ABSTRACT

GOFF, BRIAN DAVID. Distributed Resource Monitoring Tool and its Use in Security and Quality of Service Evaluation. (Under the direction of Dr. Mladen A. Vouk.)

As networks become increasingly more open and complex, their management becomes that much more important. Numerous commercial and open-source tools already exist that are capable of providing useful network analysis. Unfortunately, the majority of these commercial tools are either quite expensive, or require a significant amount of effort for their deployment. Other tools are very specific in their tasks, and offer little in the way of customization. Tools able to provide favorable statistics often fall short when it comes to operating on loaded high-bandwidth networks. Also, as the majority of network management tools operate at the packet level, they often require administrator-level access to capture the data. In addition, there are privacy issues that may also limit who has access to what part of the data. On large networks, this can amount to a lot of data that needs to be scrutinized by a limited number of people.

The purpose of this project was to develop an inexpensive, customizable network-monitoring tool (called Resource Usage Monitor) that a) is capable of providing a variety of traffic-related statistical data on a high-bandwidth network, b) provides a user-friendly selective access to that data for users with different privacy privileges, and c) interfaces to a policy management toolset to allow pro-active management of the network based on security, quality and resource information it gathers. The processing engine behind the Resource Usage Monitor (RUM) examines data at the monitored gateway, collecting

inbound and outbound information on the number of bytes, packets, network and application flows, and such, transmitted or received for each internal and external host. A web interface provides persistence graphs and reports that can disclose general and specific traffic patterns on the network. This information can be used to assess security, resource usage, and quality of service (QoS) assessment of the monitored network and hosts. For example, setting flow, load, and other activity thresholds at different levels of granularity allows for the detection of anomalies throughout the network. Port scans, Denial of Service (DOS) attacks, and Trojan applications have been detected through surveillance of simple threshold-based patterns. More complex, possibly multi-probe, patterns can reveal much more subtle anomalies and side effects.

RUM operates in a statistical mode, rather than continuous mode. It samples the network every few minutes. After each sample the collected data is analyzed and appropriate warnings and interactions with the policy services are effected. Collected packets are sorted and stored by pre-defined subnets allowing parallelism in the processing of the data. This also separates datasets for another reason; it enables secure access to just the network traffic for which an administrator is responsible. Persistence data is kept in the form of graphs generated using RRDtool, a round robin database utility, and in logs. The information logged is completely customizable and can even be offloaded for analysis by other systems. This thesis describes the RUM architecture, data it collects, analysis modules, user handling, and other features.

DISTRIBUTED RESOURCE MONITORING TOOL AND ITS USE IN SECURITY AND QUALITY OF SERVICE EVALUATION

by
BRIAN DAVID GOFF

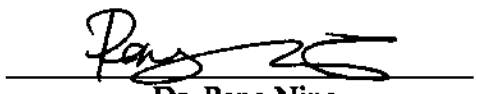
A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

COMPUTER NETWORKING

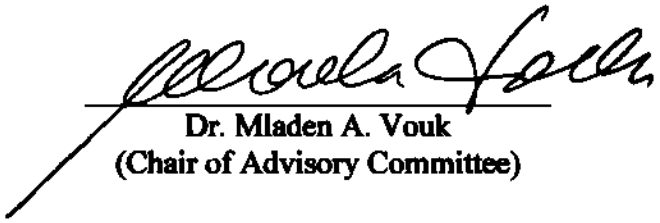
Raleigh
2002

APPROVED BY:

Dr. Rudra Dutta



Dr. Peng Ning



Dr. Mladen A. Vouk
(Chair of Advisory Committee)

BIOGRAPHY

Brian Goff was born on June 6th, 1978 in Chicago, Illinois. While growing up, he lived in Boston, MA, Boulder, CO, and Ramsey, NJ before settling in Cary, NC in 1987. In 2000, he graduated Summa Cum Laude from North Carolina State University with a Bachelor of Science in Biological Engineering with a Biomedical Concentration and a minor in Computer Science. In August 2000, he began graduate studies in Computer Networking at North Carolina State University.

ACKNOWLEDGEMENTS

I would like to express sincere gratitude to Dr. Mladen A. Vouk for his guidance and support in the pursuit of this work. Sincere appreciation is also given to members of the university's Communication Technology Division and Computer Science Department, without whom this work would not have been possible. I am especially thankful for the support of my family and friends. Finally, I am grateful to my best friend, Meghan, whose love, encouragement, and patience have contributed immensely to this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. Introduction

1.1 Motivation and Goals

The initial goal of this research was aimed at better understanding the university's network. Several different network monitoring and management tools were already in place, however the university still lacked any means of generating a usage profile of its network. Early research consisted of examining all of the current network tools available, both commercial and open-source. Two separate test machines, one running Windows 2000 and the other running Red Hat Linux, were set up to collect traffic from the university's network. One by one, any tool that could be acquired for testing was setup and examined for its usefulness.

Results of the initial research phase demonstrated that while there are several tools available, they fall into one of several categories. Many tools examined are only capable of collecting network data. These capturing tools have no true analysis included in their functionality. Other tools provided numerous types of statistics on the network data, however they either did not work with the gigabit Ethernet network card required to capture traffic on a 300+ Mbps network, or could not maintain the same level of efficiency with such a high load. The few tools that provided sufficient processing were either quite expensive or lacked customization features to provide the specific types of statistical data.

As the survey of other tools continued, it became evident that in order to provide the desired network analysis, a custom tool would have to be designed. Early research concentrated on analyzing data captured using other tools, while the most recent research

consists of a completely independent capturing and processing utility. The goal of this research was to develop an inexpensive, customizable network-monitoring tool that could then be modularly extended to perform a variety of analysis or interaction tasks. The tool has to be able to process data from a high-speed network and provide data in several different formats. In addition to providing basic network traffic analysis, such as flows and throughput, the tool was required to provide the basic security, quality of service (QoS), and policy-based control interaction functions. The latter is a special distinction of RUM since it couples the network probe and analysis function with active QoS and security control.

1.2 Thesis Layout

Chapter two of this thesis examines several of the other tools available to perform various network monitoring tasks. In order to help establish the reasoning behind the development of the Resource Usage Monitor (RUM), the similarities and differences of each tool are highlighted. Chapter three explains the actual design of the Resource Usage Monitor application. Each part of the tool, from the data analysis and user interface, to the installation and configuration is included. RUM can be applied to several different network management tasks. Chapter four explains the usefulness of the tool for security analysis, while chapter five demonstrates the quality of service application. Chapter six offers conclusions regarding the usefulness of RUM and offers suggestions for future work. Finally a copy of the RUM user manual is included in the appendix to offer more information about the application itself.

2. Tools Survey

Network management is not a new field. The International Organization for Standardization (ISO) first started working on the simple network management protocol (SNMP) in the mid-1980s [3]. Since then, numerous open-source and commercial tools have been developed. Some of these tools are designed for a specific functionality, such as just collecting network data. Other tools both collect and analyze network data making them better suited for stand-alone operation. The majority of the tools are designed for smaller networks where the network bandwidth stays below 100 Mbps. Tools capable of monitoring larger networks exist as well, however they are often only available through commercial vendors at a considerable cost. Each of the following tools has at least one type of network management functionality comparable to a function in the Resource Usage Monitor.

2.1 tcpdump / WinDump

Early versions of the Resource Usage Monitor actually used tcpdump to capture traffic. Initial releases of RUM built on top of tcpdump. However, tcpdump was found to be a bottleneck when dealing with very high-speed inputs. Therefore, a separate data collection interface was built for more recent versions of RUM. Still several similarities between RUM and tcpdump remain. tcpdump is an open-source packet capture program that runs on UNIX platforms [14]. WinDump is the windows distribution that provides similar functionality [6]. Run in promiscuous mode, these programs are able to provide data about each packet sent past the network interface card (NIC). Both tcpdump and RUM use the pcap library to capture network traffic. The packet capture library (libpcap) provides a high level interface to the packet capture systems [15]. Since each utility uses

this library, each is also capable of handling BSD Packet Filter (BPF) commands [19]. These commands allow for specific types of packets to be filtered out before they are handed over to the high-level program. tcpdump is also capable of analyzing network data from a previously captured raw pcap file. RUM does not have this functionality and must capture live network traffic for each sample.

The higher-level processing of each packet is where tcpdump and RUM differ. Since tcpdump's function is to provide a dump of the network data, it supports several different network protocols. Processing decisions are based on the link layer, network layer, and transport layer information. RUM, too, pulls data from these parts of each packet, however, all non-IP packets are omitted and only a limited amount of information is collected from the transport layer of TCP and UDP packets. This smaller amount of processing allows the RUM probe to collect packets on a high-bandwidth network where tcpdump drops too many of them. In general, tcpdump is best suited for small to medium-sized networks, since each packet is immediately written to disk, or the screen, upon arrival. This difference in packet processing is further discussed in section 3.2.1.

As part of the tcpdump processing, there is a limited amount of packet assembly. This mainly consists of the reassembly of packet fragments, but nothing as complex as reconstructing TCP streams. The Resource Usage Monitor does almost the same, but treats every single packet individually at first. Later analyses provide the assembly of packets into network flows. This is another difference between tcpdump and RUM. After tcpdump determines the details of each packet, it is done. The only summary statistics provided by tcpdump are total counts of the number of packets collected. There

is also no interface, except the command line, for tcpdump to interact with other applications. In order to use tcpdump data in some other analysis, the application needs to parse the output from tcpdump. This is how RUM initially incorporated tcpdump into its operation.

2.2 Ethereal / tethereal

Ethereal works much like tcpdump in that it is also a packet capture program [30]. While Ethereal is a free application with a graphical user interface, a text-based version, tethereal, is also available. This protocol analyzer also uses libpcap to provide the packet capturing functionality. This enables the use of BPF commands while capturing data. Like tcpdump, Ethereal can either collect data live, or parse a previously captured network data file. Ethereal is able to understand data from several different other capture programs.

The actual analysis that takes place in Ethereal also supports several different protocols. This analysis is even customizable as new protocol modules can be added to the program. Filters enable the display of just the data that a user is interested in viewing. Ethereal does provide some post analysis of captured data such as TCP stream reconstruction. Upon analyzing a sample of network data, summary statistics are provided for each protocol analyzed. These statistics are only displayed for the overall capture and are limited to just the quantity of packets. In order to interface with the results of an Ethereal sample, the data would have to be first exported and then parsed by another application. Like tcpdump, this tool performs better on small to medium-sized networks. The

overhead from the graphical interface makes captures, even for a limited time, especially difficult when operating on a high-bandwidth network.

2.3 ntop

Out of the numerous monitoring tools, ntop probably has the most similarities with the Resource Usage Monitor. The components making up ntop include a packet sniffer, traffic analysis, information storage, and an embedded web server [10]. This design is very similar to the design of RUM. Each tool is capable of producing graphs and reports about the network data, and displaying them through a web interface. ntop also has a command line interface, intop, which is able to access any of the processed reports form the ntop processing engine. This type of data collection and display make both ntop and RUM very good for anomaly detection when performing network security analysis.

Like several of the other tools, ntop uses the same packet capture library to obtain the network data. Since ntop operates in continuous mode, it is designed to operate on networks of less than 100 Mbps [9]. This is different from the statistical mode RUM uses to monitor the higher-bandwidth networks. While ntop does provide several methods of data analysis that are the same as those of RUM, such as examining traffic by protocol, viewing traffic distributions, or even examining traffic to/from a specific host, it does not permit the separation of data by subnets. If a user is permitted access to the ntop data, they will subsequently have access to all of the data provided within. This differs from the RUM system where individual users can be granted access to just specific areas of the network.

Other features of ntop include the analysis of network flows, some intrusion detection (IDS) functionality, remote host operating system (OS) identification, and the support for several non-IP protocols. RUM also posses network flow analysis and remote host fingerprinting capability, however not the other features. The ntop processing engine is much more involved, then again it is designed for less loaded networks. The remote host OS identification is based on Nmap, a tool that attempts to guess the operating system on a remote host through actively sending bad packets to a machine and analyzing the remote machine's response [11]. The Resource Usage Monitor provides remote host identification through passive fingerprinting instead. This is explained in more detail in section 3.3.6.

2.4 Snort

Another open source network tool, Snort, provides network intrusion detection system (NIDS) functionality [28]. Snort again uses the same packet capture library as all of the previous tools and RUM. Snort contains the same BPF command capabilities as well. Snort begins to differ in how it processes each packet. Instead of simply capturing packets, Snort examines each packet against a list of rules. Upon matching specific rules, the corresponding action is taken, such as logging the packet. The Resource Usage Monitor has this same functionality, however, it takes place after all of the other processing it completed. Also, while in RUM different users can create their own conditions to match packets, these criteria are much more limited than those of Snort. Snort actually examines the data content of each packet, while RUM does not. Through this extra analysis Snort is able to identify several different packet signatures and act as an effective misuse detection utility.

On the other hand, this extra processing limits Snort in its ability to perform on larger networks. Snort is marketed as a lightweight open source tool designed for the UNIX environment and ported to Windows. Since Snort operates in a continuous mode, it is unable to keep up the extended processing for every packet on a high-bandwidth network.

2.5 Network Flight Recorder

Network Flight Recorder (NFR) does packet analysis very similar to snort [25]. The same process of continuous network monitoring and quickly examination of each part of every packet is used. In recent years, NFR progressed into a network intrusion detection system (NFR NID) [22]. NFR NID is a misuse detection application with more in-depth signatures than snort. NFR NID comes in two different versions. One version runs on customer-supplied hardware and it is designed for use on small to medium sized networks. The other version requires special hardware and system configuration, however it is able to provide continuous monitoring up to 100 Mbps. RUM is able run under an ordinary Linux configuration, and through the process of intermittent monitoring, it has been able to run at network bandwidth speeds about 300 Mbps on 750 MHz Netfinity X330's.

2.6 Network Associates Sniffer Technologies

Network Associates Inc. (NAI) offers several different products in the Sniffer line [21]. Each product is designed for a specific type or size of network and runs in the Windows environment. Sniffer Portable products are intended for use on individual probe machines and small to medium-sized networks. In combination with a specialized network interface card (NIC) provided by Network Associates, these products are able to

capture at least 95% of the traffic on a half-duplex 100 Mbps segment [1]. These products provide much of the same analysis as RUM, such as network graphs and reports to display network statistics. The Sniffer Pro line supports decoding for over 300 network protocols with real-time analysis. NAI has other Sniffer products capable of monitoring high-speed networks, however both their cost and the required effort for deployment are very high. Sniffer products do not possess the ability to separate data between users.

2.7 Microsoft Network Monitor

The network monitor provided by Microsoft is another commercial network protocol analyzer tool [20]. The simple version is included with Microsoft NT Server distributions, while the full version is part of the Systems Management Server. The full version provides similar packet capturing and processing functions to the NAI Sniffer products. Comparing this tool to RUM, the Microsoft tool specializes in decoding the upper level protocols vital to a Microsoft network such as Server Message Block (SMB), Microsoft Browse, and Microsoft Remote Procedure Call (MS RPC), where RUM solely analyzes IP traffic with minimal transport layer analysis [13]. This extra processing again limits this tool in its ability to perform on faster networks. The Microsoft products also offer the ability to set triggers based on specific criteria much like the feature in RUM.

2.8 Cisco's NetFlow, cflowd, and FlowScan

These three products together provide a very similar type of network data processing and analysis to that found in the Resource Network Monitor. NetFlow records flow

information at the Cisco switch and stores that information to be later retrieved [18]. NetFlow uses seven different parameters to determine a network flow; source IP address, destination IP address, source port, destination port, layer three protocol, type of service (TOS) value, and interface. RUM uses a very similar format without the TOS value and without the interface since RUM does not operate like a switch and only receives packets on one interface. NetFlow is the basis of Cisco's IP accounting functionality, which enables quality of service evaluation.

cflowd is a flow analysis tool specifically designed to extract data from Cisco's NetFlow [4]. When combined with other utilities, cflowd can provide a report of the flows encountered by NetFlow. FlowScan takes the cflowd processing one step further and presents the data in graphical format [5]. FlowScan is made up a modified version of cflowd to act as the processing engine and a persistence database and graphing utility in the form of RRDtool. This same type high-performance database and graphing function is used in the Resource Usage Monitor.

Since Cisco's NetFlow and FlowScan operate from a switch or a router, no probe machine is required. The reports and graphs that can be obtained from cflowd and FlowScan are much like the statistical analysis available in RUM. RUM is different though, in that it provides more packet level data. Where the network flow data is all that is kept in the NetFlow, RUM maintains packet level data through the initial processing stage. RUM also collects the data by subnet category allowing for subnet-specific user access through the web interface.

2.9 MRTG / RRDtool

The Multi Router Traffic Grapher (MRTG) and Round Robin Database tool (RRDtool) are similar in that they each provide graphs [23][24]. MRTG is specifically designed for traffic monitoring however like RRDtool, it can realistically be used to monitor any type of data. MRTG uses simple network management protocol (SNMP) to poll routers, switches, and other SNMP enabled devices for data. This data is then used to create the graphs. MRTG is usually used to display traffic summary statistics.

While RRDtool can create the same type of graphs as MRTG, it is not set up to work directly with SNMP. RRDtool does not handle data collection on its own. Through the use of other applications, traffic data can be entered into its round robin database (RRD) and later used to create statistical graphs. RUM uses RRDtool for all of its graphing functionality.

2.10 Other Tools

This list of tools only touches on the wide variety of tools that exist today. Several other open source tools can be found on the Internet, even some that perform very similar functions to the Resource Usage Monitor. When it comes to network protocol analyzers, many more commercial tools also exist. Generally, the majority of these tools are slated for a small to medium sized networks. Those that can handle higher network loads, just like some of the NAI Sniffer products, start to get quite expensive. Most of these tools do not provide an interface to a network management policy server, and are not used to pro-actively control networks. RUM has that capability.

3. Implementation

This section describes the design of Resource Usage Monitor. There have been four major stages in the development of RUM. The first three stages are briefly discussed in the next subsection, while the remainder of the chapter concentrates on the most recent stage, version 1.1. Successes and drawbacks encountered throughout the design process are included.

3.1 Early Versions

The very first implementation of a monitoring tool that has developed into RUM was a shell script. This tool took one-minute samples of network traffic data using tcpdump [14]. Then, using a series of awk and other shell processing commands, the data was parsed and sorted to provide statistics on inbound and outbound traffic, high-usage hosts, and high-usage ports or applications. In the next stage of development, tcpdump was still used to collect one-minute long data samples, however now the data was parsed using a Perl script. Flow statistics were introduced at this point along with persistence graphs built using RRDtool [24]. The third stage of development was the first official release of the Resource Usage Monitor, version 1.0. This version included a processing engine that collected and analyzed data as a whole and by internal subnets. Again, tcpdump was used to acquire the data, and all of the processing was done using code written in Perl. A web interface was also included with this version to provide user specific access to various graphs and reports. Since the data was available by subnet, users could be granted access to only the specific datasets for which they were responsible. The fourth and current stage incorporates additional features discussed below.

3.2 Data Collection and Processing

In RUM, data collection is statistical. That is, traffic is only sampled for a relatively short period of time before being analyzed. After a sample is analyzed, the process is repeated. There are several reasons why it was decided to use statistical rather than continuous. RUM is designed to run on top of a common Linux installation. In order to capture packets from a high-speed network, the processor must be able to immediately set each packet aside. Processing each packet as it is captured will only add to the amount of work for the processor during the resource-critical packet capture procedure. Since packets will accumulate quickly, at some point the processor needs to stop collecting and work with the recently acquired data. Statistical mode also provides each user with a specific set of data to examine. A high-bandwidth network possesses a vastly large amount of data to sort through. RUM is able to take a portion of that data and provide a representative viewpoint of the network. Network management decisions can then be made accordingly based on this statistical analysis.

The RUM processing engine is responsible for data collection and analysis. The only persistent data are kept in the form of graphs or log files. The actual packets captured from each sample are overwritten by the next set of captured data. Logged information is kept for at most 30 days, while graph data is kept for up to 1 year. This reduces the risk of storing possibly sensitive network data for a significant amount of time. The flow of data through the processing engine is shown in figure 3.1. Each of the following subsections discusses one particular step in this process.

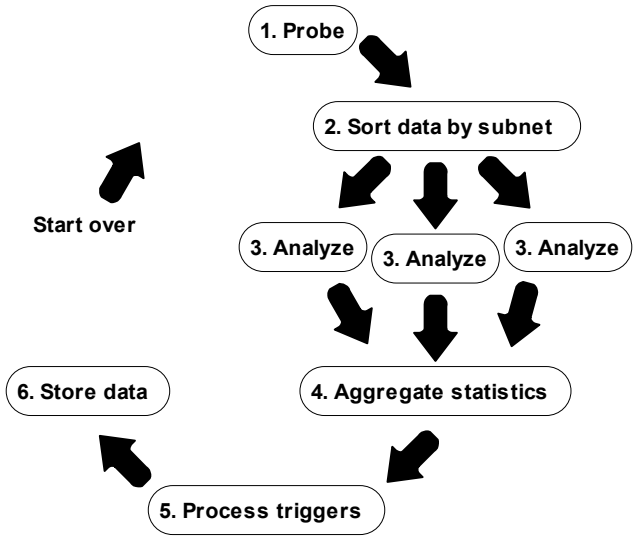


Figure 3.1. Data flow through the RUM processing engine

3.2.1 Probe

The probe is responsible for capturing the actual packets from the network. Until version 1.1, tcpdump was used to handle this task. Using tcpdump to collect data presented several difficulties. Initial development of the first RUM version was coded on Red Hat Linux 7.1, which shipped with tcpdump version 3.4 [26]. In order to extract the packet data from the tcpdump output, a Perl program was used to parse the ASCII text output. This worked well until Red Hat Linux 7.2 shipped with tcpdump version 3.6 [27]. Although the newer version of tcpdump still performed the same function, the output format was slightly different as demonstrated in figure 3.2. The omission of the packet classification in the first column of the version 3.4 output caused the indices for the remaining columns to be shifted to the left by one. While installing the older version of tcpdump, or even modifying the RUM code, could overcome this problem, the dependence on the exact output format of another application could have more drastic effects with future releases and needed to be avoided.

tcpdump version 3.4:

```
# /usr/sbin/tcpdump -i eth0 -penqt
P 0:c0:b6:ee:61:78 0:0:0:0:0:1 60: 192.168.10.63.3714 >
216.239.33.101.http: tcp 0
```

tcpdump version 3.6:

```
# /usr/sbin/tcpdump -i eth0 -penqt
0:c0:b6:ee:61:78 0:0:0:0:0:1 60: 192.168.10.63.3714 >
216.239.33.101.http: tcp 0
```

Figure 3.2. Output differences between tcpdump 3.4 and tcpdump 3.6

Another reason not to use tcpdump for capturing data is its performance on a high-bandwidth network. tcpdump is a very powerful tool able to recognize several different types of packets and the numerous options associated with each packet. Unfortunately, in order for this much classification to take place; a significant amount of time must be spent processing each packet. Figure 3.3 illustrates the process of capturing packets on a Windows platform. If any process along the pipeline takes too long, the buffers will start to back up. A chain reaction can take place when one buffer becomes full and previous buffers begin to fill up as well. After the initial packet buffer in the network interface card (NIC) is full, any new packets simply overwrite existing packets and the overwritten data is lost.

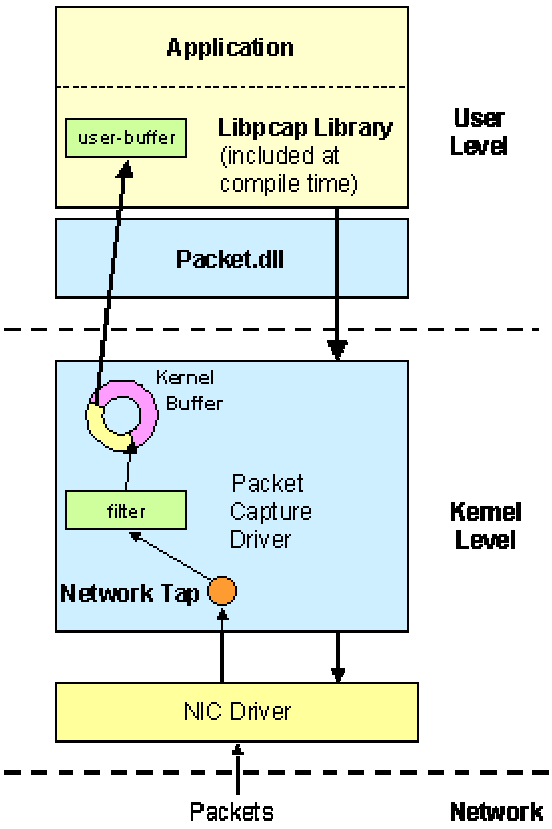


Figure 3.3. Capturing packets on an Ethernet network [7]

On our test network, network loads of roughly 300 Mbps appear to be the threshold where tcpdump does not keep up with the amount of network traffic and started to drop traffic[1]. This is a rough estimate because packet-capturing programs perform specific actions on a per-packet basis. A particular load doesn't always directly coincide with a specific amount of packets. There are several less packets on a 300 Mbps network when the packet sizes are all 1540 bytes than when the packet sizes are all 60 bytes [12]. Another parameter to factor into this rough estimate is the performance of the actual

[1] Table 3.2 lists the hardware specifications for each of the test machines. tcpdump on Test machine A was able to capture at a slightly higher load than test machine B however it still had considerable losses with high-bandwidth traffic. Disk access was believed to be the most significant bottleneck.

machine doing the capturing. In a program such as tcpdump, each packet it written either to file or stdout immediately after it is processed. Of all of the steps within the process, writing to screen or disk is the most probable bottleneck. Once data backs up here, and the tcpdump program must wait for the I/O buffers to empty, arriving packets will be dropped before tcpdump has a chance to respond to them.

In order to measure packet loss during the capturing process, RUM utilizes the statistics for a particular NIC provided by the kernel. Within the Red Hat Linux operating system, these statistics are located in the /proc/net/dev file. The number of packets captured by the device is extracted from this file at the beginning and at the end of the capture run. The difference between these two values, taking into consideration the possibility of rollover in the 32-bit counter, is compared to the number of packets physically captured by the probe to obtain the estimated sampling coverage percentage. Since the statistics in the device file are not updated with each packet, there is a significant amount of error when using these numbers with small captures. However, when capturing upwards of three to four million packets in one minute, the numbers in the device file act as a pretty good indicator of the probe performance.

This influenced the development of a specialized capturing program specific to the needs of the Resource Usage Monitor. The probe is written in C using the same packet capture library behind tcpdump, libpcap from the Lawrence Berkeley National Laboratory [15]. Default options of the library allow for the constant capturing of packets for an infinite amount of time or for the capturing of a finite number of packets. Since RUM is designed to capture packets for a specified amount of time, one minute by

default, the pcap library was modified to handle this new option. Instead of passing the total number of packets to capture through to the pcap callback function, the memory address of the variable specifying the capture duration is used. The callback function is RUM-specific code that is processed on each packet captured. During each call, the timestamp of the current packet is checked against the timestamp of the very first packet captured and the duration of the overall capture. If the newly captured packet is outside of the allowable time range, it is discarded and the capture process is stopped. With this design, the capturing of packets does not technically begin until the first packet is captured and does not end until after the first packet outside of the desired time range has been received. This may cause the appearance of abnormal capture times on a network with very light traffic[2], however such abnormalities are insignificant on a loaded network.

Another feature of the packet capture library is the ability to compile packet filter expressions into the capture engine. The same filter expressions that work in capture programs such as tcpdump and Ethereal, work in RUM. The default filter expression in RUM is "ip". As the Resource Usage Monitor only collects statistics on IP traffic, there is no need to even capture other broadcast traffic. When capturing traffic for only specific networks or subnets, this expression can be changed to meet those requirements by simply updating the line in the RUM global configuration file.

The callback function in RUM that is called for each captured packet is much simpler than the majority of other packet capturing utilities. After the initial check to see if the packet is an IP packet, the packet can fall into one of three categories.

[2] Consider a low-bandwidth network with high inter-arrival times between packets. For a capture time of 60 seconds, the probe process may run for 80 seconds while waiting for the first and last packets to arrive.

1) TCP and UDP packets are examined to see if the packet is an original packet or a fragment from another larger packet. Port numbers are extracted from these original packets.
2) For ICMP packets, the ICMP type and code values are recorded instead of port numbers.
3) For all other IP packets, no port information is examined.

For every packet, the IP protocol number, source IP address, source port, destination IP address, destination port, IP type of service (TOS), IP time to live (TTL), IP don't fragment (DF), and possibly, the TCP window size are recorded. No payload data is ever examined during the capture process. All values, including the source and destination IP addresses are stored as unsigned integers. In the case of the IP addresses, unsigned long integers are used. This eliminates the process of converting each value to a dotted decimal IP address during the time-critical capturing process. If a specific parameter doesn't exists for a given packet, such as port number or TCP window size, a zero is placed into that field. As each packet is captured, instead of writing the data to disk or to the screen, it is simply pushed atop a stack in memory (this does assume large enough memory). When the capturing process is over, a pointer to the top of the packet stack is returned.

The RUM probe currently only supports IPv4. Any type of IPv4 protocol traffic is supported but if it doesn't fall under the category of TCP, UDP, or ICMP, no protocol-specific information is extracted from the packet. Testing of the RUM probe produces

continual 100% estimated sampling coverage with network bandwidth well above the 300 Mbps threshold experienced while using tcpdump.

3.2.2 Data Classification

In the most recent version of RUM, data classification takes place as part of the probe functionality. Using the stack of packets captured in the first step, each packet is popped off the stack. Iterating through the complete list of internal subnets and their respective masks, both the source and destination IP addresses are examined for a match. The iterative process continues until all subnets have been checked in order to catch any possible internally sourced packets with an internal destination address as well. See section 3.4.2 for probe placement and the presence of internal-to-internal traffic in the capture sample. If either a source or destination subnet has been determined, the packet is pushed atop a new stack for the matched subnet. In the case where there is no match to an internal subnet, the data is stored in the "no_subnet" category. After all of the packets have been examined from the original stack, each group of packets for a particular subnet is written to a file consisting of traffic either inbound or outbound from just that subnet. Data from the "no_subnet" category is written to its own separate file. Table 3.1 shows the format of each line in the dump files. Note that even at this stage, the information is still stored as unsigned integers.

Sample output:

```
6 1057663168 3714 1696722904 80 60 0 118 1 8160
```

Corresponds to:

IP Protocol Number	6 (TCP)
Source IP Address	1057663168 (192.168.10.63)
Source port	3714
Destination IP address	1696722904 (216.239.33.101)
Destination port	80
Packet length	60 bytes
IP type of service (TOS)	0
IP time to live (TTL)	118
IP don't fragment bit (DF)	Yes
TCP window size	8160

Table 3.1. Example of packet format as it is stored in dump file

There are several benefits to separating and classifying each packet by its internal subnet. After sorting the packets, unique datasets exist for each area of the overall network. It is now possible to allow access to a particular subnet for an internal LAN administrator without giving them access to data from other parts of the network. Also, as several million packets are captured on a high-bandwidth network during any given sample, splitting up the data allows for independent and more efficient processing of each dataset. This type of analysis is described in the next section.

Earlier versions of RUM sorted the data after the packets had already been combined into flows. This allowed a smaller quantity of data to be sorted, however it also meant that less data could be carried through to the next step in the processing pipeline. The sort process has been moved up to a point before the determination of network flows in order to forward packet-specific data such as TOS, TTL, DF, and TCP window size into the subnet level of processing.

3.2.3 Subnet Analysis

At this stage of processing, there is a separate dataset for each subnet containing every inbound or outbound packet. Processing of data in one subnet if independent of data from another subnet. Hence, the processing can be done in parallel. Subnet analysis is done using a boss/worker thread model. The actual number of worker threads and the size of the work queue can be manually configured in the global RUM configuration file.

The first step in analyzing a particular subnet involves combining the packets into network flows. A **flow** is a unique IP protocol, source IP, source port, destination IP, and destination port combination. In a one-minute sample, there is an extremely high probability that packets containing the same values for each of the parameters belong to the same flow. Also, the combination of TOS, TTL, DF, and TCP window size is saved for each source host. This information is used to attempt to passively identify the operating system of the sender as described in section 3.3.6.

Once the flows have been identified, inbound and outbound statistics are tallied for the number of bytes, packets, and flows. Figures are calculated for each internal host, external host, IP multicast address, web server, and identifiable **application**. For example, statistics in the web server grouping indicate that a machine has served a request from one of the common web ports (http – 80, https – 443, http-cache – 8080). Application statistics are based on the port/application mapping found in the /etc/services file. If a mapping exists for the source port, it is used to represent the packet's application. Otherwise, the mapping for the destination port is used as the packet's application. If no mapping exists for either the source or destination port, then just the

source port number is used. In the case where there are no port numbers available to determine the application, such as with Internet control message protocol (ICMP) or Internet group message protocol (IGMP) traffic, just the IP protocol number is recorded as the application type. Along with each of these classifications, overall totals for inbound and outbound TCP, UDP and other IP traffic are computed for the given subnet.

3.2.4 No-subnet Analysis

This type of analysis is used on the "no_subnet" dataset where neither the source nor the destination address matched to an internal subnet. Packets in this class can fall into one of three categories.

1) If either the source or destination address fall within the range of addresses for the internal address, then the packet is labeled as "***Classification Pending***". Packets in this category may be legitimate but just not part of a recognized internal subnet. If the subnet list in the RUM configuration is not completely up to date, then packets not being accounted for will show up in this category. Packets with a spoofed internal source address that doesn't match to given subnet will also show up here. Section 4.5 explains in more detail about how RUM handles spoofed packets.
2) When the source and destination addresses don't match to an internal network address, then the destination address is checked to see if it is in the range of an IP multicast address (224.0.0.0 – 239.255.255.255). Since ***inbound multicast*** packets would have no characteristics matching them to a specific internal subnet, they need to be accounted for separately.

3) Finally if the packet doesn't fit into either of the first two groupings, it is labeled as "***Illegal***". These packets are generated within the internal network; however do not have a valid internal source address. Most likely these packets are either illegal spoofed, or are due to host configuration errors. Once the packets have been split into one of the three different groups, inbound and outbound statistics are tallied for the same classifications of data as used for the subnet analysis.

3.2.5 Merging Data

After all of the subnets and non-subnet datasets have been examined, some regrouping of data is required in order to provide meaningful data summaries. In order to efficiently produce graphs for a given VLAN or network, data from every subnet that makes up that grouping is merged together to determine the total inbound and outbound traffic. This includes merging data from every single data grouping to provide summary totals for the entire network being examined. This combined data is then stored in the round robin databases created through use of the RRDtool utility. Since there is a possibility for a high quantity of packets throughout the network, no reassembly of packet datasets or even non-summary statistics is carried out. If this data needs to be combined in order to produce certain reports, the web interface scripts handle the reassembly as described in section 3.3. Earlier versions of RUM maintained redundant groupings of packets falling under a major network category or overall category throughout the entire process. While this eliminated the need to regroup any of the data summaries at the end of the analysis, this was abandoned in the latest release in order to maintain a greater amount of detail for each packet.

3.2.6 Process Triggers

Trigger functionality, associated with policy processing, is new to RUM 1.1 and subsequently has not been refined for optimum performance. Until further testing is available to develop usage patterns for the type of triggers setup by users, this feature provides very basic functionality. RUM users have the option to save the parameters for different reports and then use those reports to install triggers into the system. After all of the packets for each sample are analyzed, the reports to generate triggers are automatically processed. If there is a match for any given report, the respective action is taken. This action could be to either log the data, alert the user, or both log and alert the user. These actions are further discussed in section 3.3.4. After processing each report that has been installed as a trigger, the RUM processing engine is finished with one cycle.

3.3 Web Interface

Collecting statistics on network traffic is only valuable if there is a way to effectively convey the information [2]. Presenting data through a web interface offers conveniences such as being able to access the data from anywhere in the world and no requirement for client-side software [16]. As long as the user has a JavaScript-enabled web browser, they are capable of accessing RUM data. Using the information captured and analyzed by the processing engine, the RUM web interface provides several options for user-customizable graphs, reports, and logs. A key component of the web interface is the incorporation of user access at either the subnet, VLAN, or network level. In the RUM application, a VLAN is a collection of subnets and a network is a collection of VLANs. As is the case for large networks, it is customary that several individuals are charged with its management, each being responsible only for specific areas of the network. With

RUM, the administrator is able to assign access rights for one or more specific subnets, VLANs, or network to each user. When the user logs onto the system, the user will only have the option of viewing those permitted subnets, or a collection of subnets. The rest of the network data is invisible to them. Along with access to a particular set of data, there are two levels of access, full and limited. Full access allows the user to see all available information collected by the RUM system. Limited access hides all external addresses from the user to preserve end-user privacy. With limited access, a user is still able to see how many external hosts are exchanging data with the internal hosts. All of the statistical data even shows up in the reports. The only difference is that the actual external IP address will be replaced with a series of X's.

Another component of the web interface is the ability to save the parameters for a specific graph or report. After saving the format, the same type of graph or report can be retrieved at a later time without having to go through and reselect all of the same options. Each user has their own account to which they are able to save different sets of parameters. For security reasons, the RUM administrator is able to limit the total number of saved graphs/reports for each user. The different items that can be saved include graphs, reports, packet searches, and logs. Saving reports and packets searches have an added feature as once saved, they can be installed into the system and processed automatically during each run of the processing engine.

Other features of the RUM web interface worth mentioning include the ability to open the various graph and report screens into new windows, an interactive user manual, the automatic refreshing of any screen where the data might change with each new network

sample, and the ability to retrieve all report information without any HTML formatting. The latter becomes important when extracting data from the RUM system to be further processed by another application. Several screens make up the web interface. Each of the main screens that allow users to interact with the data collected from the RUM processing engine are described below. Screenshots of each of these components can be found in Appendix A.

3.3.1 Graph

Graphs in the RUM system provide the easiest way to get a quick synopsis of what is going on in the network. During the processing function, inbound and outbound totals for TCP traffic, UDP traffic, and other IP traffic are recorded for each subnet using RRDtool. Totals are kept for bits/second, total number of packets, and total number of flows. The user is able to select any one of these metrics as the basis for the graph. Also, the web interface allows the user to create graphs showing this data for either the past 24 hours, the past week, the past month, or the past year.

There are three basic graph types. The "totals" graph shows the total inbound and outbound values in the background of the graph with a line representing each individual value superimposed on the background. Data from each selected dataset is combined in this graph. This representation is good for determining the overall total values for a given metric or comparing inbound versus outbound. The "networks" graph stacks the values of each dataset on top of one another for both inbound and outbound. This graph is useful for making comparisons between each actual dataset. The third and final type of

graph, the “IP protocols”, compares inbound and outbound TCP, UDP, and other IP traffic. Data from each selected dataset is combined on this graph.

3.3.2 Report

Reports are another way in which data can be extracted from the RUM processing engine. Basically, reports provide a means of displaying the data collected on each internal host, external host, web server, multicast address, and application. The top entries are reported first and data can be sorted based upon bits/second, number of packets, or number of flows. Other reports available include which subnets, VLANs, or networks have the most activity.

New to the most recent version of RUM, is the ability to further parse the report data by specific traffic parameters. A minimum and maximum value for bits/second, number of packets, or number of flows can be set. This is useful for setting threshold limits to be used for triggering alarms as further explained in section 3.3.4.

3.3.3 Packet Search

The packet search is a means to examine the actual dump file of each dataset. Simply selecting a dataset, or group of datasets, will just return the actual dump file from the last packet sample. Through the use of search parameters, the dump files can be further inspected looking at specific settings for either source IP, destination IP, IP protocol, source port, and/or destination port. Like the reports, the packet search can be installed as a trigger event as explained in section 3.3.4.

Earlier versions of RUM contained a similar feature, however instead of searching through packet data, the search was conducted on the network flows for a particular group of data. This was because the packet data were merged into flow data early on in the analysis process.

3.3.4 Trigger

This functionality is new to the version 1.1 series of RUM. Using previously created reports or searches, a user can install the selected query into the system such that it is run automatically at the end of every iteration of the RUM processing engine. If the query returns any positive match, then the appropriate action is triggered. When installing the trigger, the user has the option to select from one of several possible actions to take upon a positive match. Actions include alerting the user through email, sending a notice to another host/port pair, simply logging the data, or a combination of alerting the user and logging the data. Since text-based emails can be sent to pagers, the RUM system is able to effectively alert users when a specific threshold is met. The email option can be configured to either send just a notice detailing which threshold has been exceeded, or the actual report listing of data that matched the query criteria. Through the log feature, a notice can be sent to a user, allowing the user to examine the logs at a later time and determine exactly what happened. Also, for some triggers, the user may wish to select no notice at all and just routinely check the log for each trigger. When selecting to log data, the user may also select the duration of time that data is stored by the RUM system. The default, and maximum, amount of time is 30 days. For security and performance reasons, each user can only install a limited number of triggers. The RUM administrator determines the actual number.

3.3.5 Log Report

Log reports allow the user to examine data that matched true for a particular trigger. Since trigger functionality is new in RUM version 1.1.x, logging functionality is new as well. Hard coded limits were built into an earlier version of RUM, however both the setting of the limits and the display of the data were not customizable at all. Data can only be examined for triggers that were set up to log their results. Also, data will only exists as far back as the duration set when installing the trigger. Log results can be displayed in several different ways depending on the type of log. One possibility is displaying the contents of the entire log file chronologically. If the log file contains the results of a report, the log file can be further processed to combine statistics for each host or port. Analyzing report logs also allow for the calculation of high water marks for each host or port.

3.3.6 Host Lookup

This is another new feature included with the latest version of RUM. Using results from the research done at the Honeynet Project [31], the RUM system attempts to passively identify each host. Using packets sent by the particular host, the IP type of service, IP time to live, IP don't fragment bit, and TCP window size are examined to produce the packet's signature. If a host has no source activity during a particular sampling period, then RUM is unable to attempt to identify it. No persistence data about the type of packets sent by a host are kept. Each packet signature is compared to the database of know signatures, also provided by the Honeynet Project, to attempt to discover a match [31]. Since a host may have multiple signatures, it is possible for multiple matches for any given host. The host lookup is not 100% accurate, however it does provide a

possible insight into the identity of a particular host. The percentages of total source packets that match each signature are provided to help the user determine the accuracy of a given match.

3.3.7 Customization

In order to allow the user to conveniently examine the data provided by RUM, certain characteristics can be set by each user. Alternate labels can be given to any datasets, whether it is a subnet, VLAN, or network. Each user will then see their label, instead of the default name in the reports and graph keys. The other feature that can be customized is the color displayed for a particular dataset in the "networks" graph. In order for the user to easily recognize the different graph colors, each user is able to manually assign particular colors to each dataset. Then, every time a graph is constructed using that dataset, the same color will be used.

3.3.8 Performance

Also new to the latest version RUM is the ability to view the performance of the RUM processing engine from the web interface. During each run of the processing engine, the estimated sampling coverage along the elapsed time for each step of the pipeline is recorded. Through the interface, two types of graphs to explore these values are available. One graph shows the estimated sampling coverage for either the past 24 hours, 1 week, 1 month, or 1 year. The other graph shows the time it takes to complete each section of the processing routine. For this graph, the time values are stacked atop one another to also demonstrate the overall time it takes to capture and analyze the network

data. Comparing the results of these graphs to the current network levels will permit the user to make sure the RUM system is performing at an acceptable level.

3.4 Setup & Configuration

RUM is able to display usage statistics for a high-bandwidth network. In order for the system to perform optimally, some setup and configuration is required. This section describes the hardware, software, and its configuration in order to achieve proper operation.

3.4.1 System Hardware

In order to capture packets from a high-bandwidth network, analyze them, and then store the data away before starting the process all over again, a decent amount of hardware power is required. Minimum hardware requirements are based on the performance of the test machines. Table 3.2 shows the setup of each test machine. While both machines are able to capture 100% estimated sampling coverage using RUM version 1.1, test machine A performed better overall than test machine B. The major difference between the two machines is the type of processors. Not surprisingly, a faster, better processor results in better performance. As part of the RUM processing engine code is threaded, more processors also provide better performance. Because all packets are stored in memory before being written to disk, a lot of memory is required for the RUM system. For example, 1 Gb has provided sufficient for the test network, but this requirement is likely to grow as sampled bandwidth grows.

	Test machine A	Test machine B
Server Type:	IBM xSeries 340	IBM Netfinity 4000r
Processors:	Dual 1 GHz Intel Pentium III	Dual 750 MHz Intel Pentium III
Memory:	1 GB ECC SDRAM	1 Gb ECC SDRAM
Hard Drives:	Six 18.2 Gb Ultra SCSI	Two 9.1 Gb Ultra SCSI
Network Cards:	10/100 Mbps integrated Ethernet, IBM Gigabit Ethernet Adapter SX	10/100 Mbps integrated Ethernet, IBM Gigabit Ethernet Adapter SX

Table 3.2. Hardware specifications of test machines

3.4.2 Probe Placement

RUM monitors network traffic flowing between an internal network and the outside world. Ideal placement for the network probe is at a gateway location on the network where internal-to-external traffic and external-to-internal traffic can be monitored without seeing any internal-to-internal traffic. Any internal-only traffic will actually increase the amount of processing required for each sample. In situations where this type of traffic constitutes a large portion of the overall traffic being monitored, performance of the tool may be severely affected. As the probe omits broadcast traffic, any internal-to-internal traffic between switches is insignificant and need not be a concern.

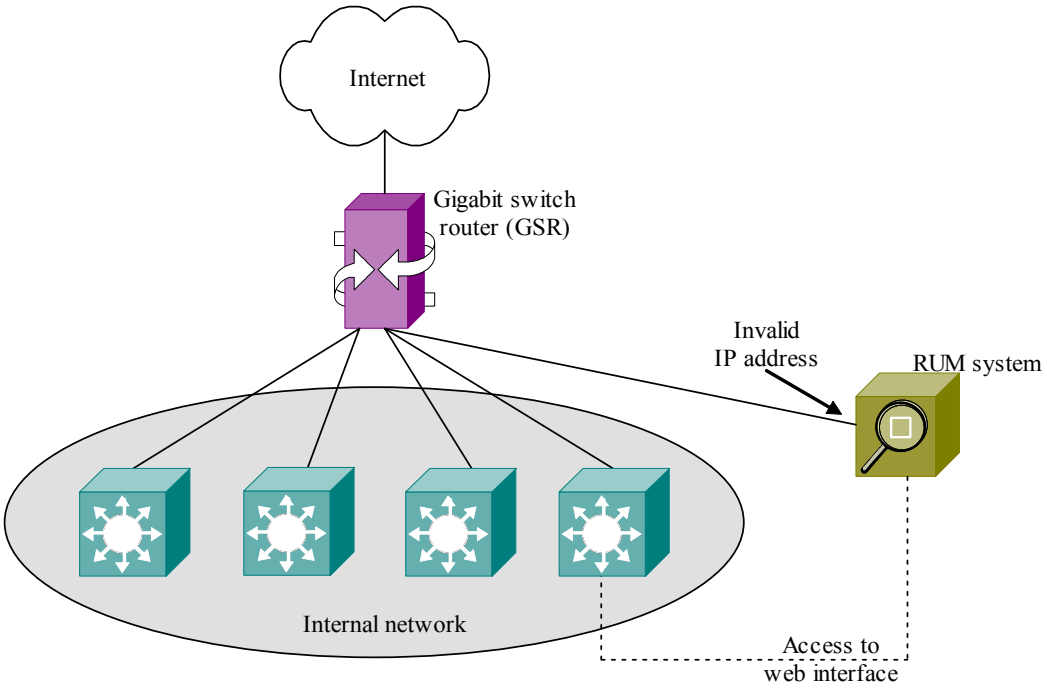


Figure 3.4. Probe placement

Figure 3.4 shows the probe placement in the test environment. Between the Catalyst 6509 gateway and the internal network, the flow of traffic is divided using a fiber splitter. This enables a complete copy of the network traffic to be diverted to the 3508 switch. Traffic is spanned across the ports of the 3508 switch such that each of the test machines sees the same traffic. As the network interface card (NIC) capturing network traffic does not have a valid IP address, there is no need to worry about the probe being in the demilitarized zone (DMZ). While traffic may be captured from this NIC, the only access to the monitoring machine is through the other NIC that is located in the internal network (noted by the dotted line in figure 3.4).

There are situations where spanning gateway traffic to a separate port solely for monitoring purposes is impractical. When the probe must be placed at a location on the network where it will see internal-to-internal traffic, RUM may be configured to not capture this type of traffic, unless this is wanted. The default filter expression in RUM captures all IP traffic. Changing this expression to only capture traffic that does not have both an internal source IP and internal destination IP will enable the probe to appear like it is collecting traffic at the gateway. This does of course have performance drawbacks at high-bandwidths, due to the extra low-level processing, however this scenario will most likely only exist on a less-loaded network anyway.

3.4.3 Dependencies

The RUM system was designed for use in the Linux environment. RUM has been tested on Red Hat Linux 7.1 with the 2.4.x kernel. Since all of the code is written in either C or Perl, other operating systems (OS) may work as well. The location where RUM extracts statistics about the NIC may be OS specific code. Section 3.2.1 describes how RUM estimates how many packets should have been captured using these results. This filename and location may need to be changed if using another OS. Other than requiring a C compiler and Perl distribution, RUM makes use of the Perl Storable module and the RRDtool utility. The Perl storable module has begun shipping with the 7.2 release of Red Hat.

In order to access the web interface, some type of web server is required. Apache 1.3.x is what is being used on the test machines. The web server must handle user authentication. RUM uses the REMOTE_USER variable of the web server to determine

which datasets a user is allowed to access. Lack of a user name would return zero datasets. Other options such as setting the web server up as a SSL server or with modperl are not required for RUM operation. The web server does needs to be able to hand Perl CGI scripts though.

3.4.4 Deployment

Installing RUM is relatively straightforward. Makefiles and install scripts are included with the distribution. After installation, the system needs to be configured before it can be used to monitor the network. Configuration includes updating the list of subnets, their respective subnet masks, and each VLAN and network to which the subnet belongs. All other configuration settings such as the network interface to sniff on are located in the RUM global configuration file. Once those changes have been made, RUM is ready to be run. RUM does not automatically routinely monitor the network. The actual RUM engine is just one cycle of the process. In order to regularly monitor network data, the processing application needs to be added to the system's crontab.

4. Security

Several features of the Resource Usage Monitor contribute to its usefulness as a network security tool. The graphs alert users of problems with different regions of the network. Further examination of the reports indicates the hosts responsible for the irregularities seen in the graphs. Since general reports only contain data from the most recent sample, log reports provide this same information when investigating previous network data. Once general network behavior has been determined, thresholds can be set, causing the RUM system to automatically alert the user when a possible security threat is occurring. Using these features, several different security concerns can be successfully detected with the RUM application. Also, as the system collects statistics on both inbound and outbound traffic, it cannot only detect security threats against the internal network, but also security treats originating from within the internal network. New to version 1.1, RUM has the ability to collect statistics on packets/sample for each host, application, etc. Since there has not been sufficient time to draw conclusions from this type of analysis, the uses of RUM will just focus on the load (bits/second) and flow analysis.

4.1 Anomaly Detection

Of the two basic types of security analyses, misuse and anomaly detection, RUM is capable of the latter. Misuse detection requires a current database of attack signatures for which packets can be matched against. A popular open-source misuse detection application, Snort, contains several hundred attack signatures [28]. Since RUM does not collect any statistics on the user data within each packet, the majority of these signatures simply would not work anyway. To carry out the extensive state-based processing of misuse detection, efficient pattern matching is required [29]. This analysis usually comes

in the form of an exclusive specification language and compiler. Even without misuse detection though, RUM is a very useful anomaly detection application. In fact, completely new forms of attack can only be detected through recognition of generalized statistical anomalies rather than attack-specific fingerprints. From this point of view, it can be argued that RUM may be even more interesting than misuse detectors since its function is pro-active, rather than reactive.

Anomaly detection involves the identification of behavior that deviates from the standard [17]. It is based on the hypothesis that illegal exploitation of a system involves abnormal use of the system resources [8]. Thus the occurrence of abnormal use indicates possible security threats. Since, for anomaly detection, no attack-specific predefined knowledge is needed, anomaly detection works well for detecting new attacks, for which attack-specific signatures have yet to be devised. Using overall graphs from either the entire network, or a specific region of the network, general usage behavior for the network can be determined. Fig 4.1, and 4.2 show daily usage graphs of the overall test network measured in bits/second and flows/second, respectively. From the first graph, the normal network load appears to be 160 Mbps outbound and 85 Mbps inbound. For an open-environment, a campus network such as the test network, uneven load isn't a surprise. Unidirectional peaks in load graphs are often characteristic of certain types of traffic. One-directional multimedia streams, such as video, are a good example of traffic that might create a load spike in one direction. On the other hand, inbound and outbound network flows are usually equal if the application uses TCP. Figure 4.2 shows that on average there are 700 flows per second in each direction. These values should be consistent with one another because most of the time the network operates in a request

and serve model. An inbound request equals an outbound service while an outbound request equals an inbound service. Of course, there are asymmetric types of traffic on a network; but in an environment that uses reliable protocols (such as TCP), a flow in one direction will be met with a flow in the opposite direction.

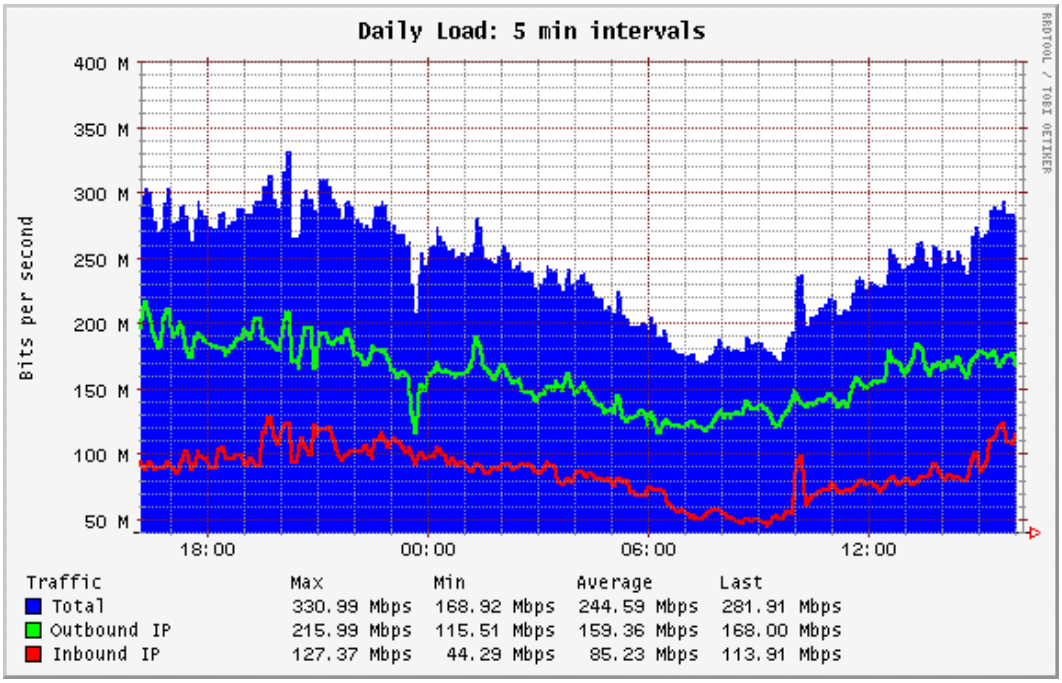


Figure 4.1. Daily bits/sec on test network

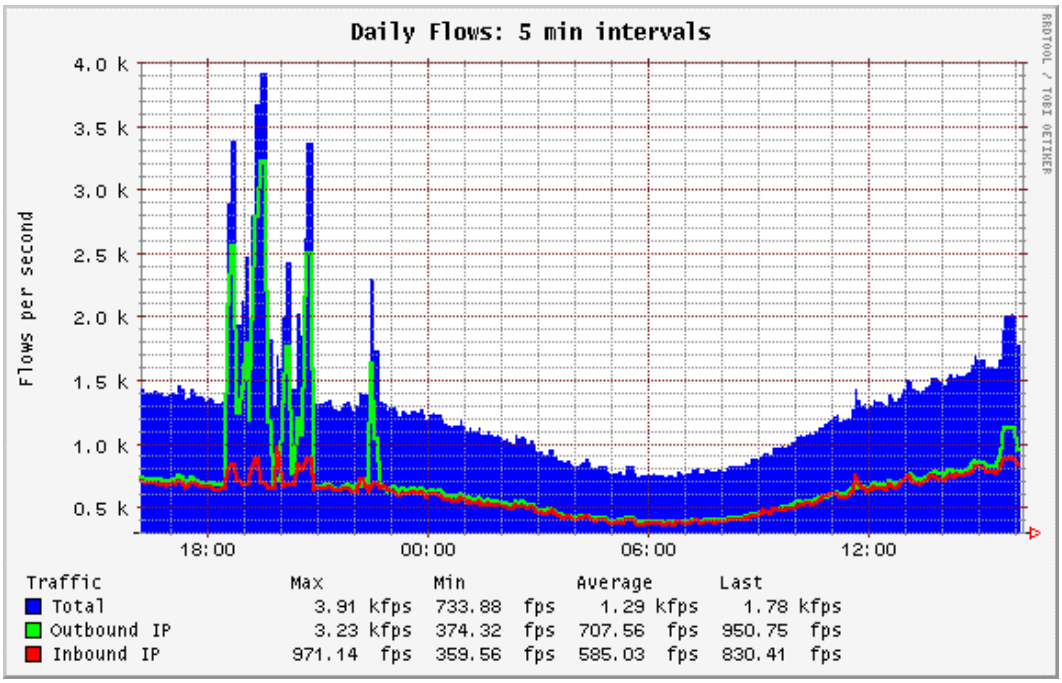


Figure 4.2. Daily flows/second on test network

Another piece of data vital to determining the normal usage profile of a network is the protocol distribution graph. Figure 4.3 and 4.4 show the IP protocol load and flows on the test network. Notice that, while TCP traffic makes up the majority of the network load, TCP doesn't maintain as much of a majority of the traffic for network flows. While protocol distribution is again network dependent, it does remain constant, allowing for a normal behavior to be determined. For example, while figure 4.3 is asymmetrical with respect to load, it does not appear to be unusual. On the other hand, figure 4.4 shows several sharp and asymmetric spikes on an otherwise symmetric background. This seems to indicate that there were anomalies in the outbound flows in the 10am to 1pm timeframe.

Anomaly detection is possible after deciding on what constitutes the normal operational profile (at least statistically). Each network will have its own profile. For a network that has the same number of users day in and day out, the normal operating levels may be determined in just a couple of days. However, when the network characteristics change throughout the year, so will the "normal" behavior. Figure 4.5 shows yearly data collected on the NC State campus gateway (our test network) for the past six months.

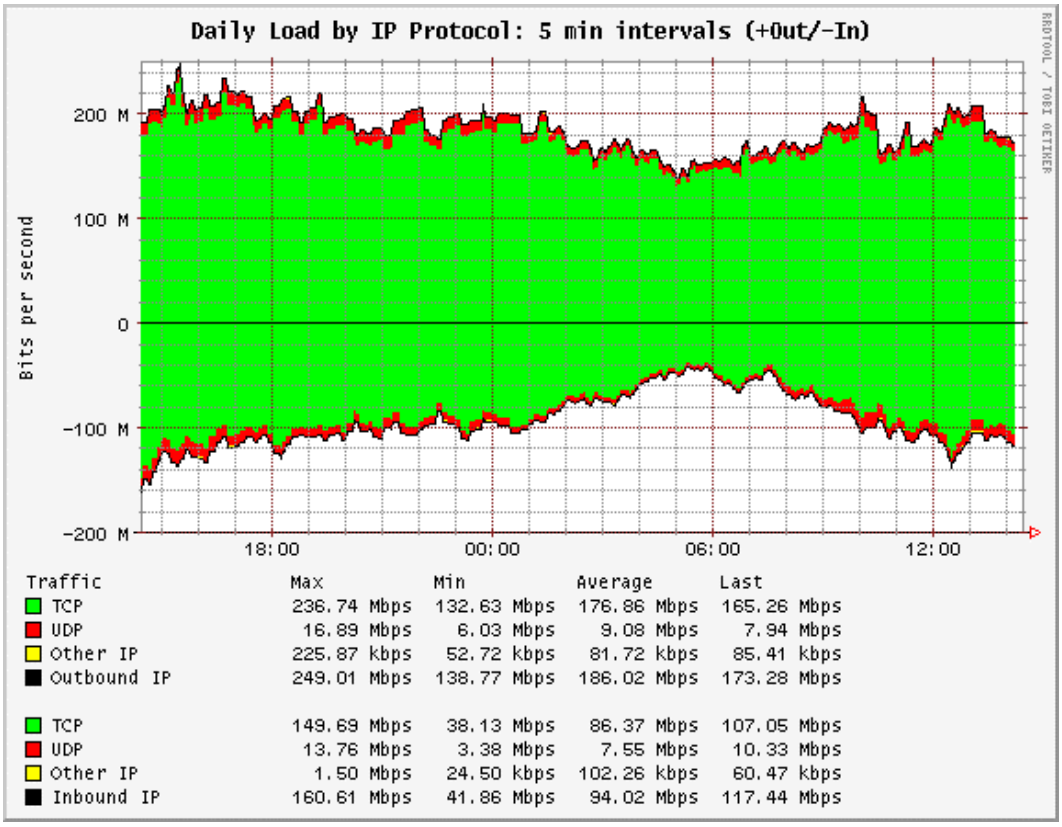


Figure 4.3. Daily IP protocol load on the test network

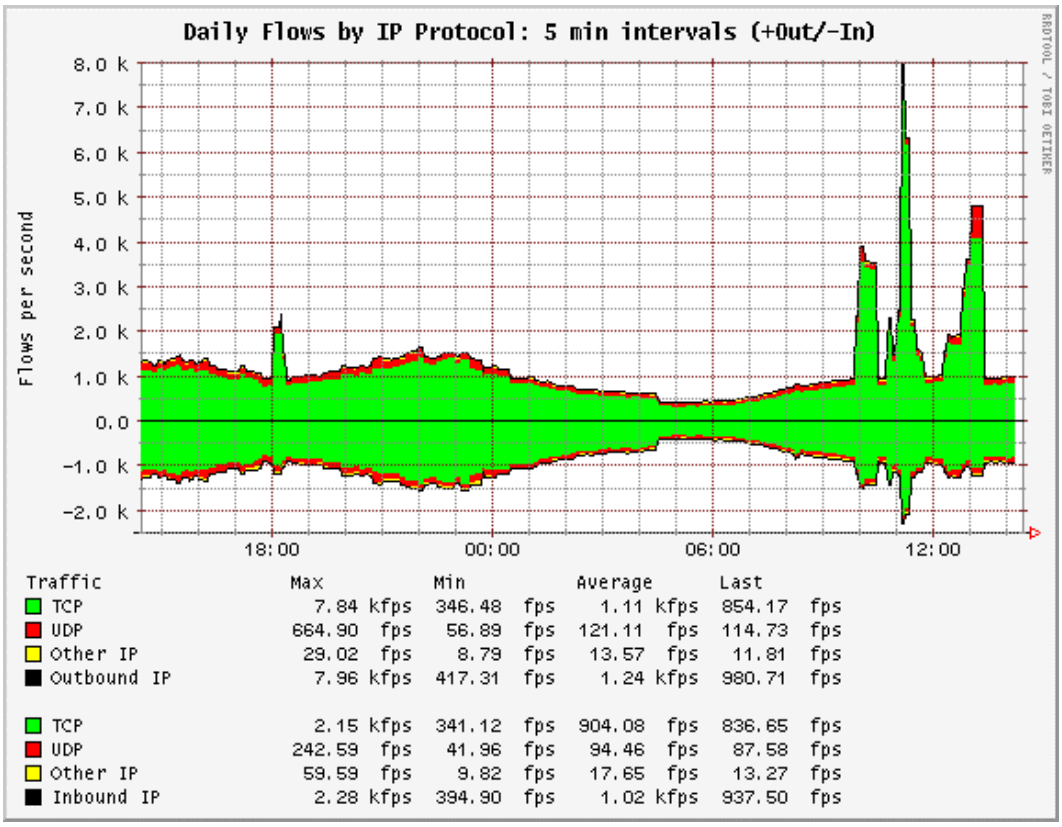


Figure 4.4. Daily IP protocol flows on the test network

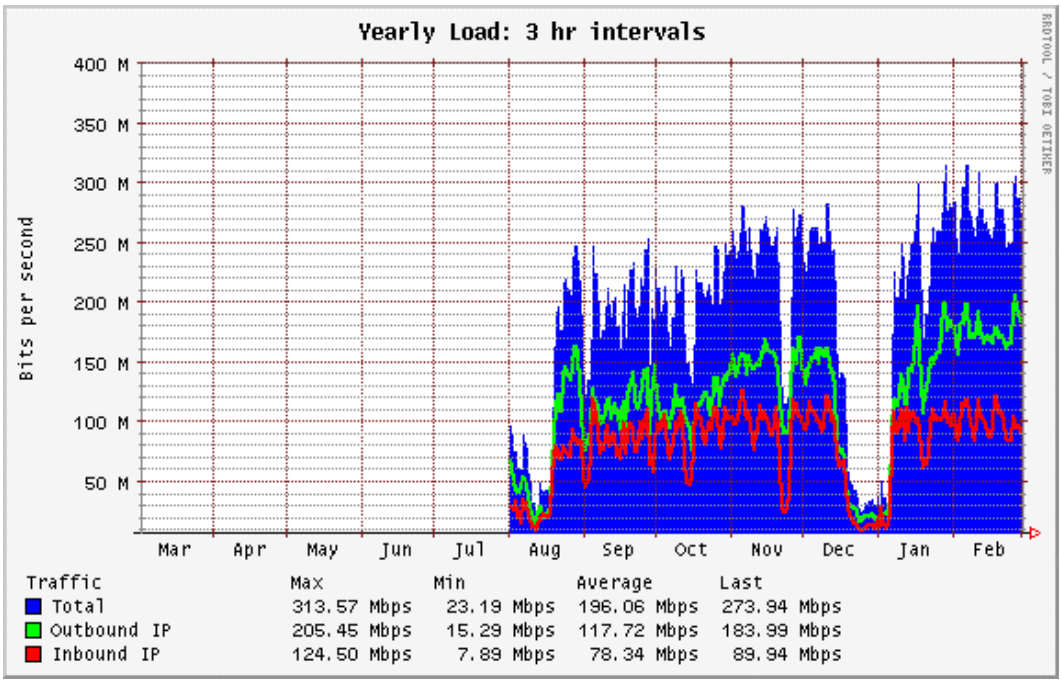


Figure 4.5. Six month load on test network

"Normal" behavior in the month of August was less than 100 Mbps. However once students came back to campus for the Fall semester, "normal" network load shot up to 250 Mbps total traffic. Similar changes in rates were observed for flows. It is even possible to note each of the university holidays on the graph when the majority of student users were away from campus. Labor Day weekend is shown as the first dip in September, Fall break is shown as the dip in the traffic in the middle of October, the Thanksgiving holiday can be seen in November, and then the drop in traffic in late December notes the longer Winter break. Basically, in order to create a reliable profile of the network, the RUM user needs to also possess a clear understanding of the expected traffic changes on the network under review.

After determining the "typical" dynamic behavior of the network parameters, anomaly detection involves looking for (statistical) abnormalities (or deviations) in any of the traffic metrics. These anomalies could include spikes in any one of the traffic parameters, dips in traffic, or even asymmetric traffic. Of course the granularity at which the network is examined comes into play. On small network segments, or small overall networks, spikes in network traffic can be common. This variability and "noise" is just part of normal network traffic patterns. True anomalies can only be detected when enough data is available to ascertain whether or not the observed deviations are statistically significant with respect to "normal" behavior.

4.2 Port Scans

There are several different types of port scans, all which may threaten the security of a network. One host could send a packet to every port on a given host, looking for holes. Another option is to send a packet to a specific port on each host in the network. As these type of scans tend to be easier to fingerprint and detect, a stealthier scan will involve the same process but with a longer interval between packets. Other scans involve the use of several different machines to probe a host or multiple machines on a network to further disguise the type of attack. Nonetheless, all of these different types of scans generate more traffic than normal [9]. There is no utility built into RUM to specifically identify port scans, however using a combination of the graphs and reports, it is possible for a user to identify this type of threat.

For the more classic type of scan, where every port on a single machine or ports on multiple machines are all hit in sequence, there will be a noticeable spike in the amount

of destination or source host-specific network flow in the traffic. Reports indicating the top traffic producing/receiving hosts can trace the problem to a single host. Further examination of the host's packets will reveal the series of packets that make the port scan. For stealthier scans, discovery becomes harder as the traffic does not significantly increase over any given one minute sample, no anomalies will show up in any of the graphs. However, this type of security threat can be recognized by examining the actual RUM reports over a longer period of time (e.g., persistence or repeats of one-host connections to different ports - sorting by source host and port - etc.). On a large network, the majority of stealthier scans will go unnoticed unless the individual traffic data is explicitly scrutinized for "scan" fingerprints.

4.3 Denial of Service

Denial of Service (DOS) attacks involve the flooding of packets against a particular host. As the goal of the attack is to send several packets within a small timeframe in order to bring the victim machine down, the spike in traffic can be easily seen in the network graphs provided by RUM. Since DOS attacks normally involve several small packets with the SYN flag set, the change in network load will most likely not be noticeable. The spike in traffic will occur in the network flow graph, as there will be many more flows to the victim host, than back to the attacker (a similar pattern, but in reverse occurs when a compromised campus machine mounts a massive scan of outside hosts, or tries to attack an outside host using DOS techniques). The spike from this attack can also be seen in the packet graph of the network.

The network reports enable the RUM user to determine the attacker(s) and victim in this type of security threat. A simple query for the top load and/or flow talker or top receiver in a particular network region should produce the host either producing the denial of service attack or the host being bombarded with the influx of packets. While this type of attack is easier to detect than a port scan, it still requires analysis of the traffic data after noticing the network anomaly in the graphs.

4.4 Trojan Applications & Viruses

A spike in network traffic to or from a particular port of a host could also indicate the presence of a trojan horse application or a virus. These applications run in the background of the victim's machine. They either open up holes for the attacker to gain access to the machine, or in trying to spread and infect other machines, produce a significant amount of outbound traffic.

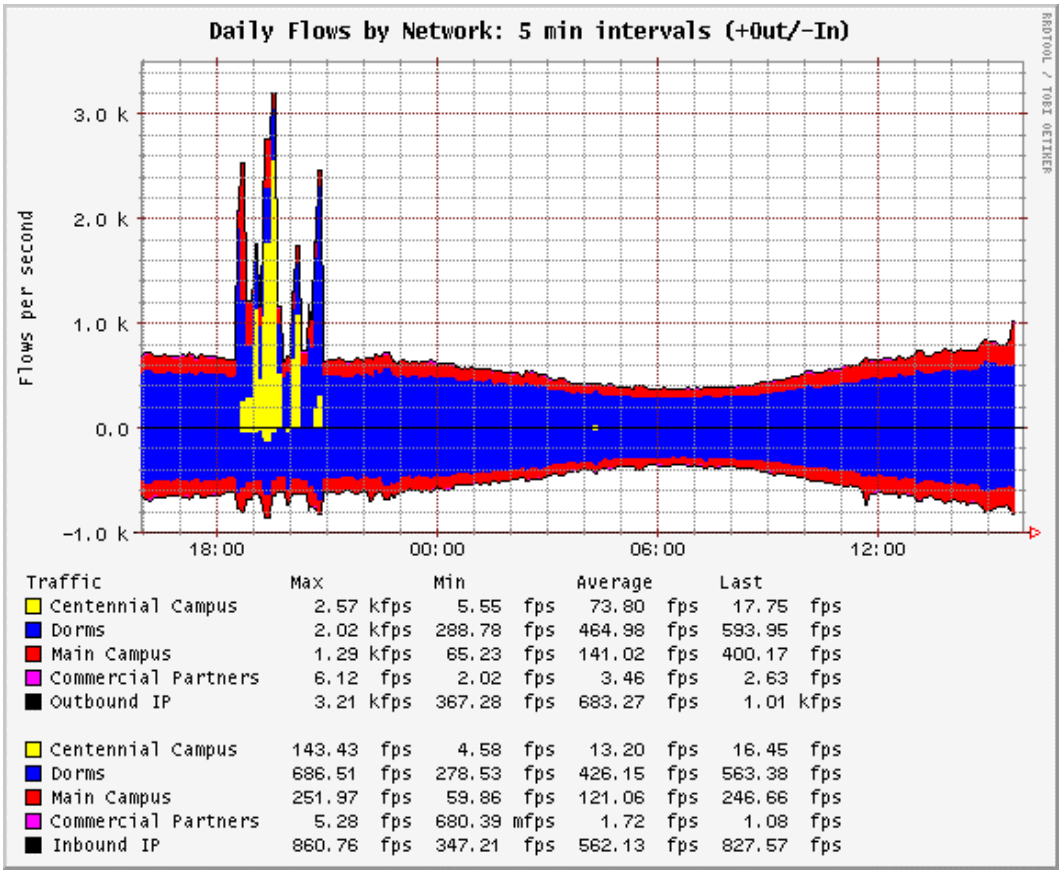


Figure 4.6. Daily graph showing spike in traffic flows

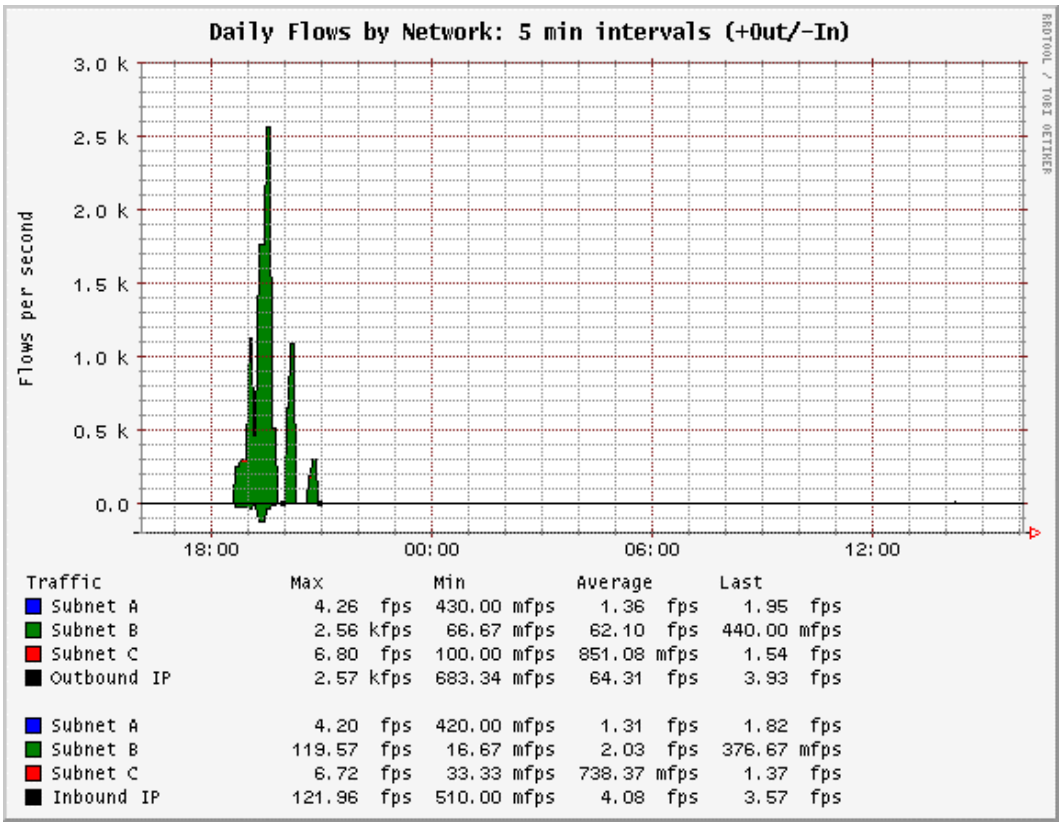


Figure 4.7. Closer examination of spike in traffic flows

Figure 4.6 shows a several flow spikes for the overall flows on the tested network. Further examination produces Figure 4.7 where the traffic can be pinpointed as originating from subnet B. Upon examining the report for top traffic-producing hosts in this subnet, the host causing the problem is identified. Explicit examination of the packet header structure from this host revealed that the host was sending packets to the same port on different target hosts. While this type of behavior could be mistaken for a port scan, it can also be an attempt of a virus to spread. An examination into the problem machine (and of its traffic payload) revealed this host was actually an infected victim trying to spread the infection.

4.5 IP Spoofing

The Resource Usage Monitor has no mechanism for detecting this type of security threat except when the spoofed addresses are "illegal," i.e., out of IP range in which the tested network is supposed to be in. IP spoofing involves the altering of the source IP address from its actual value within a packet. The only 100% certain way to detect IP spoofing is to notice two or more IP addresses mapped to a single Ethernet address. However since Ethernet addresses change as the packet travels between broadcast domains, it is impossible for a network tool such as RUM to identify this problem explicitly. RUM is positioned at the network gateway, where the original Ethernet header has long since been changed for the majority of the internal packets.

However, what RUM can detect are "illegal" addresses, unregistered addresses with excessive load or flow characteristics, and similar anomalies. Often, "illegal" addresses are the result of machine misconfigurations, but sometimes they are the result of a machine that is conducting an attack and is trying to masquerade (albeit incompetently) using another IP address. In recent times, private addresses (in the 192.168.x.x range and 10.x.x.x range) have been cropping up as "illegal" addresses when misconfigured DHCP servers appear on the open campus network.

5. Quality of Service (QoS) Evaluation

The graphs and reports from the Resource Usage Monitor provide useful data for network QoS assessment. Quality of service involves the measurement of network characteristics that impact end-to-end performance of an application, improving upon them, and possibly even guaranteeing a particular resource (e.g. bandwidth) for certain types of traffic. Since RUM does not interact with the network, the reaction to network traffic statistics must come from other systems. RUM however, has many features capable of providing network statistics that act as a basis for making network management decisions. This allows the Resource Usage Monitor to play a vital role in quality of service evaluation. Indeed, on an experimental basis, version 1.1. of RUM is being used in conjunction with a policy control server to affect and control QoS.

5.1 Host-based

Often, end-to-end quality of service involves the performance of a particular host machine, its operating system, the application of interest, and the particular network streams associated with that application. For machines that play a vital role in a network, such as a web server or application server, their reliability becomes that much more important. In order to make a decision about resources needed for a particular host a user first needs to understand the normal activity for that machine. The most basic way to start gathering information about the network component of the usage statistics for a particular host is through the host reports in RUM. Table 5.1 shows the first five entries from one such report. These reports list the top traffic producers or receivers. Several metrics are listed for each host, such as the load, the number of packets, the number of flows, the percentage of the overall traffic this host accounts for, etc. While this data is

useful, if the host is not very active during the particular sampling interval, it may be harder to find it in the list.

Num	IP Address	Hostname	Bits/sec	Total Flows	Flows/sec	%Total Load(1)	%Total Flows(1)
1	152.7.61.236	Sull-12-236.rh.ncsu.edu	89575	6525	109	0.06	4.14
2	152.7.61.133	Sull-11-133.rh.ncsu.edu	88843	4513	76	0.06	2.86
3	152.7.61.102	Sull-10-102.rh.ncsu.edu	63782	4446	75	0.04	2.82
4	152.7.61.106	Sull-10-106.rh.ncsu.edu	63279	4425	74	0.04	2.80
5	152.7.61.121	Sull-10-121.rh.ncsu.edu	52960	4167	70	0.03	2.64

Table 5.1. Top outbound traffic producers

In order to examine traffic data for a particular host over time, triggers can be set up for a particular host's IP address. Any packets to or from that host are logged. They can then be collected for review at a later time, or this data could be off loaded to another machine for processing. This provides much more information about the host, about its ports, ratio of its incoming to outgoing flows on a particular part, or stream, etc. By using the log report function of RUM, the high-water mark for a particular host can be determined for up to the past 30 days. Once the amount of traffic flow/load to and from this host is known, better decisions can be made about the type of network resources that it requires and about its QoS needs. For example, RealVideo streams tend to be both UDP and TCP based. However, the "main stream" is unidirectional (streaming in or out) and the return, or "feedback" traffic stream (e.g., acknowledgments or re-transmission requests), is usually an order of magnitude or more smaller. If that incoming to outgoing ratio is known for a "good quality" stream, it is possible to recognize, just by observing the network flows, QoS deterioration of that stream if the main to feedback stream ratio becomes smaller than expected.

Another function of RUM is to discover which hosts on the network are responsible for stealing or hogging bandwidth. As file sharing applications become more popular, more and more hosts are beginning to use them both intentionally and unintentionally. This traffic can often negatively impact normal business flows. For example, sometimes machines designated for office may have file sharing applications (such as KaZaA) running on them in the background. This may turn each of these normally low-traffic machines into high-volume servers. Through RUM, reports can be generated to indicate which machines routinely exceed a certain threshold of traffic. Thresholds can be set for the number of bits/second, packets/sample, or flows/sample. In working with the test network, basic thresholds of 1 Mbps and 300 flows/sample were used. The latter often indicates a legal server, a machine that is compromised, or a machine that may be misusing resources intentionally. Since it is possible that any machine can exceed a traffic threshold during any given sampling period (traffic bursts) as part of its normal operation, more reliable indicators of problems come from setting up triggers to catch all violating machines for an extended period of time, thus averaging out the bursts. Those machines that show up more often, and don't fall into the list of machines expected to be there, are the ones that need to be examined.

RUM can provide an additional insight into the packets being sent to and from each machine through its host logs. Table 5.2 shows the flows listing for one active host in the test network. If a number of different machines are communicating with the same port number on the violating machine, then most likely it is acting as a server. The course of action taken against this machine depends on the network policies and is outside of the scope of RUM.

Num	IP Protocol	Source	Destination	Bits/sec
1	tcp	160.39.236.136.2261	152.7.60.154.1144	2620707
2	tcp	80.11.207.207.1900	152.7.60.154.ftp-data	7636
3	tcp	62.57.119.72.2068	152.7.60.154.ftp-data	3793
4	tcp	213.16.147.233.3781	152.7.60.154.2807	1817
5	tcp	216.226.129.211.ircd	152.7.60.154.1133	825

Table 5.2. Packet listing

5.2 Port/Application-based

Another valuable metric for quality of service evaluation involves the examination of traffic for specific port activities within a network. Applications such as multimedia video or audio use a particular port for all of their traffic. As mentioned earlier, watching the traffic for a particular stream (usually associated with a particular port) allows the RUM user to make QoS decisions about each application's traffic.

Num	IP Protocol	Application	Port	Bits/sec	Total Flows	Flows/sec	%Total Load(1)	%Total Flows(1)
1	tcp	http	80	21878076	16370	273	20.68	33.29
2	tcp	kazaa	1214	19568467	11304	189	18.49	22.98
3	tcp	aim	4099,5190	601234	6696	112	0.57	13.62
4	tcp	gnutella	6346,6347	2799703	2046	35	2.65	4.16
5	udp	domain	53	60304	1814	31	0.06	3.69

Table 5.3. Report of most active inbound application

The above table shows a listing of which applications have the most usage in the test network. Port numbers are resolved to specific applications through the mapping in the /etc/services file. Just like the host reports, this report lists the amount of traffic by bits/second, packets/sample, and flows/sample for each application during the sampling period. Again, if persistence data is required, triggers can be set up watch a particular

port number or application during each sample. Once the data has been collected, 30-day high water marks, and other parameters, can be extracted to better understand usage behavior of each application on the network.

Often quality of service evaluation involves the linking of a specific application to a specific host. For example, not every host on the network will be a video server. The same packet reports used to watch a particular host or a particular application can be used to watch a host and port combination. This allows for RUM to provide even more concise data for making QoS decisions.

5.3 Network, VLAN, & subnet-based

The same reports available for hosts and applications are also available for the various collections of data. As mentioned earlier, a VLAN consists of a collection of subnets and a network is a collection of VLANs. Figure 5.1 shows the daily load for each class of the test network. From this graph, it is easy to see where the majority of the network resources need to be focused. RUM allows for similar graphs or reports to be generated for comparisons between VLANs and subnets.

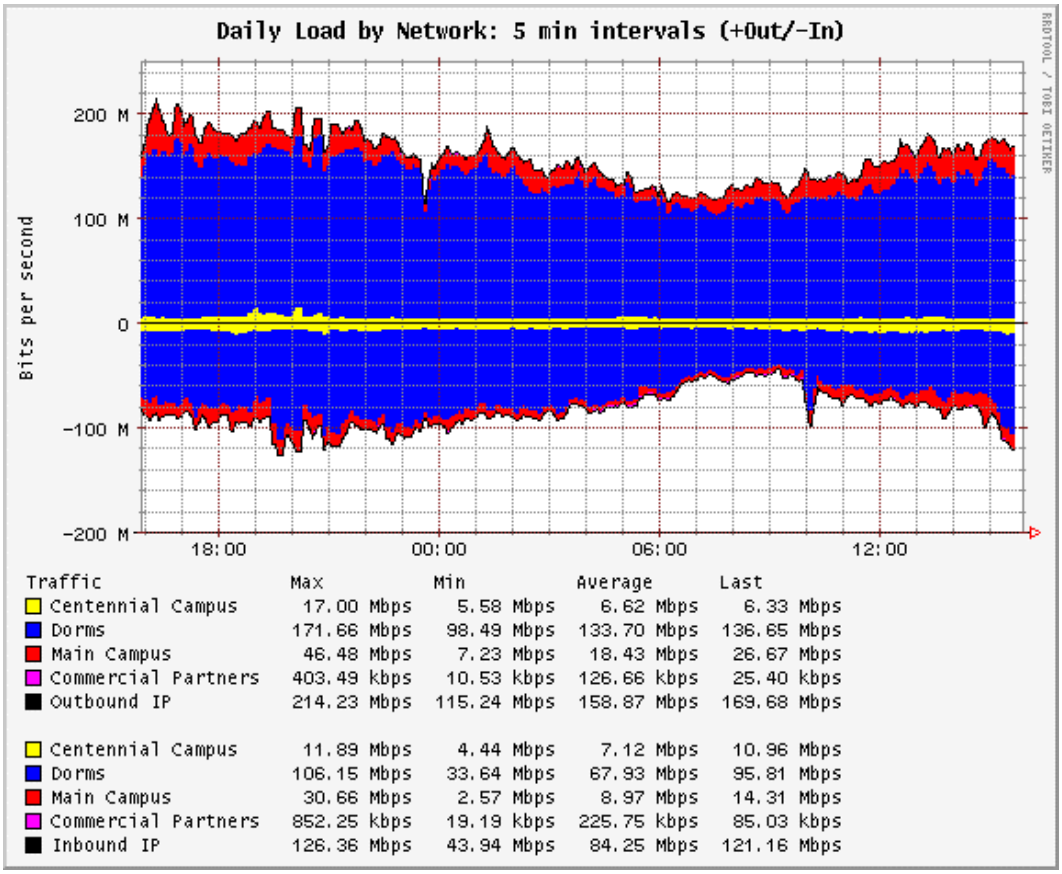


Figure 5.1 Network comparison of traffic load

Above are just examples of some of the many possibilities of how to use RUM to identify both security and QoS issues. In combination with a QoS policy server and a QoS controller, RUM becomes a valuable tool for delivering high-quality end-to-end services.

6. Conclusions

The principal objective of the work presented in this thesis was to develop an inexpensive, customizable network-monitoring tool capable of analyzing a high-bandwidth network. The Resource Usage Monitor possesses a processing engine and web interface providing statistical data analysis at different granularities throughout the network. While in many ways it is similar to a number of other tools available on the network, its main advantage is that:

a) it was designed for tapping into high-flow high-load networks, and

b) that it was designed for compartmentalized and privacy-conscious monitoring of the network flows suitable for support of a number of different LAN administrators.

6.1 Security

Anomaly detection is used for security analysis within RUM. Using the network graphs and reports, normal usage profiles can be developed. Part of the anomaly detection involves looking for deviations from these regular behaviors. Use of the trigger functionality is required in order to make the security detection process more automatic. Data from RUM is useful in identifying port scans, DOS attacks, and trojan applications within a network environment.

6.2 Quality of Service

The traffic characteristics of specific networks, hosts, and applications or ports provide the data for quality of service evaluation within RUM. Network graphs combined with reports detailing traffic to/from a given host provide IP accounting information for the

particular host. Similar reports can be generated to produce IP accounting data for each application or port number. This becomes even more useful for monitoring high-profile machines such as web or application servers, or some of the multimedia services such as streaming video.

6.3 Future Work

Improving the efficiency of the RUM processing engine is an ongoing process. While significant advances have been made through initial development, there is still opportunity for improvement. The processing engine is currently written using a combination of C and Perl. Moving all of the code to C and implementing new data structures for the storage of data from each sample should have a profound effect on the performance. Another way to improve the performance of RUM is through setting up a system of distributed probes throughout the network. Not only would this allow for more analysis points in the network, but also some of the processing could be done at the remote probes before the aggregation of network data. Other areas of future work involving RUM include its interaction with other applications and strengthening of its security and QoS assessment modules. With the added functionality of triggers and data off-loading, RUM can be the basis for policy-servers and controllers or other analysis engines.

7. References

[1] D. Backman. “Sniffer Now Does Windows”, *Network Computing* 1999 <http://www.networkcomputing.com/1004/1004sp1.html>.

[2] B. Barr, S. Yoo, and T. Cheatham. “Network monitoring system design”, *ACM SIGCSE Bulletin, Proceedings of the twenty-ninth SIGCSE technical symposium on Computer Science education* 1998; Vol. 30, No. 1, 102-106.

[3] D.M. Chiu and R. Sudama. Network Monitoring Explained: design and application Ellis Horwood: New York 1992.

[4] Cooperative Association for Internet Data Analysis (CAIDA). “cflowd - Frequently Asked Questions”, 2001 <http://www.caida.org/tools/measurements/cflowd/>.

[5] Cooperative Association for Internet Data Analysis (CAIDA). “FlowScan Architecture”, 2001 <http://www.caida.org/tools/utilities/flowscan/>.

[6] L. Degioanni, F. Risso, and P. Viano. *WinDump* 1999 <http://netgroup-serv.polito.it/windump/>.

[7] L. Degioanni, F. Risso, and P. Viano. *WinPcap* 1999 <http://netgroup-serv.polito.it/windump/>.

[8] D.E. Denning. “An Intrusion-Detection Model”, *IEEE Transactions on Software Engineering* 1987; Vol. 13, No. 2, 222-232.

[9] L. Deri and S. Suin. “Practical network security: experiences with ntop” *Computer Networks* 2000; Vol. 34, 873-880.

[10] L. Deri, S. Suin, and G. Maselli. “Design and Implementation of an Anomaly Detection System: an Empirical Approach” 2001 <http://www.ntop.org>.

[11] Fyodor. “Remote OS detection via TCP/IP stack fingerprinting”, *Phrack* 1998; Vol. 8, No. 54 <http://www.phrack.org>.

[12] R. Graham. “FAQ: Network Intrusion Detection Systems.” 2000 <http://www.robertgraham.com/pubs/network-intrusion-detection.html>.

[13] J. Haugdahl. “Network Monitor Finally Comes Out of Hiding”, *Network Computing* 1998 <http://www.networkcomputing.com/906/906ws2.html>.

[14] V. Jacobson, C. Leres, and S. McCanne. *tcpdump(8)*. Lawrence Berkeley National Library 1989 <http://www.tcpdump.org>.

[15] V. Jacobson, C. Leres, and S. McCanne. *pcap(3)*. Lawrence Berkeley National Library 1989 <http://www.tcpdump.org>.

[16] R.D. Jenkins. “Why web-based network monitoring? Leveraging the platform”, *International Journal of network management* 1999; Vol. 9, No. 3, 175-183.

[17] W. Lee and S.J. Stolfo. “A framework for constructing features and models for intrusion detection systems”, *ACM Transactions on Information and System Security (TISSEC)* 2000; Vol. 3, No. 4, 227-261.

[18] M Lin. “NetFlow: Product Manager - IOS.” 2001 <http://www.cisco.com/warp/public/732/Tech/netflow/docs/nms_vt_2001.ppt>.

[19] S. McCanne and V. Jacobson. “The BSD packet filter: A new architecture for user-level packet capture”, *Proceedings of the Winter 1993 USENIX Technical Conference* 1993; 259-269.

[20] Microsoft Corporation. “About Network Monitor 2.0”, 2001 <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/netmon/netmon/about_network_monitor_2_0.asp>.

[21] Network Associates, Inc. “Sniffer Total Network Visibility Brochure”, 1999 <http://www.snifferpro.com/tnvbro.html>.

[22] NFR Security, Inc. “NFR Network Intrusion Detection Overview”, 2001 <http://www.nfr.com/products/NID>.

[23] T. Oetiker. “What is MRTG”, 1996 <http://people.ee.ethz.ch/~oetiker/webtools/mrtg/mrtg.html>.

[24] T. Oetiker, “RRDtool Manual”, 2000 <http://www.rrdtool.com/manual/>.

[25] M.J. Ranum, K. Landfield, M. Stolarchuk, M. Sienkiewicz, A. Lambeth, and E. Wall. “Implementing a generalized tool for network monitoring”, *Proceedings of the 11th Systems Administration Conference (LISA ’97)* 1997.

[26] Red Hat, Inc. “Red Hat Linux 7.1 Reference Guide”, 2001 <http://www.redhat.com/docs/manuals/linux/RHL-7.1-Manual/ref-guide/>.

[27] Red Hat, Inc. “Red Hat Linux 7.2 Reference Guide”, 2001 <http://www.redhat.com/docs/manuals/linux/RHL-7.2-Manual/ref-guide/>.

[28] M. Roesch. “Snort - Lightweight Intrusion Detection for Networks” *Proceedings of the 13th Systems Administration Conference (LISA ’99)* 1999; 229-236.

[29] S. Sekar, Y. Guang, S. Verma, and T. Shanbhag. “A high-performance network intrusion detection system”, *Proceedings of the 6th ACM conference on Computer and Communications Security (CCS)* 1999; 8-17.

[30] R. Sharpe. “Ethereal’s User Guide”, 2001 <http://www.ethereal.com/docs/user-guide/>.

[31] C Smith, P Grundi, Subterrain Siphon Project. “Know Your Enemy: Passive Fingerprinting” 2001 <http://project.honeynet.org>.

Appendix A: RUM User Manual

RUM User's Manual

Release 1.0 for RUM v1.1

Brian Goff

Table of Contents

---

1. Introduction

What is RUM?

Resource Usage Monitor (RUM) is a distributed network analyzer tool producing statistics on traffic entering and leaving a network. Collected data is stored by subnet, allowing an external view of any particular subnet or collection of subnets. An external view means that only communications or communication attempts between an internal host and a host outside of the network are analyzed.

Features implemented in this release of RUM include:

- Capturing of IPv4 packets from high-bandwidth networks
- Three levels of data granularity (subnet, VLAN, network)
- Two levels of user-specific access to data
- Customizable web interface including the ability for each user to save their own graphs and reports
- Graphs displaying bits/sec, packets, and flows for each level of data classification
- Reports describing inbound and outbound traffic activity for internal hosts, external hosts, applications, IP multicast addresses, web servers, subnets, VLANs, and networks
- Queries detailing the actual captured packets
- Triggers activated by packet characteristics or traffic exceeding user-specified threshold levels
- Log reports of trigger activity
- Passive fingerprinting for remote OS identification
- Graphs displaying the overall performance of the RUM processing engine

Data is collected from a network probe monitoring the network at the gateway between the internal network and external world. The web server hosting the data to the web is not necessarily the same machine collecting the data.

Packet data is collected using the libpcap library. The data from RUM is extracted form a few header fields of each packet. No actual user data is ever collected, nor stored. The average number of bits/sec, packets/sec, and flows/sec of each subnet are stored to provide persistence data in the graphs. Trigger-based logs are used to provide persistence data at the host and application levels.

The data presented on the web is not real time, but close. Due to the amount of processor time required for the analysis of the data, data is first collected for a period of time and then analyzed. The length of this sample time and the interval between each sample can be found on the main page of the web interface. After being analyzed, the data is transferred to a location where it can be accessed from the web.

There are currently three types of analysis implemented, load, packet, and flow. These analyses are helpful in various type of security and quality of service evaluations as described in section four.

> Load Analysis
> The *load* is the actual number of bits that are transmitted during the sampling period. Bits are summed throughout the sampling period and then divided by the sample duration to obtain the rate (*bits/sec*).
> Packet Analysis
> This is simply the total number of *packets* transmitted during the sampling period. *Packets* are summed throughout the sampling period and then divided by the sample duration to obtain the rate (*packets/sec*).
> Flow Analysis
> A *flow* is a unique source IP, source port, destination IP, destination port, and IP protocol combination. *Flows* are tallied throughout the sampling period and then divided by the sample duration to obtain the rate (*flows/sec*).

Development and maintenance of RUM

RUM was developed by Brian Goff of North Carolina State University. Current development and maintenance is still handled by Brian.

Several other people have contributed to the development of RUM.

Platforms RUM runs on

RUM currently only runs on Intel platforms running UNIX. Testing has only been carried out on Red Hat Linux 7.x though other flavors of Linux should work with minor modifications. The location where RUM extracts NIC statistics may be OS specific. For Red Hat Linux, this information can be found in /proc/net/dev. This filename and location may need to be changed if using another OS.

RUM requires Perl, RRDtool, and a web server in order to work properly.

Where to get RUM

You can download the latest copy of RUM through the development site http://traffic3.cc.ncsu.edu. Currently this is a restricted site. Email Brian Goff if you are interested in obtaining access to the RUM tool.

Contact information

Problems related to access and general RUM usage should be directed towards you RUM administrator. See the contact page for more information.

For problems related to the development of RUM or installation issues, email Brian Goff.

Send any feedback about this document to Brian Goff as well.

---

2. Installing and Configuring RUM

Dependencies

The majority of the code required to run RUM such as its modified libpcap library are included with the distribution. Other applications required by RUM include Perl, RRDtool and a web server.

Perl
RUM was written using Perl 5.6, however RUM doesn't make use of any 5.6-specific features so version 5.005 or even 5.004 should work fine. The following Perl modules are required (or which, the majority are included with a normal Perl distribution.)

- CGI
- Data
- Fcntl
- File
- Getopt
- POSIX
- Storable

You can download and install needed modules from http://www.perl.com/CPAN/.

RRDtool

RRDtool is required for the graphing functions of RUM. It can be downloaded from http://www.rrdtool.com. When installing RRDtool, be sure to install the Perl modules in the default location so that RUM is able to find them. This can be down with

```
make site-perl-install
```

Web server

Some type of web server is required to display the RUM graphs and reports. Apache 1.3.20 was used on the test machines. User authentication needs to be handled by the web server. RUM uses the REMOTE_USER variable of the web server to determine which datasets a user has access to. No user name means no data sets. Also, the web server needs to be set up to associate files ending in the .pl extension as CGI scripts.

Installing

There are two steps to the RUM installation process. The first is to build the source files that are written in C. The following commands will unpackage and build the required executables.

```
% gzip -cd rum-1.1.1.tgz | tar xvf -
% cd rum-1.1.1
% cd src
% make
```

The second step creates the RUM directory tree and installs all of the executables and scripts. In the rum-1.1.1 directory, execute the following command:

```
% perl INSTALL.PL
```

This will install RUM in /usr/local/rum-1.1.1. If you prefer to install RUM in another directory use:

```
% perl INSTALL.PL --prefix=/some/where/else
```

RUM is now installed. You need to configure the system before you are able to use it. You also need to configure your web server to accommodate the RUM web interface. The suggested way is to alias the RUM www directory.

Global configuration

Basic RUM configuration is handled through the rum.conf file located in RUM's conf/ directory. Data in this file is of the format

```
variable value
```

The table below describes all of the variables from this file. All of these variables must be present, in the same case, in order for RUM to function properly. More verbose descriptions can be found in the configuration file itself.

Variable	Sample Value	Description
RUMHost	traffic	Descriptive label of probe machine to show up in web interface
RUMAdmin	admin@traffic	Email of RUM administrator to show up in web interface
RUMNetwork	192.168.1.0 255.255.255.0	Network address and mask that make up internal network
CaptureInterface	eth0	Network interface through which to sniff network
CaptureDuration	60	Number of seconds to sniff the network on each run
CaptureInterval	300	Number of seconds between runs
CaptureFilter	"ip"	BPF command to use when capturing traffic
ThreadPoolSize	2	Number of parallel threads to use when processing data
RequestQueueMax	100	Maximum number of subnets that can wait for worker thread at any one time
SavedEntriesMax	10	Maximum number of saved reports and graphs per type of form.

Configuring subnets

Subnet configuration is handled through the subnet.conf file located in RUM's conf/ directory. Data in this file is of the format

```
subnet mask VLAN network
```

The subnet and mask are both in dotted decimal format. Where as a subnet is a group of IP addresses, a VLAN is a group of subnets. A VLAN is basically a description of the subnet. The network is a third level grouping, or a collection of VLANs. There must be both the VLAN and the network values for each entry in the configuration file.

After editing this file, run the initialization file in RUM's bin/ directory

```
% perl RUMinit.pl
```

This file will initialize the graph datasets. This file can be run as often as needed after you have started using RUM. It must be run after any changes are made to either the rum.conf or subnets.conf files. Changing the list of subnets, VLANs, or networks will create new directories and alter which subnets are being used. Old data directories must be removed

manually. Therefore if a subnet is used again for a second time, the old data will still show up in the graphs if it was never removed. In order to reset an RRD file, simply erase it and rerun the initialization file.

Configuring OS fingerprints

In order to attempt to passively identify the OS of a remote host, the known signatures for each operation system need to be entered in the fingerprints.conf file located in the conf/ directory. Data in this file is of the format

```
name version platform TTL (TCP window size) (DF bit) TOS
```

Several signatures are already provided complements of the Honeynet Project.

Configuring users

After running the initialization script, a users.tmpl file will be created in RUM's conf/ directory. This is just a template file and does not have to be used. Using the template file or not, you need to create the configuration file users.conf in RUM's conf/ directory. Format of this file is

```
[full]
net::network = user1, user2, user3
vlan::VLAN = user1, user2, user3
sub::subnet/mask = user1, user2, user3

[limited]
net::network = user1, user2, user3
vlan::VLAN = user1, user2, user3
sub::subnet/mask = user1, user2, user3
```

Network, VLAN, and subnet/mask are the names of predefined categories set up in the subnet configuration file. The usernames need to be separated by commas and must be recognized by the web server's user authentication. Users listed under the 'full' heading will receive full access to those datasets. Users listed under the 'limited' heading will receive partial access to those datasets. Partial access means that the data will still show up but all external addresses will be blocked from viewing. In processing the access list, 'full' access is processed before 'limited'. This means that if a user has both 'full' and 'limited' access to the same dataset, their 'limited' access will overwrite their 'full' access. Also, the processing of the different groupings is in the order or overall, networks, VLANs, and finally subnets. After editing the user configuration file, run the RUM optimization file found in RUM's bin/ directory.

```
% perl RUMoptimize.pl
```

This file will take the subnets, fingerprints, and user configuration files and optimize them for use with the web interface. The web interface will not work properly until this file is run at least once. This file should be run after every time any of these three

configuration files are updated. This file can be run as often as needed after you have started using RUM.

Internet services

RUM uses the inet services file located at /etc/services to associate port numbers with specific applications. Changes to this file will proliferate into the RUM reports.

Scheduling RUM to run automatically

The main execution script for RUM (RUMengine.pl) can be found in the RUM bin directory. In order to run RUM once, run this script.

```
% perl RUMengine.pl
```

In order to run RUM continuously, you must add this command to root's crontab. Be sure to use the same increments between runs as you had indicated in the global configuration file.

---

3. Using RUM

Web interface layout

The web interface for RUM makes use of JavaScript and DHTML as well as cascading style sheets (CSS). Any browser that supports all of these should work fine, however the appearance of the web interface may vary between browsers. Make sure that JavaScript is enabled in order for all of the features to work.

The layout of the RUM interface can be divided into four parts: the title bar, status bar, menu, and main page.

Title Bar
The title bar is located across the top of the page with a series of links on the right hand side. 'Home' will return you to the main RUM page listing all of the user-specific details. The 'Save' link is used to save graphs, reports, packet searches, and log reports for later viewing. 'New Window' will open whatever is currently being viewed in the main window into its own browser window outside of the frames. 'Contact' lists email information for issues encountered while using RUM. Finally, 'Help' will direct you to this user manual.

Status Bar
The status bar is located across the bottom of the page. In order from left to right, this contains the time of the last sample, the duration of that sample in parentheses, and then data totals from the last sample. These totals include the estimated sampling coverage, bits/sec, packets, and flows.

Menu
The menu is located on the left hand side of the page. Links to the customizable features of RUM are located in the top half of the menu. Links to the other RUM screens are found in the bottom half. Since some screen resolutions may not be big enough to fit all of the menu items in at once, arrows have been included at the top and bottom of the menu window to scroll the window in either direction.
Main Page
The main page is the most predominant window in the RUM interface. This is where all configuration options and data are displayed.

Creating a new graph

In order to create a new graph, select first the type of metric to analyze. This can either be bytes, packets, or flows. Next, select the type of graph. A 'Totals' graph displays the total inbound and outbound values as a blue background with the individual inbound and outbound values as lines superimposed on the background. This graph is good for getting a general description of network traffic and comparing inbound to outbound. A 'Network' graph is good for comparing individual datasets to one another. All selected datasets will be stacked on top of one another with outbound traffic shown on the positive axis, and inbound traffic shown on the negative axis. The 'IP Protocols' graph compares the total TCP, UDP, and other IP traffic for each dataset selected. Again, outbound traffic is on the positive axis and inbound traffic is on the negative axis. The last graph parameter to select is the period. Graphs can be displayed for the previous 24 hours, previous 7 days, previous 30 days, or previous year.

Finally, select with dataset(s) to examine. Only datasets for which you have access to will be displayed. You may only select from one classification of datasets (network, VLAN, subnet) at a time. However, multiple datasets within each classification may be selected.

Creating a new report

In order to create a report, the type of report analysis needs to be selected from the list. Applications describe the port activity of the last sample. External hosts describe just the addresses that are not part of the local network. Internal hosts describe just the local network addresses. Multicast examines the IP multicast addresses that traffic is being sent to. Network, subnets, and VLANs examine the activity at each of the respective levels of granularity. Finally, web servers describe which hosts are serving HTTP requests.

The direction of traffic needs to be selected from either inbound or outbound. Unless the "Inbound Multicast" dataset is selected, there will never be inbound multicast traffic to display.

Traffic parameters allow you to further limit the results of the reports. Minimum and maximum values can be selected for bits/sec, packets/sample, and flows/sample. The minimum value must be less than the maximum value. A value of zero for either minimum or maximum is the same as 'no limit' and will be ignored.

Select the order in which you would like the results displayed. Results can be ordered by bytes, packets, or flows, from the greatest to least.

For faster downloads, the number of entries displayed to the screen can

Finally, select with dataset(s) to examine. Only datasets for which you have access to will be displayed. You may only select from one classification of datasets (network, VLAN, subnet) at a time. However, multiple datasets within each classification may be selected.

Creating a new packet search

Packet searches display the actual packet data captured during each sampling period. This search can be limited by a source IP address, destination IP address, IP protocol, source port, and destination port. IP addresses must be in dotted decimal format. IP protocol must be entered in numerical format. For example, the number 6 represents TCP. Port numbers must fall in the legal port range and are only allowed if the IP protocol is TCP (6) or UDP (17).

Source and destination IP addresses or port numbers can be combined by either the logical operator AND or OR. With AND, only packets with both the inputted source and destination values will be matched. OR allows packets with only one of the values to be matched as well. To search for traffic to or from a particular host, use the machine's IP address for both the source and destination address, and select a logical operator of OR.

Finally, select with dataset(s) to examine. Only datasets for which you have access to will be displayed. You may only select from one classification of datasets (network, VLAN, subnet) at a time. However, multiple datasets within each classification may be selected.

Creating a new log report

Log reports display the data written to the log files by previously installed triggers. If a trigger has been deleted, the log file will no longer exist and cannot be examined. To create a log report, first select from the list of log files currently on the system.

The log file can be examined in one of several ways. The log file can be examined in chronological order. This will display the log file pretty much as is. For triggers based on packet searches, this could be a long list. The log file can be examined in byte, packet, or flow order from greatest to least. Flow order is not available for packet search-based triggers. Report-based log files can also be examined by high-water mark. This will display the greatest value for every entry in the list. High-water analysis can also be byte, packet, or flow-based. One last way of examining report-based log files is through frequency. Entries who appear on the list the most often will be at the top of the list.

For faster downloads, the number of entries displayed to the screen can be limited.

Saving graphs, reports, and searches

You may save common settings for graphs, reports, packet searches, and log reports to be able to quickly and easily view them at a later time. In order to save one of these, it must first be created using the 'New' link under the appropriate heading. After the network is displayed in RUM's main window, select 'Save' from the title bar. A window will pop up where you can enter an appropriate title for the graph or report. If you use the same title as an already existing graph or report, the older one will simply be overwritten.

The RUM administrator has the ability to limit the number of saved graphs and reports for security purposes by setting the value in the RUM global configuration file.

Retrieving graphs, reports, and searches

Saved graphs, reports, and searches can be retrieved through following the 'Retrieve' link beneath the appropriate heading. A list of names corresponding to all of the saved entries will be displaying. Click on the name of the graph or report to display the current statistics.

To edit a particular entry, select the checkbox next to the entry you wish to make changes to and select edit from the list below. A form much like the one used to create the entry will be displayed with the current settings already displayed. After your changes are made, select 'Update' from the bottom of the form.

Deleting graphs, reports, and searches

Unwanted graphs, reports, and searches can be deleted from the system by following the 'Delete' link beneath the appropriate heading. A list of names corresponding to all of the saved entries will be displayed. You will be prompted one last time before the entry is deleted. Once deleted, there is no way to get back a saved set of parameters.

Reports and packet searches that are part of a trigger cannot be deleted until the trigger has been deleted first.

Installing a new trigger

Triggers are created using previously created and saved network reports or packet searches. Once the trigger is created, the report or packet search may not be deleted unless the trigger has been deleted first.

To create and install a new trigger, follow the 'New' link beneath the trigger heading in the RUM menu. First assign a name to the trigger you are creating. Select from the list of saved reports and packet searches, the type of auditing you will like to match for on each run of the RUM processing engine.

There are three alert options, email, notice, or none. Email will send a short email listing the trigger name and time that the alert occurred. Notice will send the same information

to a provided host and port number. None will simply log the information. In the alert information box, enter either the email address, or host/port combination depending on the type of alert you selected. Host/port combinations should be written in dotted decimal notation with the IP address and port number separated by a semicolon (i.e. 192.168.1.45:2005).

Finally select how far back log reports should record data. The maximum length is 30 days. After 30 days, the information in the log will be automatically deleted. After selecting 'Install' from the bottom of the form, a message indicating whether or not the trigger was properly installed will be displayed.

Retrieving triggers

In order to change the settings for a trigger select 'Retrieve' under the triggers heading in the RUM menu. Select the checkbox next to the trigger you wish to edit. Any changes will take effect immediately after selecting 'Update' from the form.

Deleting triggers

Unwanted triggers can be deleted from the system by following the 'Delete' link beneath the triggers heading. A list of names corresponding to all of the saved entries will be displayed. You will be prompted one last time before the entry is deleted. Once deleted, there is no way to get back a saved set of parameters.

Deleting a trigger will also delete the log file associated with it.

Host lookup

RUM uses passive OS fingerprinting to attempt to identify the source of each packet. With full access, you are able to view any fingerprint matches for all internal and external addresses with current activity in any of the datasets for which you have access. With limited access, you may only view fingerprint matches for internal IP addresses. Since passive OS fingerprinting is not 100 percent accurate, other statistics are provided to help you evaluate the accuracy of the fingerprint results. The total number of packets sent by the host being examined is provided along with how many of those packets match to each fingerprint. Multiple fingerprints may match the same host.

Access list

There are two different levels of access for RUM users, full and limited. Full access enables the user to view all available data in RUM. Limited access provides the user with all available statistics, however any external addresses are blocked from being viewed.

The RUM administrator determines the access levels for different users. Users can view their access level to each subnet through the access list link located on the menu bar. If a user does not have any access to a particular dataset, than that dataset will not show up in their list.

Customization

In order to make viewing graph and report data easier, each user is able to assign a specific label and graph color to each dataset. Customization setting can be reached via the Customize link near the bottom of the menu.

Custom labels will show in the legends of each graph as well as the reports. When a dataset does not have a custom label, the default name is used.

Graph colors are assigned from a default list of colors. Depending on the order of datasets being examined, different colors will show up each time a graph is viewed. Custom colors allow the same color to show up every single time, making it easier to recognize a particular dataset.

Performance

Since RUM can be used to monitor high-bandwidth networks, the actual performance of the processing engine is important. Two types of performance graphs are available.

Estimated Sampling Percentage
This graph shows the sampling percentage as a function of time. Estimated sampling percentage describes the amount of packets actually captured by the RUM probe compared to the physical number of packets reported by the NIC. This percentage value is used to adjust the graph and report data of each sample.
Processing Time
Processing time shows the elapsed time of each section of the RUM processing time as a function of time. The times of each code section are stacked on top of each other in order to show the elapsed time of the overall analysis. Time are included for the capturing of packets, sorting of packets by subnet, analysis of data in each subnet, analysis of non-subnet data, the merging of subnet totals for each graph, and the processing of event triggers.

Each graph can be displayed with statistics from the previous 24 hours, 7 days, 30 days, or 1 year.

Related links

RUM is just one of many useful network monitoring tools. Links to other useful monitoring sites can be found on this page.

---

4. Network Analysis with RUM

Security

Several features of the Resource Usage Monitor contribute to its usefulness as a network security tool. RUM uses anomaly detection in order to discover possible security threats. The graphs alert users of problems with different regions of the network. Further examination of the reports indicates the hosts responsible for the irregularities seen in the graphs. Since general reports only contain data from the most recent sample, log reports provide this same information when investigating previous network data. Once general network behavior has been determined, thresholds can be set, causing the RUM system to automatically alert the user when a possible security threat is occurring. Using these features, several different security concerns can be successfully detected with the RUM application. Also, as the system collects statistics on both inbound and outbound traffic, it cannot only detect security threats against the internal network, but also security threats originating from within the internal network.

Quality of Service

The graphs and reports from the Resource Usage Monitor provide useful data for network assessment. Quality of service (QoS) involves the measurement of network characteristics, improving upon them, and possibly even guaranteeing a particular available bandwidth for certain types of traffic. Since RUM does not interact with the network, the reaction to network traffic statistics must come from other systems. RUM however, has many features capable of providing network statistics that act as a basis for making network management decisions. This allows the Resource Usage Monitor to play a vital role in quality of service evaluation. RUM is even capable of offloading its statistics to another system to allow pro-active management of the network.

---

5. RUM Error Messages

No access

If there is no entry for your username in RUM's user configuration file, you will receive this type of error. This could be caused by a typo in the configuration file. If you feel like you should have access, contact your RUM administrator.

Invalid datasets

This error occurs when the datasets for which you are listed as having access to do not actually exist in RUM's subnet configuration file. This could be caused by a typo in the configuration file or an out-of-date user file. If changes are made to the subnet configuration file, changes need to also be made to the dataset listings in the user configuration file. If you receive this error, contact your RUM administrator.

Cannot open configuration file

This error occurs when RUM is unable to locate the RUM configuration files. This is usually due to the web server not having proper access to the file. If you receive this error, contact your RUM administrator.

Missing parameter

Whenever submitting a form, either a graph, report, search, log, etc., if a necessary piece of data is missing, you will receive this type of error. This type of error message also contains the exact parameter that is missing. It lists the parameter by the name that RUM uses to refer to it, so it may not match the same description in the form, however it will be close. RUM will only list one missing parameter at a time, so you may receive this type of error one after the other until you provide all of the correct information.

Not a valid parameter

For each submitted form, either a graph, report, search, log, etc., each parameter is checked to make sure it is legal. For checkbox and radio buttons, each parameter must correspond directly to what is available in the form. IP addresses must be entered in dotted decimal format. All protocols must be entered using their numerical value.

No dataset selected

This error occurs when the user has not selected from either the group of network, VLAN, or subnet datasets. Select which dataset to examine and resubmit the form.

Not a valid combination of datasets

This error occurs when datasets from different categories are selected at the same time. For each form, the dataset(s) must be chosen from just one category, either network, VLAN, or subnet. There are further limitations on which network datasets may be selected at the same time. "Overall" dataset must be selected independently of any other type of dataset. The "Illegal", "Classification Pending", and "Inbound Multicast" cannot be selected with the "No Subnet" dataset. Edit your dataset choices and resubmit the form.

Not a valid combination of limits

This error occurs when limits on the report form are not logical combinations. Usually this means that the minimum value is greater than the maximum value.

Report not available

Some reports, such as the "Network", "VLAN", and "Subnet" report are only available when selecting datasets from that group or a group above in the

classification hierarchy. For example, the "VLAN" report is only available for VLAN and network datasets. Subnet datasets do not work with "VLAN" reports.

RRD error

This error occurs when there is a problem with the graph databases. Contact your RUM administrator if you receive this error.

Release 1.0 for RUM v1.1 - 27 Feb 2002