, IEEE Transactions on Information Theory, [23] 3 337-343

[68] Huffman, D. (1952), "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., pp 1098-1102

[69] Scarpaza D., Villa, O. and Petrini F. (2008) String Search on Multicore Processors, Dr Dobbs Journal April 2008

[70] Ntop.org (2007) Ntop Overview http://www.ntop.org/overview.html (Access date 2 July 2008)

[71] Cranor, C., et al (2003) Gigascope: a stream database for network applications, SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data, 647—651

[72] Moore, A., et al (2003) Architecture of a Network Monitor, Passive and Active Measurement Workshop (PAM 2003) 77-86

[73] Michaut, A. and Lepage, F. (2005) Application-Oriented Network Metrology: Metrics and Active Measurement Tools 7 [2] 2-24

[74] Computer Associates (2008) Spectrum http://www.ca.com/files/ProductBriefs/spectrum_family_brief.pdf (Access date 2 July 2008)

[75] Hewlett Packard (2006) HP HP OpenView Network Node Manager, https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-15-119^1155_4000_100 (Access date 2 July 2008)

[76] ManageEngine (2008) OP Manager http://manageengine.adventnet.com/products/opmanager/index.html (Access date 2 July 2008)

[77] PacketMotion (2008) Packet Sentry, http://www.packetmotion.com/English/Products/Products.html (Access date 2 July 2008)

[78] Q1 Labs (2008) QRadar http://www.q1labs.com/products/ (Access date 2 July 2008)

[79] Lancope (2008) StealthWatch http://www.lancope.com/products/ (Access date 2 July 2008)

[80] Mazu Networks (2008) Profiler http://www.mazunetworks.com/products/mazu-nba.php (Access date 2 July 2008)

[81] Rifkin, J. (2005) IPAudit, http://ipaudit.sourceforge.net/ (Access date 3 July 2008)

[82] Curry J. (2003) SANCP - Security Analyst Network Connection Profiler http://www.metre.net/sancp.html (Accessed date 23 April 2008)

[83] Han, S. et al (2002) The Architecture of NG-MON: A Passive Network Monitoring System, InProceeding of 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, http://dpnm.postech.ac.kr/papers/DSOM/02/ngmon/ngmon-dsom2002.pdf (Access date 3 July 2008)

[84] nPulse (2008) nProbe: an Open Source NetFlow Probe for Gigabit Networks http://www.npulsenetworks.com/images/docs/nProbeWhitepaper.pdf (Access date 2 July 2008)

[85] CISCO (2008) Cisco IOS NetFlow Introduction http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html (Accessed date 23 April 2008)

[86] Juniper (2008) JFlow, http://www.juniper.net/techpubs/software/erx/junose60/swconfig-routing-vol1/html/ip-jflow-stats-config.html (Access date 3 July 2008)

[87] Sflow.org (2008) sFlow, http://www.sflow.org/ (Access date 3 July 2008)

[88] Sflow.org (2008) sFlow Collectors http://www.sflow.org/products/collectors.php (Access date 3 July 2008)

[89] QosMos (2008) Q-Work, http://www.qosmos.net/category/content/products/other-products (Access date 2 July 2008)

[90] Solar Winds (2008) Netflow Traffic Analyzer, http://www.solarwinds.com/products/orion/nta/index.aspx (Access date 3 July 2008)

[91] Arbor Networks (2008) Peakflow X, http://www.arbornetworks.com/en/peakflow-x.html (Access date 2 July 2008)

[92] CISCO (2007) CISCO IOS Netflow Version 9 Flow-Record Format, http://www.cisco.com/en/US/tech/tk648/tk362/technologies_white_paper09186a00800a3db9.shtml (Access date 24 June 2008)

[93] CISCO (2007) Netflow Performance Analysis http://www.cisco.com/en/US/tech/tk812/technologies_white_paper0900aecd802a0eb9.shtml (Access date 24 June 2008)

[94] Turner, A. and Bing, M. (2008) Tcpreplay, http://tcpreplay.synfin.net/trac/ (Access date 18 July, 2008)

[95] Libes, D. (2006) Expect, http://expect.nist.gov/ (Access date 18 July, 2008)

Annex A CALEA Versus ETSI Standards

The ability to exchange packet data with law enforcement agencies can be a desirable feature when choosing a packet logging solution. There are two major standards for transferring packet data. ETSI and CALEA. The ETSI standard is popular in Europe while the CALEA standard is used in the United States. Both the ETSI and CALEA standards specify that a LEA can obtain a warrant to compel a telecom/ISP to capture all call/communication related activity associated with a target. The warrant can request the data at several levels of detail. In the case of CALEA the warrant can ask for one of three things [59]:

- Call records (records that are similar to a phone bill)
- Real time call data (include time of call, answer, disconnect, call forward etc.)
- Call content (all the data associated with a call)

All captured data needs to be sent to the LEA in an agreed upon data format. In CALEA the telecom/ISP provider needs to provide two different data interfaces: one for the summary data surrounding the call and another for the call content. The summary data is sent using the Communication Identifying Information (CMII) interface [60] which contains:

- Dialed Digits (Voice Calls)
- Subject Login (data)
- Network Addresses (data)

The call content or packet data interface in CALEA is stored following the PCAP [61] file format. The binary PCAP format stores each packet in its raw form as it would be read on the wire. Each packet is prefaced with a timestamp and packet size. The file itself includes a short header that describes the files format and how the packets are captured. Prior to its use in CALEA, PCAP was a popular data format used by the open source libpcap library and tcpdump tool. The PCAP format is used in a number of open source tools and is supported and as an export option by a number of commercial packets loggers.

In contrast ETSI standard has three data intefaces named: HI-1, HI-2 and HI-3. HI-1 is used to transfer the detail of the warrant to the telecom/ISP provider. HI-2 provides a summary of the packet activity. Finally HI-3 contains the packet data. Each of these data interfaces are defined using the ASN.1 [48] data notation. ETSI does not have a single data format for all packets. In ETSI email packet are stored differently than a VoIP packet. This makes the ETSI format more data processing intensive to implement. In addition, the ETSI format includes an investigation identifier that allows different investigations to be better separated from one another when it reaches the LEA.

CALEA and ETSI provide a standard data exchange format between a telecom/ISP provider and a LEA. However the problem space investigated in this study more closely reflects a forensic analysis than a lawful wiretapping investigation. In lawful wiretapping investigation the ISP receives an identified target, maps it to an IP address, and then collects the packets associated with that IP address. In forensic analysis the analyst begins with a list of all packets, finds something suspicious, then maps it to the end host, and then converts it to end user. Since in this application the packets are captured in bulk and stored for forensic use the ETSI lawful interception identifier field [54] can not be populated during initial packet capture. With respect to the summarized data the packets being collected at the data collector do not have ready access to the NAT translation device or authentication devices which would be needed to associate a user with a packet exchange.

List of symbols/abbreviations/acronyms/initialisms

CALEA	Communications Assistance for Law Enforcement Act
COTS	Common Off The Shelf
CPU	Central Processing Unit
DND	Department of National Defence
DRDC	Defence Research & Development Canada
DRDKIM	Director Research and Development Knowledge and Information Management
GB	Gigabyte (2^{30} bytes)
Gb	Gigabit (2^{30} bits)
GZ	The file extension used to indicate a compress GZIP file
GZIP	GNU ZIP (a widely deployed compression format used available for use on a broad range of computer platforms)
IDS	Intrusion Detection System
ICMP	Internet Control Message Protocol
IP	Internet Protocol
IPX	A file extension used to indicate a file stored in the IP Index file format
JNDMS	Joint Network Defence Management System
LEA	Law Enforcement Agency
MB	Megabyte (2^{20} bytes)
Mb	Megabit (2^{20} bits)
Mbps	Megabit per second
NIC	Network Interface Card
PIDX	A file extension used to indicate a file stored in the Packet Index file format
PCAP	Packet Capture (a stand file format for saving packet capture data)
PZIP	Packet Zip (a tool developed as part of this study that is based on the GZIP software program)
R&D	Research & Development
SIEM	Security Information and Event Management system
SIM	Security Information Management system
TCPZIP	TCPdump pZIP (A program developed as part of this study that combines the functionality of tcpdump and pzip into a single program)

TDP	Technology Demonstration Program

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. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.) Defence R&D Canada – Ottawa 3701 Carling Avenue Ottawa, Ontario 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.)

Fast Packet Recovery: Technical Challenges and Solutions

4. AUTHORS (last name, followed by initials – ranks, titles, etc. not to be used)

Vandenberghe G.

5. DATE OF PUBLICATION (Month and year of publication of document.) October 2008	6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.) 60	6b. NO. OF REFS (Total cited in document.) 95

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 address.)

Defence R&D Canada – Ottawa
3701 Carling Avenue
Ottawa, Ontario K1A 0Z4

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.)	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.) DRDC Ottawa TM 2008-209	10b. OTHER DOCUMENT NO(s). (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.)

Unlimited

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.))

Unlimited

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), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

Network security analysts review the packets associated with network security events on a daily basis. The raw packet data being reviewed is stored on a distributed packet logging system. These real time data acquisition systems generate large files that are challenging to manage. This study demonstrates a new technique for storing packet data which is in a selectively recoverable compressed format. Packet data can be extracted many times faster than from a standard compressed file. In addition the compressed file adheres to the file format used by the GZIP compression algorithm. This means that compressed files can be readily exchanged and decompressed using standard LINUX tools. Although data can be extracted quickly from the compressed file there needs to be a means of finding relevant data to extract. An IP based summarization approach is demonstrated which is faster and generates more compact data files than a commonly used flow based traffic summarization tool. Finally, to simplify data collection for the distributed packet logging system a centralized user interface is demonstrated that can find and recover the packets requested by the analyst.

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.)

Compression, Packet Logging