Abstract

Zheng, Erkang. Interactive Assistance for Anomaly-Based Intrusion Detection. (Under the direction of Dr. Mladen A. Vouk)

Network and information security is of increasing concern as intruders utilize more advanced technologies, and attacks are occurring much more frequently. A simple intrusion can cause an enterprise financial disaster, a threat to national safety, or loss of human life. Network-based and computer-based intrusion detection systems (IDS's) started appearing some twenty years ago. Now, there are various synchronous and asynchronous tools for external and internal network and host intrusion detection using models ranging from signature scanning and pattern matching, to statistical anomaly detection. Although modern IDS systems are much more advanced, they still have many limitations, shortcomings, and open issues. This includes a) inability of some to handle high speed network traffic, b) poor ability to detect new or first-time intrusions, c) high false alarm rate, d) deception – such systems may have problems detecting "below noise" level intrusions, e) overload – IDS, like any other system, may be vulnerable to the same attacks it is trying to detect, including Denial of Service (DoS) attacks, f) customization and end-user integration - unless the system is open-source, customization and integrations options may be limited – including how to properly augment and integrate human anomaly detection experiences and tool detection capabilities, g) automation of the processes, and h) privacy issues.

This work is concerned with exploration of items b) and f) above, specifically on development of a prototype module for assisting human intrusion detection personnel in recognition of new threats. The work builds on system called Resource Usage Monitor

(RUM) developed at NC State by developing its IDS assistance module. The intrusion detection module utilizes RUM as its statistical packet capturing and basic analysis engine, utilizes it to cross check its problem detection abilities, and adds to its resource risk assessment ability a facility for intrusion risk assessment using a suite of behavior description measures and intrusion threshold indicators.

The RUM IDS module is an exploratory engine designed to set the tableaux for a more complete investigation of a) pro-active anomaly detection, and b) smoother integration of human intrusion detection experiences and a real-time IDS tool. The approach involves analysis of end-host databases for anomalies based on a suite of statistical change metrics. There are two principal "views" of a host and two groups of associated metrics. How it behaves with respect to a set of peers, i.e., network-relative behavior, and how it behaves with respect to itself, i.e., how its behavior changes from sample to sample. According to the behavior during the analyses, each host accumulates an anomaly index value, where a higher number represents a higher potential for misbehavior. Currently, the prototype anomaly index is based on a linear additive model. This may change as the research continues. The idea is that this index, once properly tuned, would correlate better with intuitive problem detection processes of network administrators, than does plain display of, for example, "high talkers". The primary goal of this work is to develop and test a RUM IDS module and its initial set of metrics. , While full investigation of the assistant index idea is beyond the scope of this project, formative results indicate that a subset of the metrics under investigation does indeed provide better high-speed problem detection, when combined with a human analyst, than do some other readily available tools.

INTERACTIVE ASSISTANCE FOR ANOMALY-BASED INTRUSION DETECTION

By

ERKANG ZHENG

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 SCIENCE

Raleigh, North Carolina
2004

APPROVED BY:

Dr. Ana I. Antón	Dr. Peng Ning

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

Biography

Erkang Zheng was born in Beijing, China. He moved to the United States in 1997. At the age of 19, he earned a Bachelor of Science degree in Computer Science from the North Carolina State University. He continued with his graduate study in Computer Science at NC State University with focus on Network Security. In addition to the degrees, he also holds a number of industry certifications, including Cisco certified CCNA, CCNP, CCDA, CCDP, CompTIA certified A+, Network+, Linux+, and Microsoft certified MCP, MCSA, MCSE, MCDBA.

Acknowledgements

I would like to express my deepest gratitude to my advisor, Dr. Mladen Vouk, for his invaluable advice during the course of this research study. I would also like to thank the members of my thesis exam committee, Dr. Annie Antón and Dr. Peng Ning for their reading of the manuscript and their valuable comments.

I am grateful to the Department of Computer Science at the North Carolina State University at Raleigh for its excellent computer facilities.

Finally, I would like to convey my sincerest regards to my wife and my parents for their love and support.

This work was supported in part by the IBM Shared University Research program, and by Cisco Systems.

Table of Contents

List of Tables

List of Figures

List of Equations

Chapter 1 – Introduction

1.1 Motivation and Goals

The emergence of Internet technologies, and especially, the Internet, has brought huge change to the digital world of computing and has altered the way digital information is presented [1]. With the advances in network-based computing technologies, computer crime has become more common than ever before and network related security is playing an increasingly important role [2]. Concerns about network-based intrusions started more explicitly about twenty years ago after Robert Morris released the first large-scale Internet attack via his "worm" [5][51]. Over the last twenty years a tremendous amount of money and resources have been spent on the study and implementation of method and tools for network-based intrusion detection – the so called intrusion detection systems (IDS's). Unfortunately, the returns appear to be disproportionate to the investment. While IDS's of today are quite sophisticated, there are many problems with them as well. Many IDS do not work well in a real world environment, although they may achieve good laboratory performance [7][8].

Some of the limitations and shortcomings are

a) incapability of handling high speed network traffic,

b) poor ability to detect new or as yet unknown intrusions,

c) high false alarm rate,

d) easily overloaded (fail-open and resource starvation),

e) vulnerability to Denial of Service (DoS) attacks,

f) ability to handle fragmented and malformed packets,

g) few or no customizable options,

h) inadequate integration with human analysts, and

i) privacy issues.

In addition, many high-performance IDS available today tend to be proprietary and commercial in nature. That makes it difficult to address items g) and h) above beyond what the manufacturer offers. It also makes it difficult to improve on implemented algorithms regarding items b) and c) above.

There are three principal goals for current work: 1) to develop an initial IDS module for the NC State open source Resource Usage Monitoring (RUM) tool [31], 2) investigate new ways of detecting anomalies related to network-based intrusion attempts that address some of the open issues mentioned before, and 3) start developing a suite of metrics that may be more conducive for seamless integration of human intrusion detection experiences and automated tools than what is already available.

1.2 Outline of the Thesis

Chapter 2 provides an overview of the prior work related to dynamic IDS. Chapter 3 explains the rational for and approach taken in the design of the metrics, algorithms, and policies developed as part of this thesis work. It also introduces glossary of terms used. Chapter 4 provides a detailed description of the initial implementation of the RUM IDS module. Chapter 5 reports on the formative evaluation of the approach in

terms of effectiveness, performance, and limitations using a series of examples and case studies[1]. Chapter 6 presents conclusions and discusses possible future work. References and end-user manual are included in the Appendices.

In order to protect privacy of explicit end-users source/destination addresses of fully identifiable individual flows that do not originate at or from public servers may be scrambled.

[1] Individual IP numbers may have been modified to protect privacy of source and destination nodes.

Chapter 2 – Background

2.1 Technology

2.1.1 Intrusions

A computer network is made up of computing devices connected to each other through a digital network. To guarantee normal operation of the network, among other things the security of each host on the network must be ensured [3]. Activities that create potential risk fall into two categories: destination related and source related. Former are direct attacks against one or more hosts on the network (i.e., against a network destination of concern). Latter are network-visible real or potential misbehaviors of one or more hosts (or sources of interest) on the network. Of course, not all destination attacks may be damaging since the destination node may be resilient (including via fire-wall) or non-existent. Similarly, not all source-anomalies result in a destination attack since attack may be blocked at some point along its path to a destination, but they may be indicative of a corruption of a host. Depending on who is looking at the information one or the other of the activities may be of greater import. For example, an organization that has high-confidence in its end-host security may be more concerned in detecting and blocking (perhaps centrally) malicious behavior of its registered hosts (IP addresses), than blocking incoming attacks. However, ideally, both activities should be prevented, avoided and/or eliminated. A network administrator needs to be informed when intrusion category situations occurs. That is where intrusion detection systems come into play.

Attacks can be **passive** or **active** [4]. Passive attacks usually pose a less immediate threat because an "intruder" eavesdrops to collect information on the wire, but does not transmit information to create damage. Such events may be difficult to detect. The focus of an intrusion detection system (as considered in this thesis) is on **active attacks**, i.e., situations where an intruder or an infected host collects information on the wire by active scanning of the target hosts, by modifying "sniffed" information and re-transmitting it, or by sending attack packets to create immediate damage to the hosts on the network.

For example, the best known, and frequently seen, active attack is called denial-of-service (DOS) [41]. In a DOS attack, the attacker sends an extremely large number of request to the target server, overflows the server's ability to respond to other legitimate users [41] and thus it effectively reduces the quality of the service the server is able to offer. Besides flooding the servers and/or service ports, DOS attack also involves jamming networks and misconfiguring routers. When an attacker installs an agent or daemon on numerous hosts and controls them to launch a group attack, it is known as Distributed Denial of Service, or DDOS attack. Methods of DOS/DDOS attack include FTP bounce, Ping flooding, SYN flooding, IP fragmentation, and many others [52]. Another well-known network-visible attack is a port scan. In this case an attacker tries to connect to some or all ports on the target host, in order to find a security hole [35]. Detection of DOS/DDOS attacks and port/host scans is the primary goal of most intrusion detection systems available today. Yet another is break-in attempts via password protected services. In such attacks, one is known as a dictionary attack, attackers often look for weak or non-existent

passwords. Other types of attacks include deception activities of various types such as social engineering and impersonation, use of known exploits, transitive trust attacks, data driven attacks, and infrastructure attacks [53][54]. There are too many types, methods and mechanisms of attacks to provide a comprehensive description. New attack techniques and exploits are constantly being developed and discovered.

The other type of threat to a network is a network-visible misbehavior of particular host(s) of interest that has the possibility of leading to one or more of the previously mentioned attack types. . The misbehavior includes a large variety of activities, from running illegal or dangerous services, to being visibly infected with viruses, worms or Trojan horses. For example, a Trojan horse is "instructions hidden inside an apparently useful program that do bad things"; a virus is "a set of (often malicious) instructions that, when executed, inserts copies of itself into other programs"; and a worm is "a program that replicates itself by installing copies of itself on other machines across a network" [4]. Of course, all three forms may also be used to collect sensitive information from the infected hosts. Such activities may not result in attacks on other machines, but may still result in anomalous networking activities (e.g., transmissions of data exceeding "normal" rates (or operational profile), transmissions at odd times of the day, and so on). Thus, the results of these and similar events can be serious, often times granting access to an intruder to other machines (hosts), and perhaps causing irreversible damages such as permanent loss of data [39].

2.1.2 Intrusion Detection

Intrusion detection is, exactly as its name suggests, a technique used to detect possible intrusions. More technically, intrusion detection is an algorithm (and associated security model) that is aimed at detection of attacks and/or infected hosts. Intrusion detection systems (IDS), therefore, are the implementations of such algorithms and models [6].

By definition, intrusion detection is the art and science of sensing when a system or network (and its hosts) is being used inappropriately or without authorization. An intrusion-detection system (IDS) monitors system and network resources and activities and, using information gathered from these sources, notifies the authorities when it identifies a possible intrusion [42].

We can think of intrusion detection systems installed on a computer, or another network device, as to the alarm systems installed in a building or a house. They provide the same functionality, which is to alert the owners (administrators) of the property when something bad is happening or is about to happen.

2.1.2.1 Host IDS

Intrusion detection systems come in a variety of forms and based on a variety of models, from as simple as keystroke monitoring, to as complex as stateful pattern matching.

Host-based IDS (or HIDS) were one of the first type of IDS developed and implemented [5][10]. These systems reside at a particular computer or server, scanning all traffic that passes in and out of the system. Some HIDS also examine system logs in order to find security holes and vulnerabilities. Tripwire, for example, is a tool that checks to see what has changed on your system. The program monitors key attributes of files that should not change, including binary signature, size, expected change of size, etc. Tripwire has both an open-source and a commercial version [55][56].

Intrusion Detection Agent System (IDA) [11] – This is another example of a HIDS that analyzes system log files is IDA (Intrusion Detection Agent System), developed by the Information-technology Promotion Agency (IPA). IDA primarily detects intrusions that aim at root privileges in UNIX. It is able to trace intruders using mobile agents in a local area network (LAN). One such enhancement is the intrusion detection method based on discriminant analysis [16]. It allows IDA to detect local attacks as well as remote attacks based on the number of system calls during a user's network activity on a host machine.

Kernel-based IDS (KIDS) [12] – Kernel-based IDS is a representative of another form of HIDS. The best known is the Linux Intrusion Detection System (LIDS). LIDS is a Linux kernel patch that allows users to use when needed, but not abuse, the all-powerful nature of root. Thus it attempts to prevent the awful outcomes when the root account is taken in an attack. Its main features include kernel sealing, port scan detecting, and access restriction control.

2.1.2.2 Network IDS

On the other hand, as its name suggests, **network-based** IDS (NIDS) deals with packets transmitted over the network [5][9][10]. Traditionally, with hubs connecting all the computers on a network, NIDS uses a network interface card (NIC) in promiscuous mode to catch all the packets transmitted over the network space visible to the NIC. With the advent of switching technology, a NIC in promiscuous mode cannot do the job anymore. Many NIDS nowadays utilize switch "port spanning" coupled with tools referred to as "packet-sniffers" to capture packets flowing on the wire. Issues that NIDS are currently facing include data encryption and high-speed data transmissions. Normally, NIDS are unable to examine packets that have been encrypted, and most of them perform poorly on data flows exceeding several hundred Mbps.

Whereas HIDS are effective at detecting insider abuses, NIDS are best at detecting unauthorized outsider access, bandwidth theft/denial of service, overload, and similar problems.

Hybrid Intrusion Detection [5] – The idea behind Hybrid Intrusion Detection combines the use of HIDS and NIDS. In other words, the network consists of HIDS installed on each computer, and a NIDS installed on the network using "sniffer" technology. This approach takes advantage of both HIDS and NIDS intrusion detection capabilities. Unfortunately some of the disadvantages of both systems also carry over.

Network-Node Intrusion Detection (NNID) [5] – The idea of NNID is based on that of NIDS and HIDS. It performs a task similar to NIDS. But, instead of using a "packet sniffer" on the network to collect data for a single NIDS to analyze, NNID software system is installed on each individual node on the network, and monitoring packets sent from and to the node it resides. This is similar to HIDS. However, the deployment overcomes the shortcomings in NIDS. First, switched network is not a concern anymore since each NNID is acting as a sniffer for the local node. Also, there is no need to deploy "taps" on the wires or to configure switches with a managed spanning port. Second, NNID has access to all the data including those protected by VNP encryption. Finally, it overcomes the limitation of network bandwidth and speed when it comes to monitored traffic. The throughput concern for each node is significantly lower than that of the entire network. However, NNID still cannot handle pass-through traffic that may not be destined for any of the monitored nodes, and it cannot handle encrypted traffic not intended for its nodes.

2.1.2.3 Misuse vs. Anomaly Detection

By their effect, most intrusions can be divided into six main types—attempted break-ins, masquerade attacks, penetration of the security control system, leakage, denial of service, and malicious use [13]. In turn, these fall into two major categories, misuse and anomaly intrusions, and hence, we have intrusion detection types specifically addressing them.

Misuse detection (deductive detection) attempts to map each known attack to a specific pattern or signature. Then, based on that signature an attempt is made to detect the same attack if it happens again. Thus, misuse detection is also known as signature detection or rule-based intrusion detection [14], or deductive detection. The idea is similar to that used in virus scanning systems. For example, a particular attack may be found to contain a destination port of 5112 in its packet header(s). A deductive system then looks for a known attack (and pattern) related to port 5112. If one is found, a confirmatory test may be run if one is available, and then an alert-state is declared. . However, if the pattern is changed in the next attack (or misuse), say to port of 5113, the system will not be able to identify the problem. This is, for example, the strategy that peer-to-peer systems have adopted after administrators have begun blocking "standard" peer-to-peer ports. Since there is usually a large number of variations to an existing attacks, the probability of catching, using deductive methods, a human intruder who directs an attack (as opposed to an automated attack based on a particular pattern) is relatively low. However, deductive methods work well with known patterns (such as identified viruses).

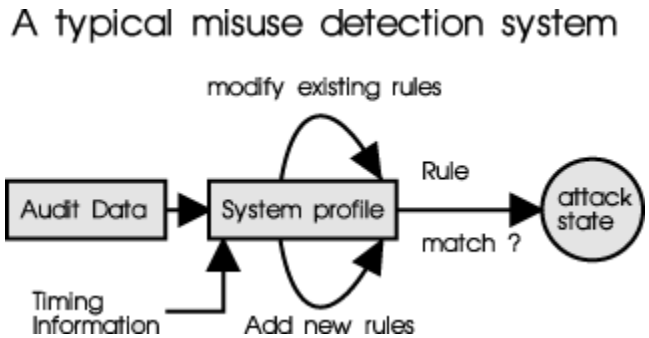


Figure 2.1 – A Typical Misuse (Deductive) Detection System [13]

Anomaly detection (inductive detection) involves the idea of defining what "normal" is. Abnormalities can be defined only if the definition of normal is clear. In general, a combination of one or more of the system events, resource thresholds, etc, are recognized as possible problems and, by induction against a larger set of such events (or a time-series), are then confirmed as an anomalous event. The technique assumes that all intrusions have an abnormal behavior associated with them, and that all abnormal behaviors are intrusions. However, this is not always true. For instance, Susan is a university full-time accountant who always comes to work at eight in the morning and goes home at five in the afternoon. A simple anomaly detector will assume that departure from that is an anomaly and will flag it as an intrusion. If Susan works late one day, she may get a security warning message, account lockout, or an annoying phone call from security staff because her legitimate behavior is categorized as a possible attack. In short, although anomaly detection does not require scanning for signatures, it may causes false positives (false alarms) at an unacceptable rate since normal behavior can be extremely hard to define.

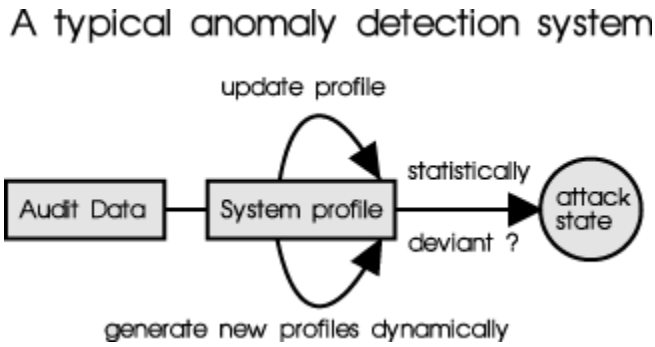


Figure 2.2 – A Typical Anomaly (Inductive) Detection System [13]

Several new techniques have evolved in the last few years based on the above two models to track the changes in networking and computing technologies and to aim for

higher success rate in the area of intrusion detection. These techniques include Intrusion Detection Agent System [11], Strict Anomaly Detection [15], Stateful Intrusion Detection [15], and Discriminant Analysis based detection [16].

2.1.2.4 Statistical and Behavior Analysis

Statistical IDS is a major form of anomaly intrusion detection. It uses statistical algorithms to categorize and analyze network traffic in order to detect anomalous activities. For example, statistical approach may monitor the variance of the present profile with respect to the originally generated behavioral profile. If a statistically significant deviation is found, an alarm is sounded. This approach requires creation of user and systems profiles [57] and definition of appropriate "deviation" metrics. The system analyzes the monitored data/users against the "nominal" profiles, and tries to identify potential problems [17]. The diagram below depicts how this technique works in general.

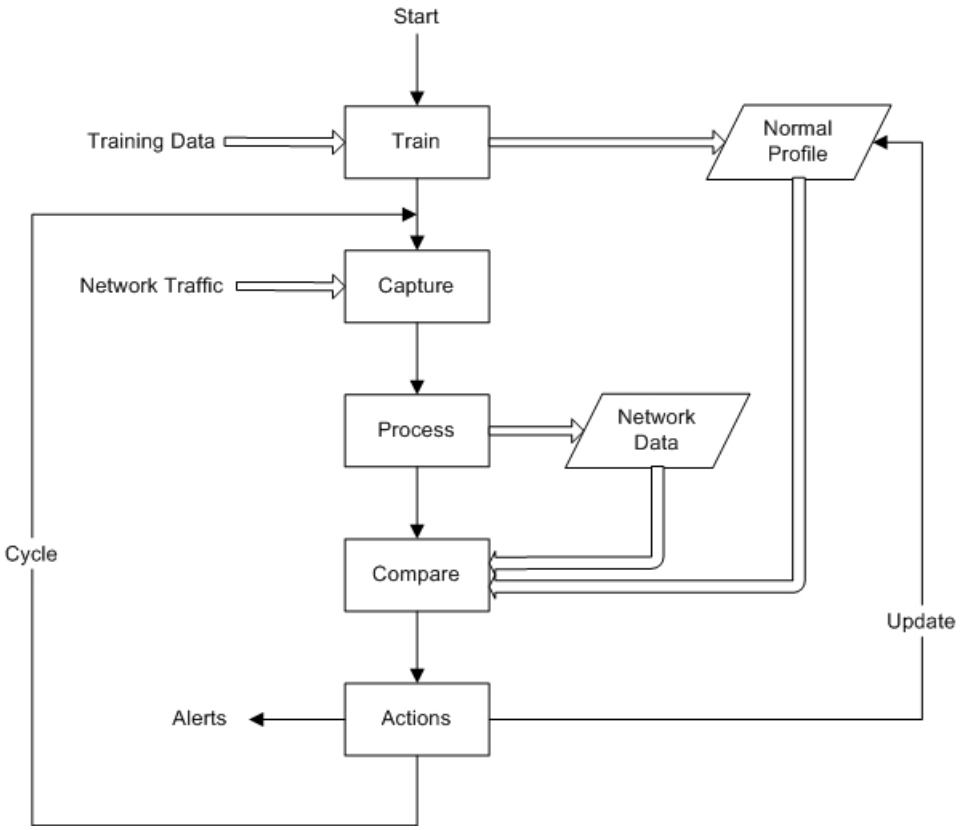


Figure 2.3 – Generic Statistical IDS

The first task of a statistical IDS is to train the system in a training environment, or a set of training data that derive from the normal operational profile, which defines the normal behavior and activities. The profile normally defines a set of variables, and their associated values that represent normal activities. After the operational profile has been successfully created, parameterized and used to train the detection mechanism, the system can be put into real action using captured network data. The system then processes the data to create the "new or current" operational profile. The next step is to analyze the difference between the "normal" profile and the "current" profile. This can be done using a variety of statistical measures and methods. If there were significant differences, or certain thresholds are exceeded, the profile is treated as anomalous and an alert is issued. After a defined period of time, the system

updates the "normal" operational profile with new information and the process continues. [6][17]

This general model contains certain shortfalls. For example, in non-stationary systems that often do not allow a normal distribution, statistical tests based on normal distribution can generate incorrect decisions [23]. Recent work has introduced several advances in statistical anomaly detection. Some examples include Haystack [19], IDES [18], MIDAS [20], NIDES [21], and HIDS [23]. Some other specific statistical models include the following [22].

Threshold Model **-** The simplest form of statistical model. It compares a new observation against a predefined fixed limit. If the observation breaks the limit, the model raises an alert or performs some other action.

Mean and Standard Deviation Model **-** Probably the most adopted statistical model. This model calculates the mean and standard deviation of a new event (or perhaps a higher order moment) based on previous observation, often a series of previous observations. If the newly calculated parameters fall outside the confidence interval, for example as defined by the Chebyshev's rule [44], the situation is treated as an anomaly.

Multivariate Model **-** Similar to the above model, except multivariate model is based on correlations and relationships among two or more metrics [44].

***Markov Process Model* – This model is** closely related to, and sometimes being the driving theory behind stateful intrusion detection mentioned earlier before, Markov process model defines the probability that an observation ought to happen based on previous state and transition matrix. If the probability is too low, which means an unlikely event has just happened, it identifies the particular event as a possible attack [44].

Behavior-based intrusion detection techniques, by definition, assume that an intrusion can be detected by observing a deviation from normal or expected behavior of a system or the users. In fact, behavior-based IDS is in essence a statistical IDS which may be based on quantitative or qualitative analysis of a broad profile of system behavior. For example, a rule-based IDS or a knowledge-based IDS would qualify as such. Just as before, the model of normal or valid behavior is extracted from the reference information collected by various means. The intrusion detection system later compares this model with the current activity. When a deviation is observed (and possibly confirmed), an alarm is generated. In other words, anything that does not correspond to a previously learned behavior (quantitative or qualitative) is considered intrusive. Therefore, the intrusion detection system might be complete (i.e. all attacks should be caught), but its accuracy is a difficult issue (i.e. one can get a lot of false alarms) [24].

Advantages of behavior-based (broad inductive) approaches are that they can detect attempts to exploit new and unforeseen vulnerabilities. They can even contribute to the (partially) automatic discovery of these new attacks. They are less dependent on

operating system-specific mechanisms. They also help detect 'abuse of privileges' types of attacks that do not actually involve exploiting any security vulnerability. In short, this is sometimes the paranoid approach: Everything that has not been seen previously is dangerous [26].

The high false alarm rate is generally cited as the main drawback of behavior-based techniques because the entire scope of the behavior of an information system may not be covered during the learning phase. Also, behavior can change over time, introducing the need for periodic online retraining of the behavior profile, resulting either in unavailability of the intrusion detection system or in additional false alarms. The information system can undergo attacks at the same time the intrusion detection system is learning the behavior. As a result, the behavior profile may contain intrusive behavior, which then is not detected as anomalous [24].

StealthWatch from Lancope is an example of a recent commercial product that utilizes behavior-based intrusion detection techniques [25].

2.2 Tools

Since network traffic analysis and intrusion detection is not a new field, numerous tools have been developed. A few tools are listed in sections below.

2.2.1 Tcpdump

Tcpdump is a utility that captures and dumps network traffic information [27] using libpcap packet capturing library [28] and BSD Packet Filtering (BPF) [45]. It can also

take a pre-recorded pcap file as traffic input. The headers of packets it reads either from a file, or from the network interface in real-time, are printed on screen or be saved to a file. Tcpdump runs in continuous mode until either stopped by a user or until a preset number of packets have been received.

The original Tcpdump runs in UNIX operating environment. A variation of Tcpdump developed for Windows, called WinDump [29] which uses library WinPCAP (the libpcap for Windows) [30] is also available.

2.2.2 RUM

RUM (Resource Usage Monitor) is an open-source network monitoring application developed at NC State [31]. It provides packet capturing capability similar to Tcpdump, using the same libpcap library. But in addition to simply capturing and displaying packet information, it presents the results through a web interface that allows application of human-administered anomaly detection models ranging from simple threshold-based analyses, to more human-based complex experiences.

RUM runs in statistical mode, rather than in a continuous mode found in most other traffic analysis tool. It samples the network for a certain amount of time, say every several minutes. Then, each sample collected is sorted according to network and subnet information. A web interface presents the current and historical information in the form of reports and RRD [46] is used generated graphs. RUM can also detect OS signatures (based on SNORT information) and generate alerts based on the threshold model. In its present form, RUM is intended to be used by a network administrator in

real-time and more complex problem detection algorithms are meant to be administered by humans through an experienced mental model. When used by a well trained operator who also has access to a behavioral knowledge base about the system under protection (e.g., NC State external connections), it can detect almost any attack and/or node anomaly that rises above environmental "noise" level. Another characteristic of RUM is that it relies ONLY on the publicly available header information and does NOT access packet content data that could be considered private.

One of the motivations for the present thesis work was to capture (reproduce) some of the experiences using an additive multi-parameter quantitative (statistical and threshold) model to automate anomaly detection beyond threshold reporting.

2.2.3 SNORT

Most of viable existing IDS tools and applications are proprietary and copyrighted. Snort is probably the best known open-source intrusion detection system available [32]. Snort is a lightweight network intrusion detection system capable of performing real-time traffic analysis and packet logging on IP networks. It is a misuse IDS based on deductions - pattern matching and signature scanning. It uses the same packet capturing library as Tcpdump, and it examines each packet against a set of rules. The rules define patterns and signatures for known intrusions, as well as the corresponding actions that Snort should take if a rule is matched. It can perform protocol analysis, content searching/matching and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB

probes, OS fingerprinting attempts, and more [33]. However, to be able to identify most of these problems, Snort has to examine the actual data content of each packet, which may violate certain privacy policy. In addition, with all the required processing and analyses it normally runs Snort may performs poorly on large-scale, high speed networks, particularly if it is run in continuous mode.

Snort uses a flexible rules language to describe traffic that it should collect or pass, as well as a detection engine that utilizes a modular plug-in architecture. Snort has a real-time alerting capability as well. It has an alerting mechanisms for syslog, a user specified file, a UNIX socket, or a WinPopup message to Windows clients using Samba's smbclient [32][33].

2.2.4 Nmap

Nmap ("Network Mapper") is another open-source command-line tool for network exploration and security audition [34]. It is a pro-active tool unlike RUM (version before current improvements), Tcpdump, and SNORT that are reactive tools. It allows the user to perform scans on a network to determine which hosts are up and the services they are running (ports they are listening to). It supports a large number of scanning techniques including UDP- and TCP- based connect(), TCP SYN (half open), FIN, and many more. It is also capable of identifying the operating system running on the host being scanned. Scanning against a specific single host instead of an entire network is also supported [35].

As part of current work, Nmap was integrated into the Dynamic Intrusion Detection System to allow administrators to perform optional active probing against any specific host.

2.2.5 Other Tools

Besides the applications and tools mentioned above, there are many more other similar tools. For example, Ethereal is a packet capturing tool very similar Tcpdump, but with a graphical user interface [36]. Network Flight Recorder (NFR) is another misuse network intrusion detection system that works much like Snort, but uses more in-depth rules [37].

Of the numerous existing open-source tools, RUM is one of the few that is strongly flow-based, supports high-speed data capturing, runs in statistical mode, and does not examine packet payload (to ensure privacy). These features are one of the reasons why RUM was originally developed when other tools were available.

Chapter 3 – Approach

3.1 Overview

Every network is different. Even a particular network may be quite different, topologically and logically, at different points in time. In addition, network usage profile is a constantly changing function of time. Therefore, it is not easy to define what "normal" operational profile of a network may be and how to define anomalous "distance" from that normal behavior. In addition, the character of the threats to the network also changes with time (whether due to resource usage without malicious intent or due to actual intentional attacks). Hence, in order for an intrusion detection system to be successful, the key is that the intrusion detection system needs to adapt dynamically not only to the changing operational environment of the network it is monitoring, but also to the changing nature of the threats to that network.

On the other hand, experience at NC State is that a trained human operator can spot intrusions, and compromised hosts, relatively easily using RUM provided that the disturbances caused are above "noise". However, capturing this set of experiences into an algorithm remains a problem, in part because a human operator may use extra knowledge (or context knowledge) that is not part of the information available within network traffic alone. Identification of sub-noise intrusions is somewhat more difficult since it appears to require even more context knowledge, and considerable amount of backtracking and inductive effort.

However, the basic premise of the current work is that, given appropriate metrics, measurement tools, statistics, and context information, the best discriminator of really new anomalies and threats still is an experienced network security administrator. The human mind is highly adaptive to changes. An administrator knows (explicitly, but often implicitly, sometimes) how a network should behave according to past experiences, but also based on a constantly changing body of knowledge related to the current network environment and context that may be difficult to quantify and cast as explicit set of rules.. Yet, it is extremely inefficient, even impossible, for an administrator to examine each single device on any network of import to identify in real time potential risks or breaches.

3.2 Architecture

The essence of the solution towards which this work is contributing is that of human IDS reasoning augmentation, rather than replacement of a human administrator by an automated IDS. While computer-based solution is extremely efficient when it comes to deductive intrusion detection (i.e., signature matching), it leaves a lot to be desired where inductive intrusion detection is concerned simply because autonomous back tracking and reasoning that inductive detection requires is not yet well developed in the domain of artificial intelligence. Hence, this work aims at increasing efficiency of a human IDS operator through pre-screening of information in a manner equivalent to what an experienced administrator may do. The algorithms and software developed here (i.e. IDS) are intended to act as a tool that assists the administrators to identify the problems. In other words, the approach is to attempt to mimic an experienced

human administrator reasoning and leverage any extra "knowledge" of the correct behavior of a network.

From a high level view, the system has four main processes. Data Capture → Data Preparation → Data Analysis → and Result Reporting. The heart of intrusion detection falls on part three – data analysis. By analyzing appropriately sorted and categorized data in a dynamic way, one can determine the problems. Analysis may take many forms, from visual clues, to threshold and range triggers, to pattern recognition. One important item to note is that it is the belief of the author of this thesis that, while collective behavior of the network is important, it is the individual behavior of a network component (node) that can make or break the security of a particular network, i.e., the weakest link is the one that counts. Hence, the design approach used is the one that recognizes collective behavior, but focuses on problem determination at the individual node (host) level.

3.2.1 Experience

To achieve this goal, an end-host specific collection of information (knowledge) is needed – an end-host database, or and end-host context database. The database consists of roles for each host, and in the future of additional context information. It identifies whether the host is a server, a client, or performs some other function (e.g., a sniffer or a scanner). It can also contain information about its allowed activities (e.g., ports that it is allowed to have open), its operational profile (e.g., frequency and periodicity of use of certain network functions, ports, protocols), and so on. For example, it would be normal for a server to run certain services it is registered for. In

addition, information is also needed about other, more mechanical components, such as the hardware, or MAC address of the host, the subnet and the network the host belongs to, and so on.

However, in addition to "personal" knowledge, the IDS "experience" module does need to have up-to-date information about the environment in which a host operates, and about the collective behavior of its peers. That is, the IDS system needs to be able to gain knowledge of each neighborhood whether defined by an IP subnet, an organizational network, or an overlay entity that may span many organizations and networks. For example, how many servers there are in a particular subnet, and how they should behave. This basic knowledge that the system has about the network environment (or the context of the host under scrutiny) and the hosts themselves can then be used to make more accurate inductions on whether a particular set of information represents an anomaly or not.

3.2.2 "Brains"

Now that we have given the intrusion detection system "experience", we need to give it a "brain". The "brain" is a set of algorithms used to analyze network behavior and react to it. The deductive part of the "brain" is relatively easy, for example one can run SNORT or another deductive package on the captured data. In fact, RUM already has the capability of doing some of that. It has built in operating system recognition software (based on network packet signatures), it has the capability of running SNORT (but this capability is not really usable in real-time right now – at least not at the speeds we need), and it has threshold model that allows identification of a

surprisingly large number of network-related problems that exhibit flow or volume excesses[2] [31]. The inductive part is much more challenging.

Humans ask questions when analyzing a problem. For example, when a network administrator tries to identify whether a host is acting normally or not, he/she may ask:

- What is this host? Is it a server, or a client? Is it supposed exhibit a special behavior today? If it is a server, what services are allowed to run? … and so on.
- Does the host have any obvious problems or violations? What is its history?
- Does it behave differently than other hosts in the same network? How differently?
- Does it behave differently than itself based on its historical record? And how differently?
- Is there a change in its network status?
- Who maintains this host?
- Etc.

These questions are raised based on the network administrator's experience[3] of monitoring and analyzing the NC State University network traffic. With answers to these and other questions, he/she can make a more educated decision on how likely it

[2] For example, almost all hosts that in NC State environment exhibit more than about 300 flows per sampling period of 60 seconds are in the category of machines with problems (either security breaches or misconfigurations). Only recognized (registered) flow generators – such as large-scale web-servers, Planet Lab machines and Internet2 quality of service probes legitimately fall on the high-side of that parameter. Similarly, almost any host that emits more than about 1 Mbps of traffic over a protracted period of time, and is not a registered server, is suspect.

[3] Presented by the author of this thesis and confirmed by the thesis advisor, Dr. Mladen Vouk, who is also the Associate Vice Provost for Information Technology at NC State University and the Technical Director of the N.C. State Center for Advanced Computing and Communication.

is that the host is problematic. Of course, other questions may be necessary or more appropriate for a different network. However, the current research study is based on the NC State University's network and other networks are outside of the current scope.

The process envisioned for RUM IDS discussed here was designed to support this type of queries and this type of reasoning. Analysis goes through six different steps, some deductive some inductive[4].

1) First, end-host database is consulted to get host context information, such as the role of the host. Knowing what services are allowed on the host and what services are actually running, the system may be able to deduce possible violations and recognize ports that may be open by known Trojans, viruses, etc. For example, Figure 3.1[5] illustrates a report RUM provides by intensity of flows observed on the outgoing NC State gateway. Of the top 20 flow[6] emitters (at the time of the snapshot) only one is an official web-server (item 6). All other units are subject to further scrutiny.

[4] **Deductive** reasoning uses a collection of known facts about a system and a set of known facts about intrusions and tries to match them (e.g., using pattern matching methods – say matching of known virus signatures against the captured information). If the match is within a pre-determined "distance" it is declared as success. If it is not, the process "backtracks" to the captured information. Information is categorized, exploratory methods are used extensively (e.g., statistical tests, additional context information), and a hypothesis is formed, hypothesis is then confirmed using **inductive** methods (basis, behavior for a general step k, proof for general step k+1, etc.), or some other form of confirmatory analysis (statistical methods are acceptable). Backtracking over logs, sequences of information, collective behaviors, etc., is used extensively to both form a hypothesis and provide enough information to prove it or disprove it and close the induction.

[5] Individual IP numbers may have been modified to protect privacy of source and destination nodes.

[6] A flow is an ordered quadruplet (source IP address, source port, destination IP address, destination port)

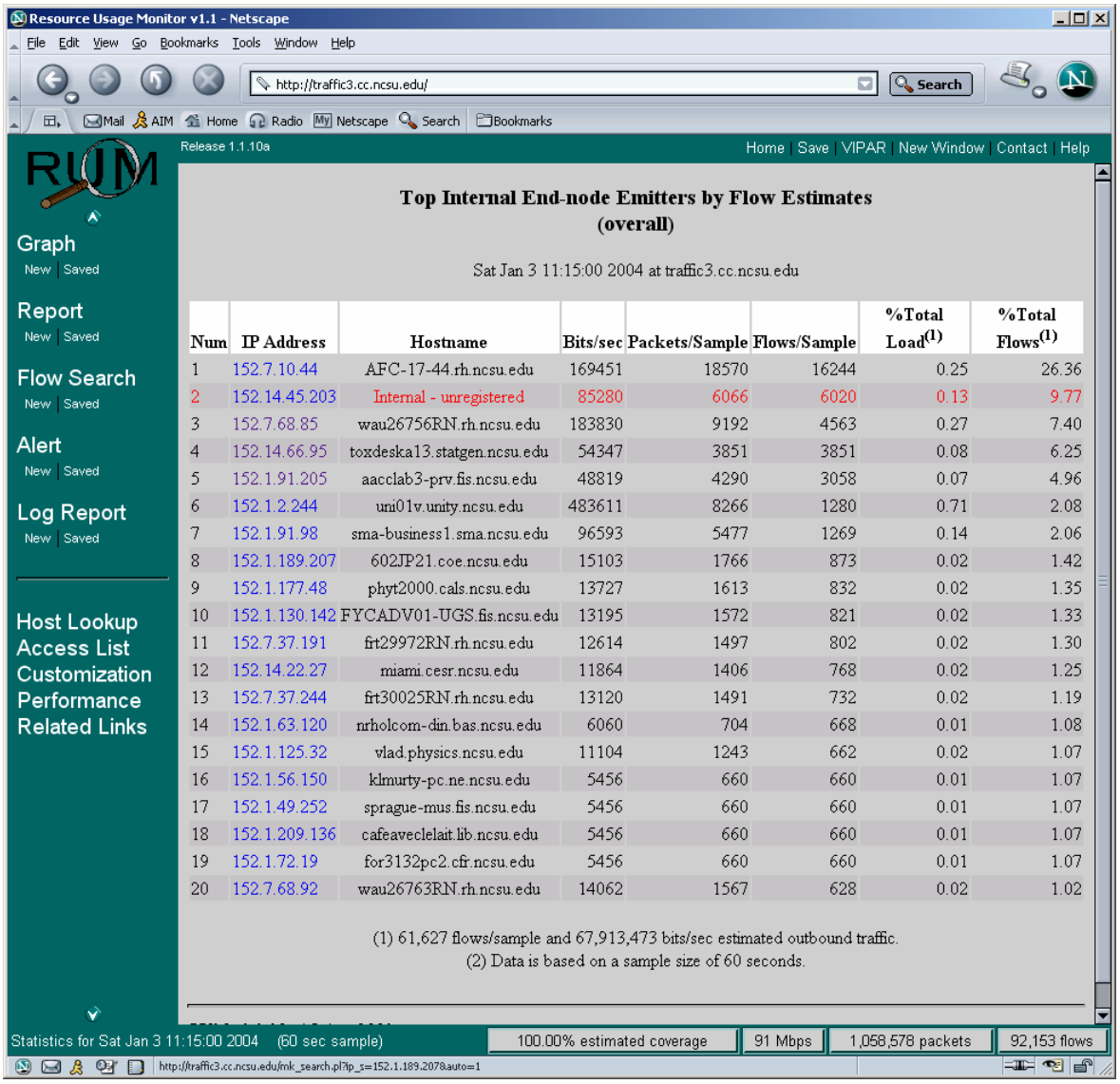


Figure 3.1 – Snapshot of Top 20 Flow Emitters

An item that stands out is number 2, an unregistered source – probably a machine that has been corrupted and is spoofing the IP address

2) Second, a dynamic threshold model is used to check for obvious violations such as high flow or high bandwidth utilization. In fact, the list of top 20 bandwidth and flow users is a routine non-threshold specific way of identifying potential problem machines. While before it was mentioned that rule-of-thumb thresholds

for NC State environment are 300 flows per 60 second sample or 1 Mbps, these thresholds may be dynamic, and generally should not be hard-wired (preset) to a certain value. They may adjust according to the behavior of the entire network. Certainly, when humans judge thresholds, the limits are fuzzy and are adjusted rapidly based on the exploration and analysis. This helps reduce false alarms.

For instance, a university network has much higher bandwidth utilization during school time than vacation time. A value that is set to work for school time may be too high to use for vacation time, and vice versa. Figure 3.2 and Figure 3.3 illustrate this well. They show snapshot of flows and load respectively observed during the last few weeks of 2003. The onset of end-of-semester school break is obvious. Another, interesting thing to note is the large amount of non-IP traffic observed for outgoing flows. Unfortunately, most of these flows were probably generated by computers infected by one of the recent viruses or worms that attack Windows machines through port 135 such as the W32.Welchia worm (http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.worm.html)

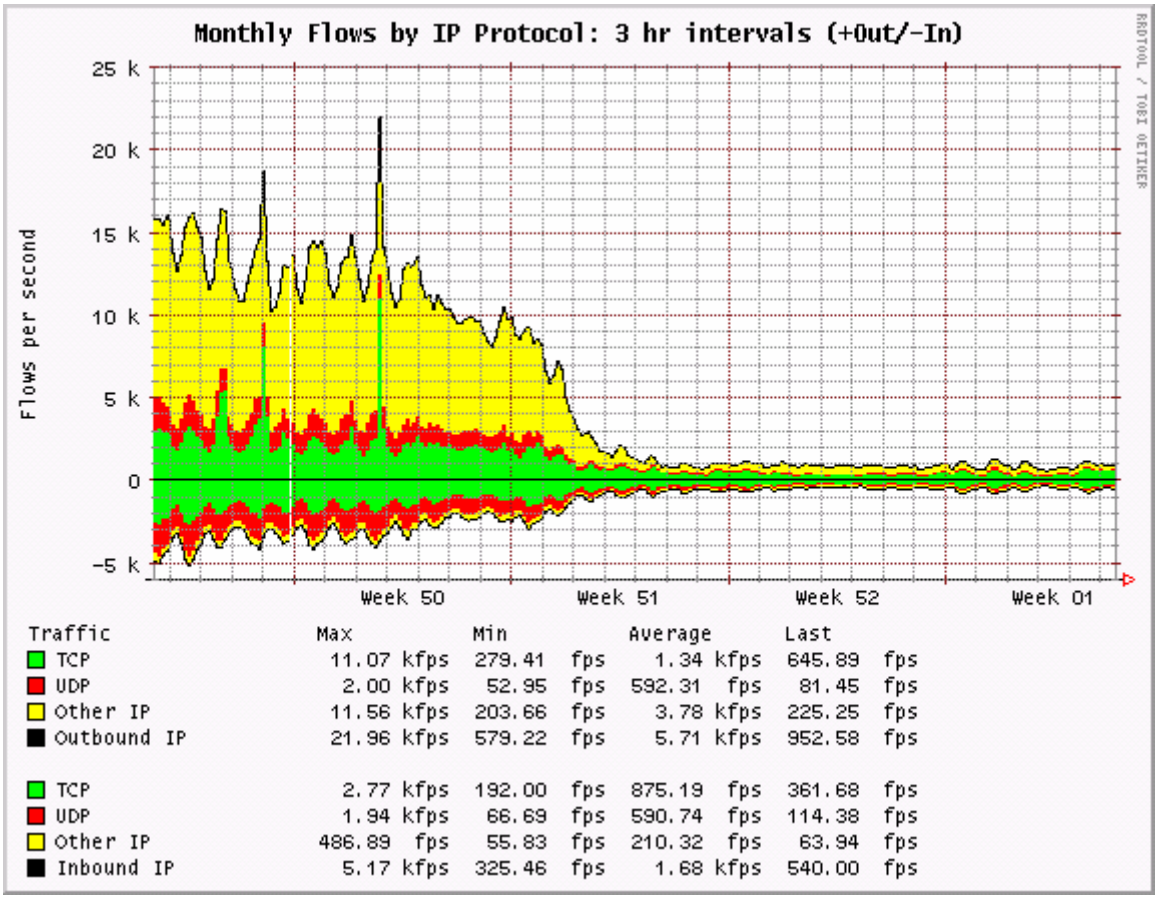


Figure 3.2 – NC State Gateway Flows – end of 2003
(Above zero outgoing, below zero incoming.)

Certainly, just examination of the flows associated with the top contenders can yield interesting results. For example, Figure 3.4 shows flow details for item 1 for in Figure 3.1. It would appear to either be an IRC server, or a collection point for information from a number of infected systems. Obviously, confirmation of the suspicion does need additional active work (such as scanning). However, 16,000+ connections that this host seems to have opened are excessive by any standards and this is a genuine anomaly that needs further investigation. In fact, further examination of the flows revealed attempted connections to SWAT – Samba Web

Administration Tool – port 901 on a large number of different machines; quite certainly a scan for vulnerabilities coming from an infected machine.

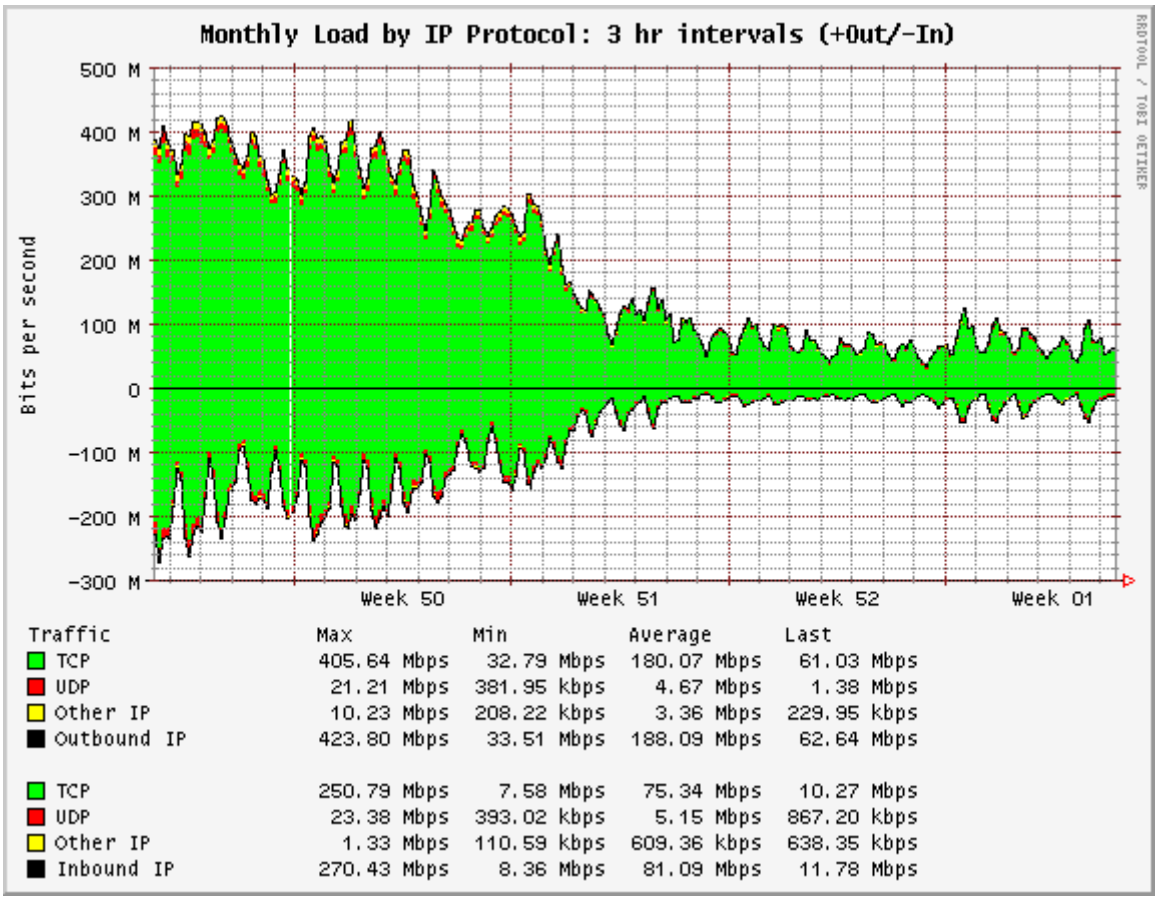


Figure 3.3 – NC State Gateway Load
(in Mbps – above zero outgoing, below zero incoming)

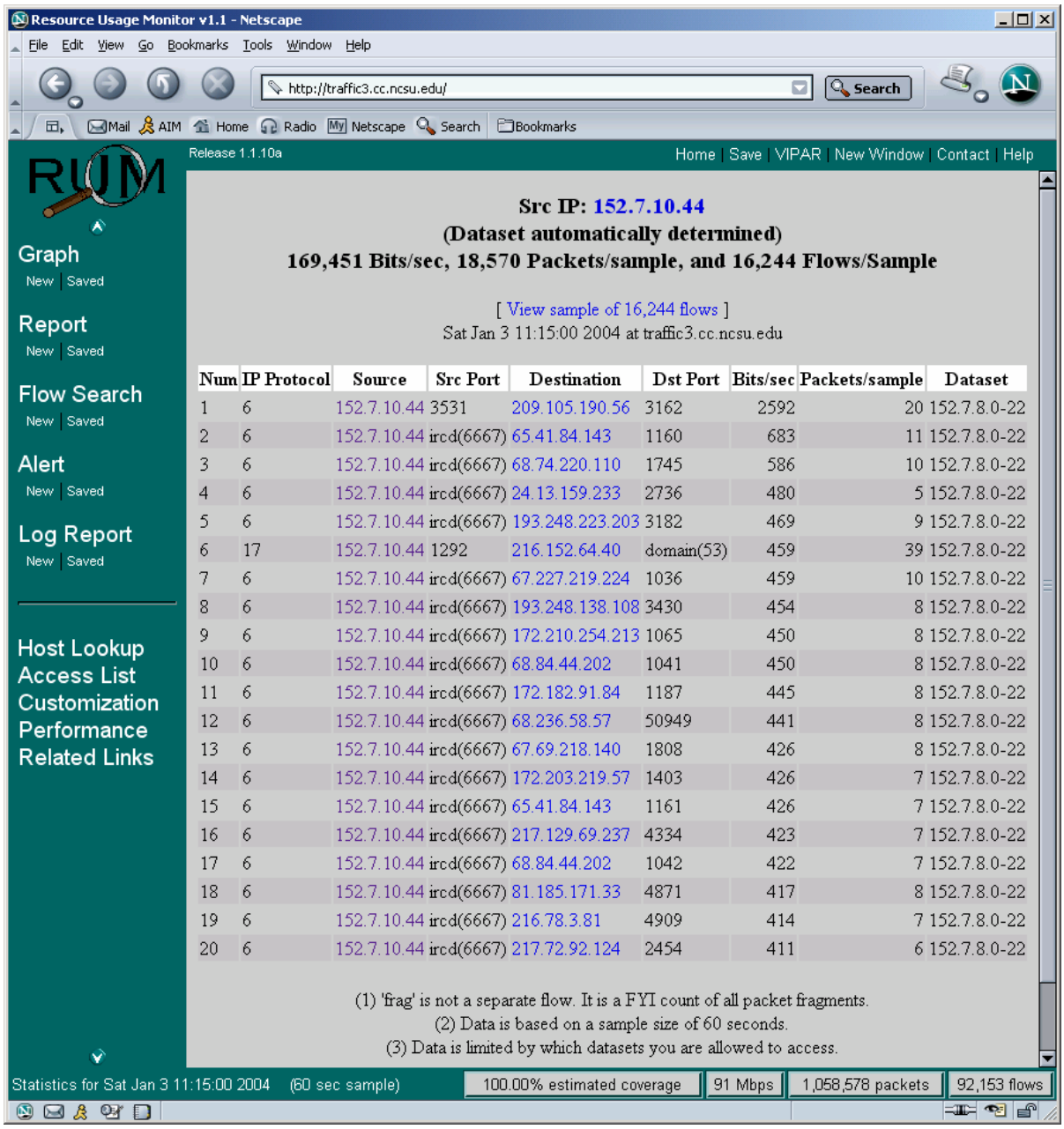


Figure 3.4 – Details of item 1 from Figure 3.1
(The unit appears to be an IRC server, although with 16,000+ flows, it is probably corrupted and the connections may be connections feeding information from other corrupted machines.)

Details of item 6 from Figure 3.1 show, as expected, regular http communication provided by a registered NC State web-server (Figure 3.5). Moving down the list – click on item 15 (giving Figure 3.6) reveals that it is a host that appears to be

serving on port 3531. The behavior is unusual, probably a peer-to-peer server that may or may not be voluntary; or probably merits a chat with the system administrator of the machine. Finally, a quick look at item 19 (details shown in Figure 3.7) shows that the host is connecting to port 135 on many other hosts. Possibly the host is infected with the Welchia worm mentioned earlier. Note that the knowledge that there is an epidemic of Welchia (or some other virus or worm) is context knowledge that is part of the solution process, but may not be normally available to a tool algorithm (unless this information is dynamically forwarded from national security monitoring sites).

Figure 3.5 – Details of item 6 from Figure 3.1
(The unit is a registered NC State Web server.)

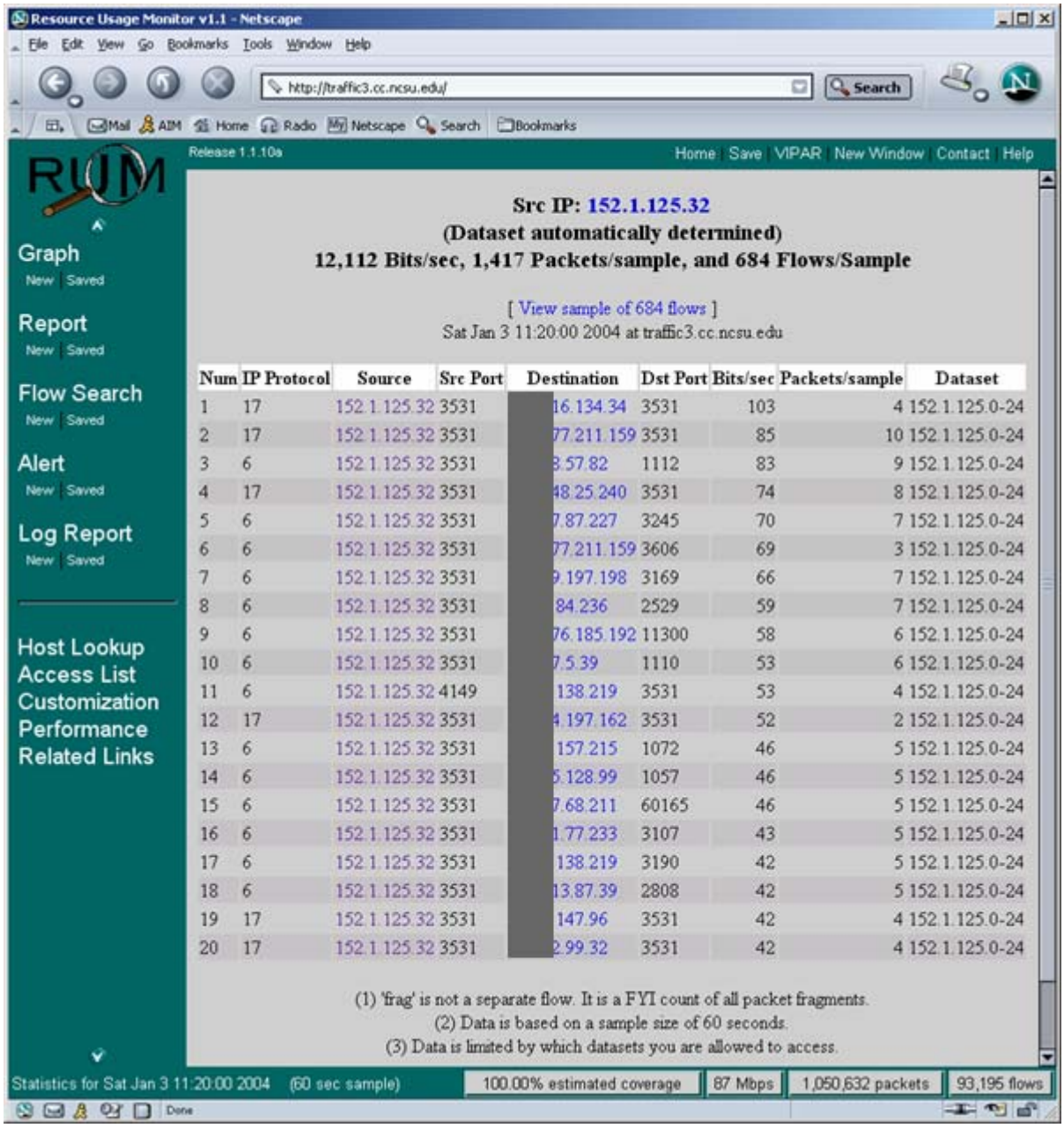


Figure 3.6 – Details of item 15 from Figure 3.1
(The unit appears to be serving on port 3531.)

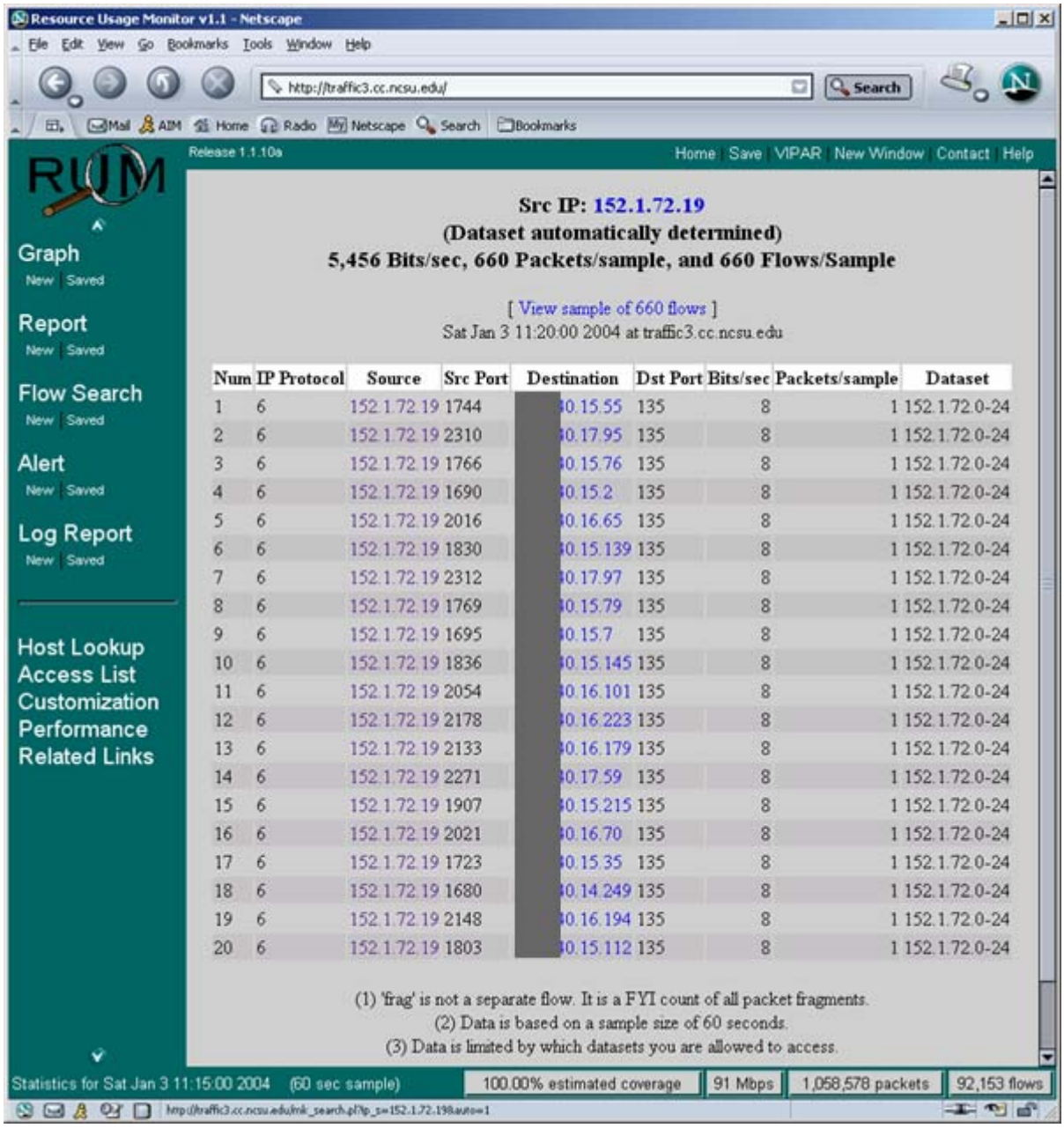


Figure 3.7 – Details of item 19 from Figure 3.1
(The unit appears to be scanning port 135 – a fingerprint for Welchia worm?)

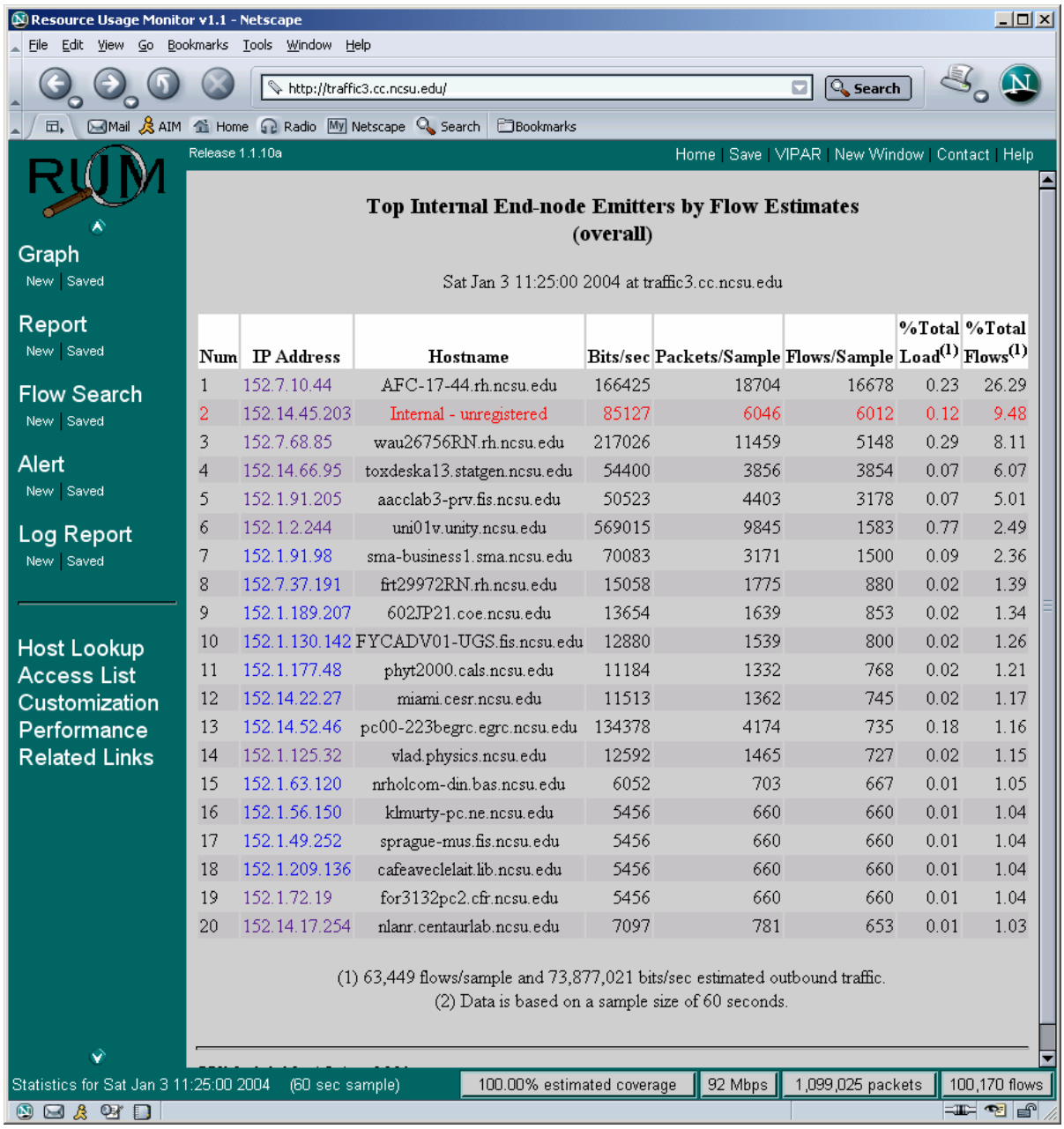


Figure 3.8 – Snapshot of Top 20 Flow Emitters (10 minutes later than Figure 3.1) (Note the slight change in the membership of the list, e.g., item 20.)

Another snapshot of the top 20 flow emitters taken 10 minutes later than Figure 3.1, is shown in Figure 3.8. This second snapshot shows a small change in the membership of the list (e.g. item 20 is replaced with a different host). To take a closer look at the host the just jumped onto the list (Figure 3.9), one can see that

the host is communicating with various Internet hosts with a small amount of ICMP (1) traffic and a large amount of traffic of unassigned IP protocol (169).

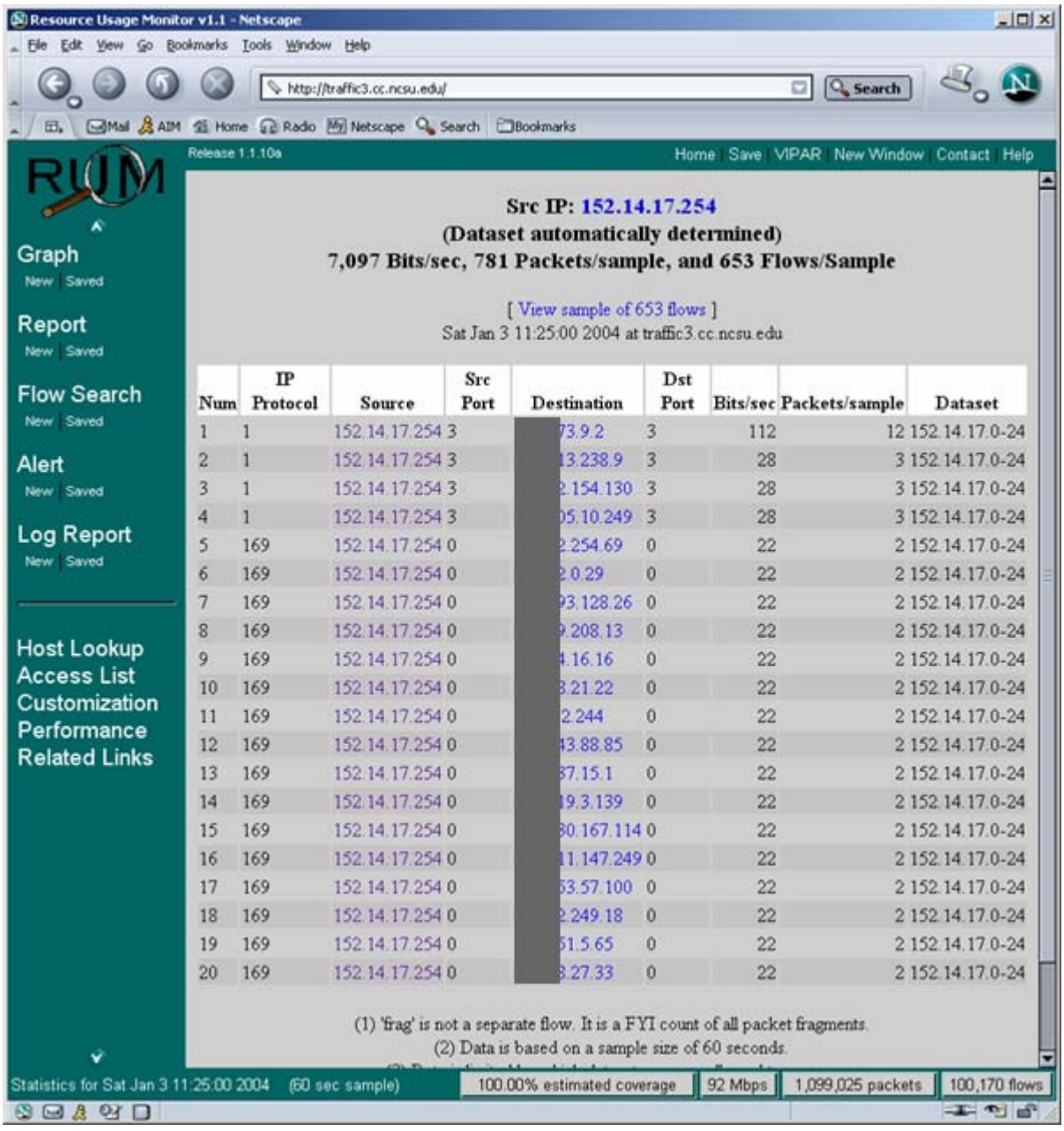


Figure 3.9 – Details of item 20 from Figure 3.8

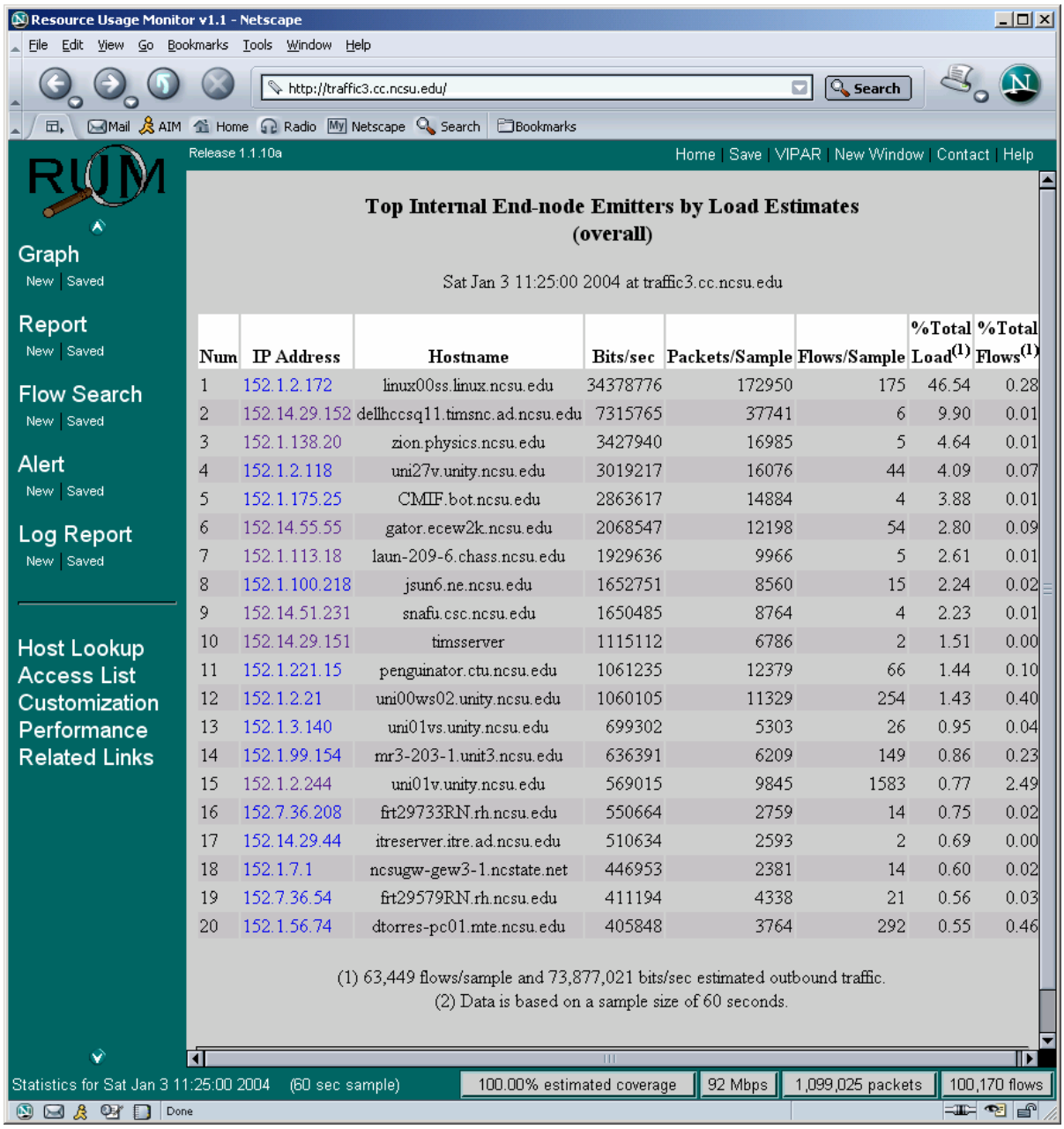


Figure 3.10 – Snapshot of Top 20 Load Emitter

Aside from top flow emitters, top load emitters can also be examined with a separate list (Figure 3.10). This list shows the hosts that are consuming most of the network bandwidth. Except for registered servers, other hosts that appear on

the top of the list are most likely problematic in some way. An example is shown in Figure 3.11.

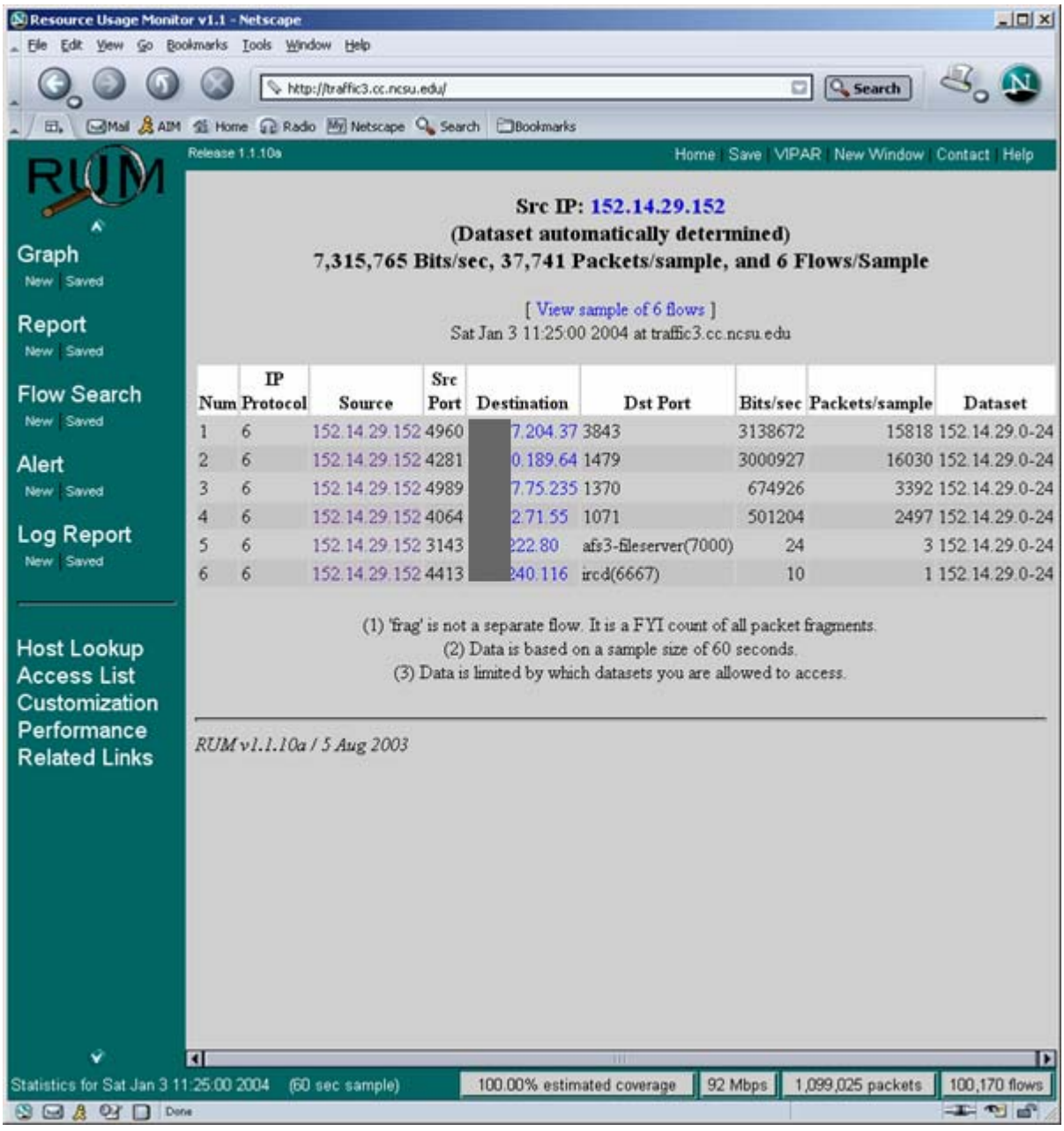


Figure 3.11 – Details of item 2 from Figure 3.10

3) Third, process calls for analyzes of the network (neighborhood) events, for frequencies and statistics generated on-the-fly, and then compares them to those of the host. This is done to determine how differently the host is behaving from the network as a whole. This essentially provides a "distance" estimate of the host state with respect to its environment.
4) Next, similar computation is done using values calculated on the host itself for a previous data sample to determine if there is any significant change in its own behavior. This is an estimate of the rate of change of the host state with respect to itself.
5) Then, the system uses the traffic flow input/output ratio aside with the unique number of hosts and ports the host talks to (or from) to determine attackers and victims of host/port scan and denial-of-server (DoS)/distributed DoS.
6) Finally, the system determines status change of the host under consideration. Has it been down and it is up, or vice versa? This helps in detection of intermittent network failures, computation of network reliability and availability, and detection of "hopping" scans and spoofs.

Each of the five stages of analysis generates anomaly indicators (or index numbers) for the host. The first two tend to have a deductive nature, while the last three tend to be more inductive in nature. However, overall assumption is that the higher a host anomaly index number is, i.e., the larger the difference in its behavior from its peers or from its past behavior, the more likely it is that it is having a problem. The actual computation of the index and its properties is still subject of research. Currently the system uses a linear additive model, discussed below, where the index units

determine index weight based on some empirical experience (discussed in the next chapter). However, the reader has to remember that the project is in the exploratory research phase, and that the best intrusion detection model for support of human assisted intrusion detection is still under investigation.

This work represents Phase I of this project. Phase I goal is to develop a flexible tool for investigation of anomaly indicators that assist in automated and semi-automated intrusion detection.

3.3 Network Node Database

Network node database, or end host database, or context database stores basic information regarding each node/host in the network(s) monitored by the IDS, and, context information in which the host operates. In this study, this database is used to determine the type of the device, and other environmental information relevant to the activity, each time the intrusion detection process is being executed. Thus, for each host in the network, that database includes at least its hostname, IP address, subnet mask, and role. But it may also have historical information, and information on when the host is NOT supposed to be active, when it is in maintenance mode, and so on. The "role" property has at least two major categories—server and client, and can be further sub-categorized into such as web server, email server, VPN server, file server, and so on. This database can simply be one of more files, or a real database such as Oracle.

Hostname	IP Address	Subnet Mask	Roles
csc-ws-rhl-1	152.1.0.20	255.255.255.0	Client
csc-ms-ms-0	152.1.0.21	255.255.255.0	POP3, IMAP, SMTP
... ...	... ...	... ...	... ...
unx-000-001	152.1.7.99	255.255.255.128	HTTP, SSL, FTP
... ...	... ...	... ...	... ...

A Sample End Host DB File

Figure 3.12 – A Sample Host DB File

It is natural for a human analyst to attempt to determine the type of a device, or the role of a network host first as they try to determine if there is an anomaly and before they look further into the actual network data and traffic statistics. This step is very critical in assisting the process that follows (be it deduction or induction) since it gives the system a basic idea of how a particular host should and should not behave. For example, a generic client should not have any well-known ports open or listening; a dedicated web server generally should not be serving anything from ports other than 80 (HTTP) and 443 (HTTPS), an email server should not be sharing files or transmitting FTP data. Any behavior determined to be out of scope will raise some concern. However, most current IDS techniques and implementations do not consider this type of context information. This often leads to false alarms or missed alerts.

3.4 Dynamic Host Policy / Thresholds

Similar to the end host database, which defines the role of a network node, host policy is another part of the context. It defines a series of preset traffic thresholds for a particular host. For example, a web server may have a policy with thresholds defined as 2Mbps outbound bandwidth, 500kbps inbound bandwidth, 1000 maximum connections per minute, etc. By consulting this policy every time, the system can

easily detect anomalies by catching the out-of-bound behaviors. This is illustrated in Figure 3.13.

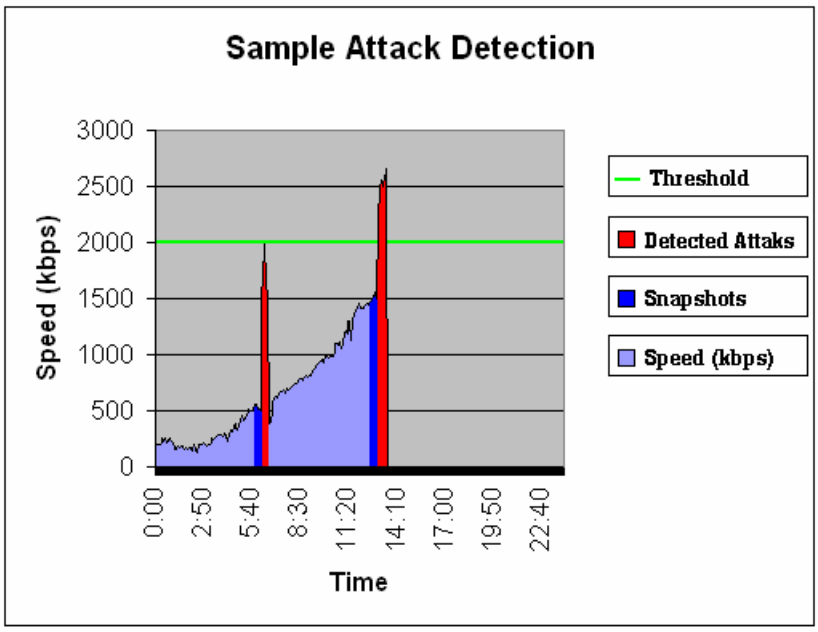


Figure 3.13 – Threshold Demonstration

Simply setting up a threshold may not be enough. The ideal threshold may be different at different times. Thresholds during peak time should definitely be higher than off-peak time. Otherwise, the detection based on the thresholds may be inaccurate. For example, a network with 100 hosts operates at 100Mbps on average during daytime and less than 10Mbps at night. A simple threshold for the daytime based on the average usage should be less than 1Mbps. However, the threshold is too low for the activities at night. If we use 1Mbps as the static threshold, many intrusions and illegal activities will be considered normal. On the other hand, if we use a low threshold that can be used to catch most of the anomalies at night, it produces too many false alarms during the daytime. The traditional approach is to

find a middle point where not too many false alarms are produced and an acceptable amount of illegal activities are left unnoticed.

In order to maximize the capabilities of catching the illegal activities and minimize false alarms at the same time, a dynamic threshold model is needed. In other words, the thresholds must change based on the change of network environment and traffic behavior caused by changes such as time. This dynamic threshold is used to determine the most obvious problems of a host. The system calculates thresholds for traffic flows, packets, and bits (load). In general, the threshold is the mean value of the respective parameter. For instance, a one-minute data sample taken from the NC State network has 61,407 total flows and 1,601,533 packets with 4668 hosts in the sample, the threshold for each individual host is hence approximately 13 flows and 343 packets per host. However, the system does not use these exact mean values as the thresholds. The reason is that the system does not expect to catch all the anomalies by analyzing the thresholds. Rather, thresholds are used to identify the most obvious anomalies and other more complicated ones are expected to be handled in later stages such as network and self comparison and/or input/output ratio. As a result, the implementation uses a relatively high and static percentage as the thresholds (1% for client hosts and 3% for server hosts – more details in section 4.3.2). This way, the system can expect to catch the most obvious anomalies with thresholds and at the same time minimize the chance of false alarms.

3.5 Dynamic Profiling

As mentioned in the previous section, simply using host policy (thresholds) is not enough. It is quite possible that an undergoing attack does not cause the victim node to exceed its predefined threshold (or that an internal attack is clever enough to work only within "noise" level). For instance, a web server is set to have a threshold of 2Mbps. During peak operations, an attack attempt may bring the web server over its threshold. However, at non-peak hours, when the web server only experiences normal traffic of 500kbps, an attack that consumes 1Mbps bandwidth will not trigger an alarm, and will remain undetected unless additional context information is consulted.

The concept of dynamic profiling aims to solve this problem. Unlike traditional statistics-based IDS, the proposed approach does not need to have a training period to "learn" about hosts "normal" operational profile. Instead, the profiles are created dynamically for each host. With dynamic profiling, a snapshot of traffic statistics for each host is kept for a (usually short) period of time, and then this snapshot is updated. It represents the instantaneous operational profile of the host. This way, the system can compare the current traffic statistics with a previous snapshot, and determine any possible anomalies by examining the differences. How the snapshot is computed and how many are kept is a matter of research. Currently, the system uses a snapshot from a period just preceding the current one. This is one extreme. Another extreme is long-term cumulative average. Intermediate options are various moving averages that may be appropriate. To illustrate we use the same example as before – when the web server is serving at 500kbps bandwidth during off-peak hours, the snapshot is taken. When an attack that consumes 1Mbps bandwidth now happens, the system will detect

a 200% increase in bandwidth. As a result, the system will raise an alarm indicating a possible attack.

The following set of statistical variables is used to represent an overall network behavior or the behavior of a particular host.

3.6 Metrics

To investigate dynamic anomaly detection, the following set of statistical variables was selected to represent behavior of an ensemble of hosts (network), or the behavior of a particular host. Specific examples are discussed in Chapter 5.

Flows, Packet, and Bit Counts: formulas, text about usefulness with an example)

Total number of flows, packets, or bits in one sample is normally a useful measure. This number is used in the threshold examination as well as the network and self statistical analysis. If the host's number is significantly greater than that of the network average or its own from last sample, the host is likely to be abnormal. This is observed by studying the hosts using RUM[7].

Average, Min, Median, Max, and Range: (for packets and bits per flow)

These parameters help further describe the behavior and characteristics of the network traffic[7].

[7] Please refer to Figure 3.1, Figure 3.3 and the discussions in section 3.2.2.

Variance and Standard Deviation: (for packets and bits per flow)

The variance measures the "spread" of the traffic data about its average, or how widely values are dispersed from the average number of packets or bits per flow. Standard deviation is simply the square root of the variance. Hosts with frequent bursts in traffic will have a high variance and high standard deviation.

Skewness: (for packets and bits per flow)

An optimal skewness should be zero skewness, which means the average packets per flow (or bits per flow) equals the median (middle value) and equals to the mode (most frequent value). A negative skew indicates the average and median is smaller than the mode. In other words, there are many flows with greater than average packets (or bits) per flow. And vice versa for positive skew. This is illustrated in the distribution diagrams in Figure 3.14. In addition, the skew function is shown in Equation 3.1, where μ is the mean (average) and σ is the standard deviation.

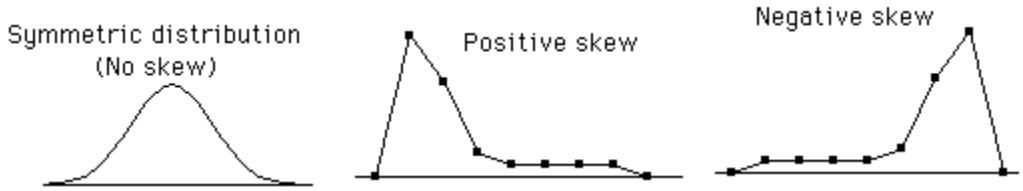


Figure 3.14 – Skew Distribution Diagrams

$$\text{skew} = \frac{\Sigma(X-\mu)^3}{N\sigma^3}$$

Equation 3.1 – Skew Function

Kurtosis: (for packets and bits per flow)

Kurtosis measures the peakedness of the traffic sample. A positive kurtosis indicates a relatively peaked traffic sample and negative kurtosis indicates a relatively flat traffic sample. Figure 3.15 shows the sample distributions for both positive and negative kurtosis and Equation 3.2 shows its function (where μ is the mean and σ is the standard deviation). For example, a host with very small amount of low and high packets (or bits) per flow traffic gets a positive kurtosis, where as a host with a traffic pattern that is distributed relatively evenly gets a negative kurtosis.

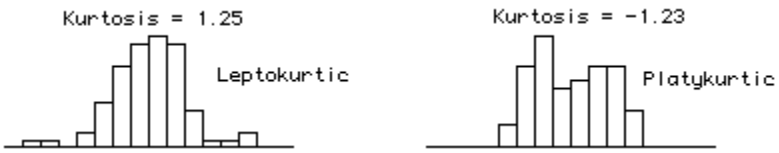


Figure 3.15 – Samples of Positive & Negative Kurtosis

$$\text{kurtosis} = \frac{\Sigma(X-\mu)^4}{N\sigma^4} - 3$$

Equation 3.2 – Kurtosis Function

Number of Communicating Ports:

Total number of communicating ports on the host is used intensively in the port analysis (section 3.8.2.1).

Percentage of TCP, UDP, ICMP, and Other Protocols:

Under normal circumstances, the majority of traffic should be TCP traffic, with very limited UDP, ICMP and other protocol traffic. According to Amogh Dhamdhere's study on Internet Traffic Characterization [58], TCP protocol dominates the traffic on the Internet (95% of bytes, 90% packets, 75% flows), UDP second (5%

bytes, 10% packets, 20% flows), and ICMP most of the remaining. This is also the case for NC State's network environment. A spike of other protocols other than TCP indicates is abnormal and indicates potential problem (see Figure 3.2). Therefore, abnormal host traffic can be identified by examining traffic which deviates from this behavior.

Hosts To, Ports To, Host From, and Ports From:

Number of unique destination hosts or ports that the host is sending packets to (only applies when host is a source) and receiving packets from (only applies when host is a destination).

These four values are used in conjunction with flow input/output ratio to determine attackers and victims for port/host scan and DOS/DDOS intrusions. (See section 3.7 for detailed algorithms and 4.3.5 for implementation).

In addition to the instantaneous snapshot of profiles generated for each individual host, a network profile is also generated dynamically with the same statistical parameters. This way, the system can compare the host snapshots against the profile of the network on which the hosts reside.

3.7 Input/Output Ratio

Another component used in inductive detection is the traffic input/output ratio, defined for flows, packets, and bits, as follow:

```
Flow I/O Ratio = Incoming Flows / Outgoing Flows

Packet I/O Ratio = Incoming Packets / Outgoing Packets

Bit I/O Ratio = Incoming Bits / Outgoing Bits
```

Equation 3.3 – Input/Output Ratios for Flows, Packets, and Bits (Load)

Ideally, in a given period of time, a host communicates using TCP protocol should have identical number of incoming flows and outgoing flows, because TCP is a synchronized protocol [47]. As a result, a host behaving correctly should have a flow ratio of 1 or close to 1. A high ratio number shows that the host is improperly receiving traffic from one or multiple hosts and it is unable to respond, which indicates that the host is very likely a victim of scanning or DOS attack. On the other hand, a number significantly less than 1 indicates a possible port/host scanner or a DOS attacker because it is improperly sending out a lot of traffic without receiving any response. In addition, TCP traffic consumes 95% of all flows on the Internet [58]. That makes this examination every effective.

The hosts to, ports to, hosts from, and ports from in the host profiles are used to further distinguish among different scans and DoS attacks. To explain the detection algorithm in further details, the following diagram is provided.

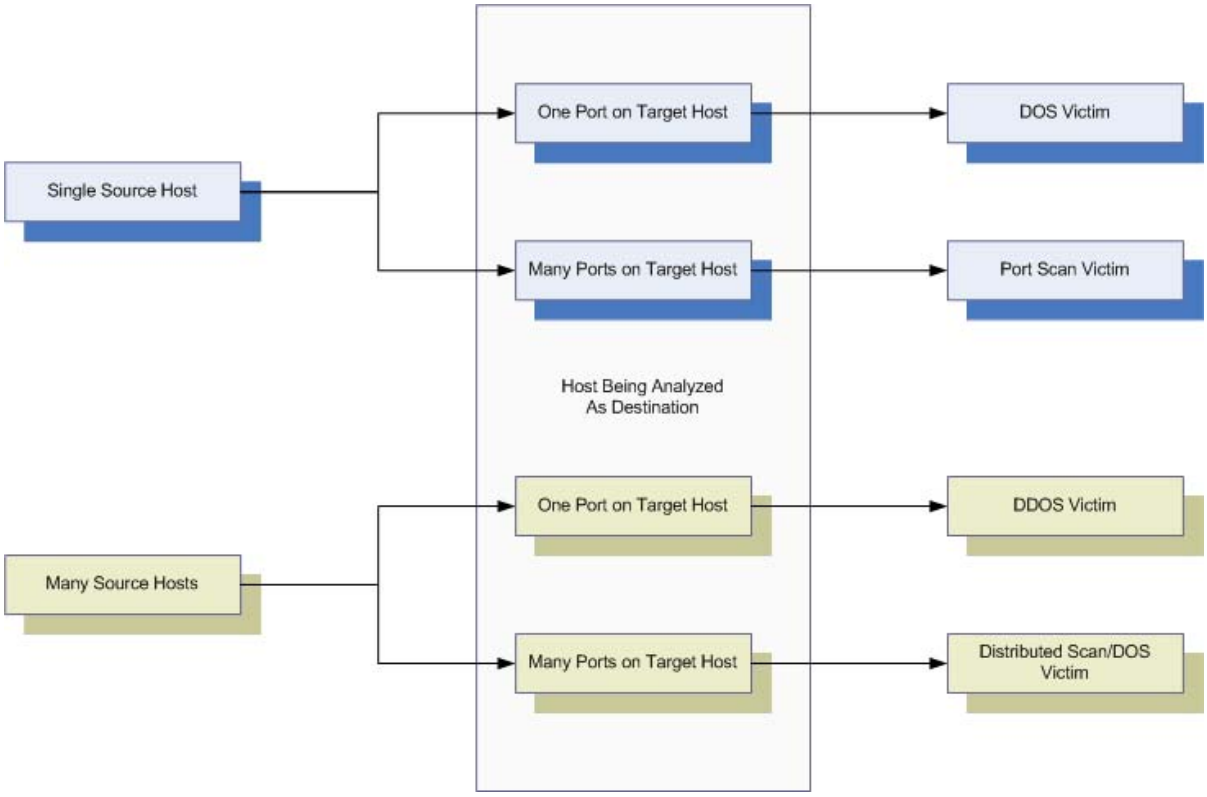


Figure 3.16 – Victim Determination

When a host is having significantly more incoming requests than outgoing responses (i.e. incoming flows much greater than outgoing flows, or flow I/O ratio much larger than one), it may be the victim of some sort of attack or scan. Under such situation, if most of the incoming flows are from a single unique host, targeting a single port (or very few ports) on the local machine, the local machine may be undergoing a DOS attack. Or, if flows from a single host are targeting many different ports on the local host, the local host is a possible port scan victim. Similarly, when there are multiple source hosts sending data to a single local port, that is likely a Distributed DOS attack. And if many hosts are transmitting to multiple local hosts, the local host is probably a distributed scan or DOS victim.

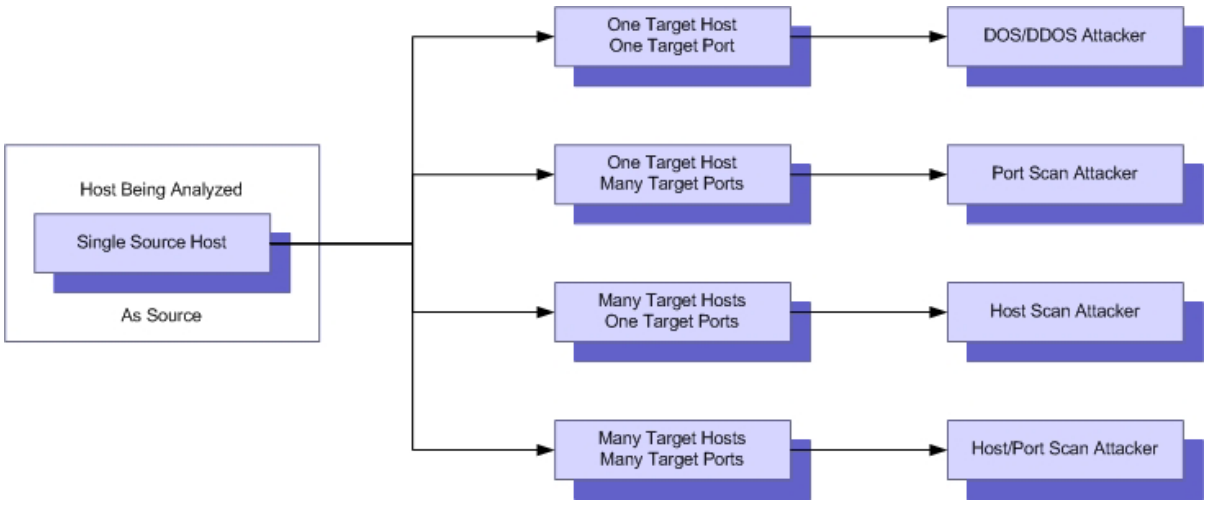


Figure 3.17 – Attacker Determination

In contrast to the previous diagram, this diagram shows the four possible conditions when a host is having significantly more outgoing requests than incoming responses, it may be initiating a DOS/DDOS attack or host/port scan. First, if the local machine is initiating many flows to a specific port on a target machine without getting much response back, it is possibly performing a DOS or DDOS attack. A host scan is very likely when the target is one host and multiple ports, whereas multiple hosts and one port indicate a possible host scan. The case in which there are many target hosts and many ports is the last case, which can be either a host scan or a port scan.

3.8 Node Anomaly Indexing

3.8.1 Rationale

Algorithms associated with node indexing, that is, with a metric that would indicate whether an anomaly is present or not, represent the crux of the inductive engine. They are NOT intended to be used automatically, but strictly as a decision aid for a human intrusion detection expert. This entire research is an exploratory investigation that is based mostly on observational and not statistical assessment of the interactions. The

scope of the work is to find measures intuitive to a human administrator, rather than statistical metrics that would automate the process completely. The basic premise is that fully automated intrusion detection may be possible in the deductive mode, but it is not a realistic option in the inductive mode. Instead, in the inductive mode, the IDS produces a number of (presumably useful) metrics that are then combined with cognitive power and experiences of a well trained administrator to yield results.

Most of the products now on the market seem to lean in the other direction, i.e., they aim at automation. As a consequence, many lack a good human-computer interaction module [48]. This is somewhat surprising. New intrusions are being crafted by clever humans. It seems unrealistic to expect that machines that lack in even rudimentary cognitive powers could match that skill. It is the belief of the author of the thesis that, for effective intrusion detection of really new attacks, it is highly important to actively retain a trained professional in the loop.

As more and more security specialists are professionally trained, it is becoming increasingly more important for IDS to allow enough flexibility for the security specialists' interactions. An intrusion detection system is after all, no matter how efficient and accurate, a tool for the administrators. It should be up to the administrators to decide whether an event is counted as an anomaly or not. IDS should assist the administrators to accomplish the goal, rather than trying to take over the role completely. Unfortunately, the latter is how a lot of existing IDS currently work. Those IDS generate alarms whenever *they* think there is an attack, or an anomaly, and prompt the administrator to investigate. This often overloads the

administrator since there may be hundreds, thousands, or more alerts generated everyday and sent to the administrator. No one is capable of investigating all the alerts generated in such a short time frame. The result is that, often the administrators simply do not look at most of the alerts, and the ones they do look at may not be the most pertinent ones.

We need a better mechanism for alerts categorizing and reporting. The goal is to get the most serious issues in the hands of the administrators first, or provide a convenient way for the administrator to decide which alerts are of higher priority. This efficiently utilizes human resources. Then comes the idea of node anomaly indexing. Simply speaking, the system generates an index value for each node in the network based on the results of the series of analysis explained previously (end host database, host policy, and dynamic profiling). The higher the index value is, the more risky it is likely to be.

More specifically, when the system examines each stage against the network nodes, it will increase the nodes' index value if it finds something abnormal. Let's look at two web servers. Web server A is serving web pages legitimately at 500kbps, but it is undergoing a DOS attack at 1Mbps. It will pass the first two stages because it is a registered web server and it is under the preset 2Mbps threshold. However, it fails the third stage because the system detects a 200% increase in bandwidth consumption compared to previous snapshot. The system increases its index value by 200. Web server B running 1.5Mbps normal operation with 1Mbps DOS attack passes the first stage but fails the last two stages and generates a large index value, say 700. Now the

alert reporting engine will present web server B above A to the administrator because it thinks B is more dangerous. Indeed it is. Although both servers are undergoing the same DOS attack, it may cause more damage to server B since it is over its preset bandwidth limit. In other words, although A is experiencing a DOS attack, since it is not yet over its bandwidth limit, there is a much smaller chance the A will drop legitimate user requests than the chance B has.

3.8.2 Anomaly Index Model

The best intrusion detection model for support of human assisted intrusion detection is still under investigation. The current index calculating algorithm is based on a linear additive model. In other words, the final index value is the sum of several components that each contributes certain index value itself. The specific components can vary with different implementation and/or network scenarios. The work presented in this paper implements the following six categories –

- by examining the communicating ports (and the host database),
- from raised alerts,
- from detected intrusions,
- from network-relative,
- from self-relative changes, and
- by analyzing input/output ratio of host traffic.

These anomaly index categories are chosen based on the analysis approach discuss previously (section 3.3 – 3.7). The first category (index from communicating ports)

represents the results from analyzing the host database. The next two categories may include index generated from alerts and intrusions raised during any analysis stage. Index from network-relative and self-relative comparisons each have their own individual category since they are the two major processes the reveal any undefined anomalies. Similarly, input/output ratio is a vital analysis that detects several DoS and scan attacks with its own anomaly index category. Other categories are possible depending on the actual network and implementation.

In developing a metrics suite, particularly one where the end-goal may be to come up with a single indicator based on a combination of the metrics, it is important to design them in such a way that they span a reasonable range representative of what can be measured and presented, that their weight is such that the metrics can be reasonably combined into more complex metrics. Hence, there needs to be a value that is best suited as the baseline for representing the problem of individual hosts in different index categories.

For each of the six categories of the anomaly index values, a value of 1000 is used to represent the minimal alert level that the host is having some relatively significant problems. This puts, at least initially, all metrics on the same footing. Of course, as the metrics are further developed, we may find that different base-line index values may be more appropriate. Indeed, other values have been tried during the experimentation, from a few hundreds to up to ten thousand, but 1000 appears to be the best initial choice for all six categories.

Initially, values for parameters and metrics were used straight, without any weight. For example, each communicating port generates an anomaly index of 1, each alert and each intrusion also contributes 1 to its index, while the anomaly indices for network-relative (in the current sample frame) and self-relative changes (with respect to previous sample) are taken directly as the percentage changes or as a direct difference in the counts. Finally, the I/O ratio anomaly indices are just that, ratios of Input to Output flow counts, bits per second, or packet counts.

For example, observations of the NC State network showed that the number of ports opened by a single host could range from a single digit to over a thousand in some extreme cases. However, the majority stayed at about several hundred. In general, each problematic host usually does not accumulate less than about 10 alerts[8] or intrusions[8] in each single session. Network-relative changes and self-relative changes generally result in an anomaly index of over 1000 for hosts that are problematic. Finally, input/output ratio for problematic hosts tends to show considerable asymmetry. There may be too many outgoing flows for each incoming flow (e.g., hundred to thousand times more), when, for example, the host is attacking others, or scanning. There may be too many incoming flows for each outgoing flow when the host is under attack.

Since we initially consider all six major anomaly categories to be of equal importance, in order to construct the total anomaly index, the six categories have to be normalized. The base value of 1000 was chosen because that appears to be the threshold observed

[8] See section "Index from alerts and intrusions".

for many of the network-relative and self-relative changes when problematic hosts were involved.

As a result, each communicating port has a weight of 10 points, each alert and intrusion is worth 1000 index points, network and self changes were used as is, and I/O ratio is also multiplied by 10 in order to get the desired base index. However, the weight of 10 for I/O ratio generated a large number of false alarms. For example, a legitimate host can be downloading files from an FTP server, but the host will accumulate a high packets or bits I/O ratio because it is receiving much more data than it is sending. Therefore, additional weighting for I/O ratio is not used. Specifically, flow I/O ratio is used to identify host/port scans and DOS/DDOS attackers and victims, and the results are categorized into intrusions with the corresponding index values (see section 3.7).

3.8.2.1 Anomaly Index for Communicating Ports

By looking at the raw data and analysis from RUM, most hosts that appear to starting having relatively significant problems have approx. 100 or more communicating ports. Therefore, in the implementation, when examining the host database, each unregistered port that a host communicates on contributes 10 index values to the category, so when there are 100 communicating ports it reaches the base index value of 1000.

The following formula shows the calculating of the basic ports index mathematically, where N is the number unregistered ports and X is the index value that each port contributes to the total (10 in this implementation).

$$Index = N \times X$$

Equation 3.4 – Ports Index Generated From Unregistered Communicating Ports

Additional conditions are considered when the total number of ports reaches a multiple of 500 (i.e. 500, 1000, 1500, 2000, etc.) because at each of those points, the significance of the host's problem reaches a new level. Therefore, at each point, an alert is logged with a corresponding intrusion index value. For example, a host with 363 unregistered communicating ports has a ports index of 3630, whereas a host with 1021 unregistered ports has a ports index of 10210 and an alerts index of 2000 as well (1021 is two times over the 500 level).

This is shown by the equation below, where C is the alerts index accumulated when additional level is reached (1000 is our case), N is again the number of unregistered ports, and V is the threshold of the additional level (500 ports in our case).

$$Index_a = C \times N / V$$

Equation 3.5 – Alerts Index Generated From Communicating Ports

Another analysis is the Trojan/virus port scan. In this case, all the communicating ports are scanned against a database of predefined and publicly available common Trojan/virus ports. If a matching port is identified, the system raised another alert with 1000 in alert index. In addition, to distinguish and signify the number of possible Trojan/virus ports so that hosts with multiple matches may get more attention from the administrator, an additional index of 10 is added to the alerts index for each matching port.

$$Index_a = C + N \times X$$

Equation 3.6 – Alerts Index Generated From Trojan/Virus Ports

3.8.2.2 Anomaly Index for Alerts & Intrusions

An alert or intrusion is an anomalous event generated as a result of analyzing the traffic data and its parameters. Each alert or intrusion carries an index value of at least 1000 because every alert or intrusion signifies something wrong with a particular host and deserves some attention. The total index for alerts is simply the sum of index from all individual alerts (Equation 3.7), and same for intrusions (Equation 3.8).

$$Index_A = \sum Index_a$$

Equation 3.7 – Total Alerts Index

Lists of Alerts:

- Host does not belong to any subnet (index: 1000)
- Host role not in database (index: 1000)
- Communicating on large number of ports (index: Equation 3.5):
- Possible Trojan or Virus Detected (index: Equation 3.6):
- Traffic threshold exceeded (explained here):

  For every 100% (rounded) of the threshold that the actual traffic parameter exceeds, 1000 alerts index is generated. For example, a client host with total flows that equals to 2% (rounded) of total of traffic flows in the sample gets a flows threshold exceeded alert with 2000 index points.

$$Index_I = \sum Index_i$$

Equation 3.8 – Total Intrusions Index

Lists of Intrusions:

- Host/port scan attacker or DOS/DDOS attacker:

  Index equals 1000 when the host is identified as an attacker with flow ratio of < 0.5, 2000 when < 0.1, and 3000 when < 0.01 due to the increasing likelihood of being a definite attacker. (Please refer to section 3.7 for complete details on algorithms and section 4.3.5 for implementation details.)

- Host/port scan victim or DOS/DDOS victim:

  Three times the indices of the attackers because victims deserve more immediate attention. Index equals to 3000 when host is identified as a victim with flow ratio > 2, 6000 when > 20, and 9000 when > 100 due to the increasing likelihood of being a definite victim. (Please refer to section 3.7 for complete details on algorithms and section 4.3.5 for implementation details.)

To summarize, there are three possible types of computer attacks [23]: 1) attacks that deny someone else access to some services or resources a system provides; 2) attacks that allow an intruder to operate on a system with unauthorized privileges; and 3) attempts to probe a system to find potential weaknesses. The Dynamic IDS addresses all three types in certain degree. For the first one, the system focuses on detection of attackers and victims of Denial of Service (DoS) and

Distributed DoS (DDoS)[9]. For the second type, the system tries to identify Trojan and virus on the network that may lead to unauthorized intruder access. Finally, the system detects host/port scan in a general form to identify probes.

3.8.2.3 Anomaly Index for Network-relative and Self-relative Changes

These two index categories contain the values generated by using a set of statistical measures described in section 3.6. One set of metrics is computed with respect to host parameter changes against the network average from the same sample period – network-relative. The other set is based on comparison of changes in host behavior in the current sampling period with respect to and its behavior in the last sample sampling period. For example, number of flows, open ports, etc. are some of the parameters under scrutiny.

The assumption is the majority of the hosts in a large network behave "normally" at any given instance of time. This assumption is true, based on NC State observations, most of the time. It may not apply in a specific subnet, but in general it appears that [31] not more than about 5-10% of the network clients active at any one sample exhibit undesirable behavior. Hence comparison of statistical behavior of a host with respect to that of the ensemble has a good chance of indicating anomalous behavior of the client. Of course one has to be careful with statistics like this and further work is needed.

For each of the parameters used in the host profiles (section 3.6), the difference in percentage between the host and the network is calculated (Equation

[9] See section 3.7 for complete details on algorithms and section 4.3.5 for implementation details.

3.9). These differences are in turn taken directly as is, or applied with a weight, and summed to get the total index from network comparison (Equation 3.10).

$$\%Difference = \left| \frac{V_{host} - V_{network}}{V_{network}} \right|$$

Equation 3.9 – Host-to-Network Comparison

$$Index_{Network} = \sum_{all\ V} a_V \cdot \left| \frac{V_{host} - V_{network}}{V_{network}} \right|$$

Equation 3.10 – Total Index from Network Comparison

In addition, network average varies depends on the network a particular host resides on, the host's role, and traffic direction. For the NC State environment, the different networks include Main Campus, Centennial Campus, Dorms, Illegal, Multicast, Unclassified, and Combine (used when a host is not found to be in any one of the other predefined networks). The actual network a host belongs to is used so that the network average data can be more closely related to the host, and the NC State network is large enough (usually has around 10,000 hosts in each sample) to be categorized further and does not break the base assumption. The roles include client and server, so that the clients and servers are compared to a different set of unbiased average, since their behavior significantly different. Last, traffic directions obviously are inbound and outbound (or hosts as destinations and hosts as sources).

In the case of self-relative analysis, assumption is somewhat different. This analysis looks at the rate of change in the metrics of interest (in some cases this is an estimate of the equivalent to first differential of the metric). Comparison happens between the values in the "present" and those collected in the immediately "preceding" period (Equation 3.11). Assumption is that under normal circumstances, a particular host's behavior should be relatively consistent to itself from one sampling period to another, and that sudden changes in its behavioral parameters may be an indicator of anomalous behavior. Similar to total network index, the total index from self comparison is the results of all differences directly or with a weight (Equation 3.12).

$$\%Difference = \left| \frac{V_{current} - V_{previous}}{V_{previous}} \right|$$

Equation 3.11 – Host-to-Self Comparison

$$Index_{Self} = \sum_{all\ V} a_V \cdot \left| \frac{V_{current} - V_{previous}}{V_{previous}} \right|$$

Equation 3.12 – Total Index from Self Comparison

In the current system, the index values of both network- and self- relative analysis are often taken directly from the calculating of the differences in percentages of the parameters. Since percentages are relative metrics already, they can produce

biased results[10]. Therefore, to minimize some of these biases, if a single parameter results in an anomaly index of over 1000 then the value for that parameter value is considered biased because it in fact is proven during the experiments to greatly and wrongly affects the total index and creates false alarms. In the current implementation, the system replaces such values with -1. Negative one is used to indicate to the user that this particular parameter may be biased and has been taken out of the consideration by the system. An experiment of hosts with greater than 1000 individual parameters was performed on multiple traffic samples. The results indicate that many of such hosts had high index for other parameters as well (that did not exceed 1000). Therefore, the system can count on other parameters and analysis to identify problems. However, this process is still not entirely safe because it is also likely for other parameters to have over 1000 in index. In that case the system would possibly fail to identify the anomalies. This is one of the limitations that the current system has. To overcome this limitation, correlation among the parameters need to be studied.

3.8.2.4 Anomaly Index for I/O Ratio

The index of I/O ratio is also taken directly from the calculation of the sum of the actual ratios, or the inverse of the ratio if it is less than 1 (see Equation 3.3) and all three ratios are considered equally in this category.

[10] Index based on differences in the actual parameters with the appropriate thresholds may produce more accurate results than based on percentages. This requires further study and investigation and is described in future works.

$$Index_{Ratio} = \sum_{flows,packets,bits} \begin{cases} input/output & if \quad input/output > 1 \\ \dfrac{1}{input/output} & if \quad input/output < 1 \\ 0 & if \quad input/output = 1 \end{cases}$$

Equation 3.13 – Total Index from I/O Ratio

Moreover, flow ratio is used specifically to determine host/port scan and DOS/DDOS attackers and victims (section 3.7) and intrusion warnings are generated with corresponding index (section 3.8.2.2 and section 4.3.5).

3.8.2.5 Total Index

Index from any single one of the above six categories will not necessarily put a host on the top of the IDS's list, unless it is really extreme. It is the combination of the six that count. The total index is simply the sum of all the six categories above. To represent this model in a function, as follow:

$$Index = a_1 I_1 + a_2 I_2 + \cdots + a_n I_n$$

Equation 3.14 – Total Anomaly Index

Where *I* is the index value of each individual component of the system, and *a* is the weight (i.e. importance) of the corresponding component. Currently, the system considers all of its six index categories equally important so they are weighted equally – all *a*s equal to 1. In general, the model does not have to be additive and is likely to be weighted. The rational is important though. It attempts to pool information about a host or a flow suite that otherwise may go unnoticed ("in the noise"), and that as a collective set of information may be indicative of "unusual"

events and thus may indicate host(s) that may require additional scrutiny. It is important to note, however, that this index is just a "flag" tool that does require active engagement of a human operator who can, and is expected to, perform inductive reasoning about the information and thus act (among other things) as an additional knowledge element in the end-host database, and as a false-alarm filter.

Chapter 4 – Implementation

4.1 Design

High-level architecture of RUM intrusion detection module is depicted in Figure 4.1.

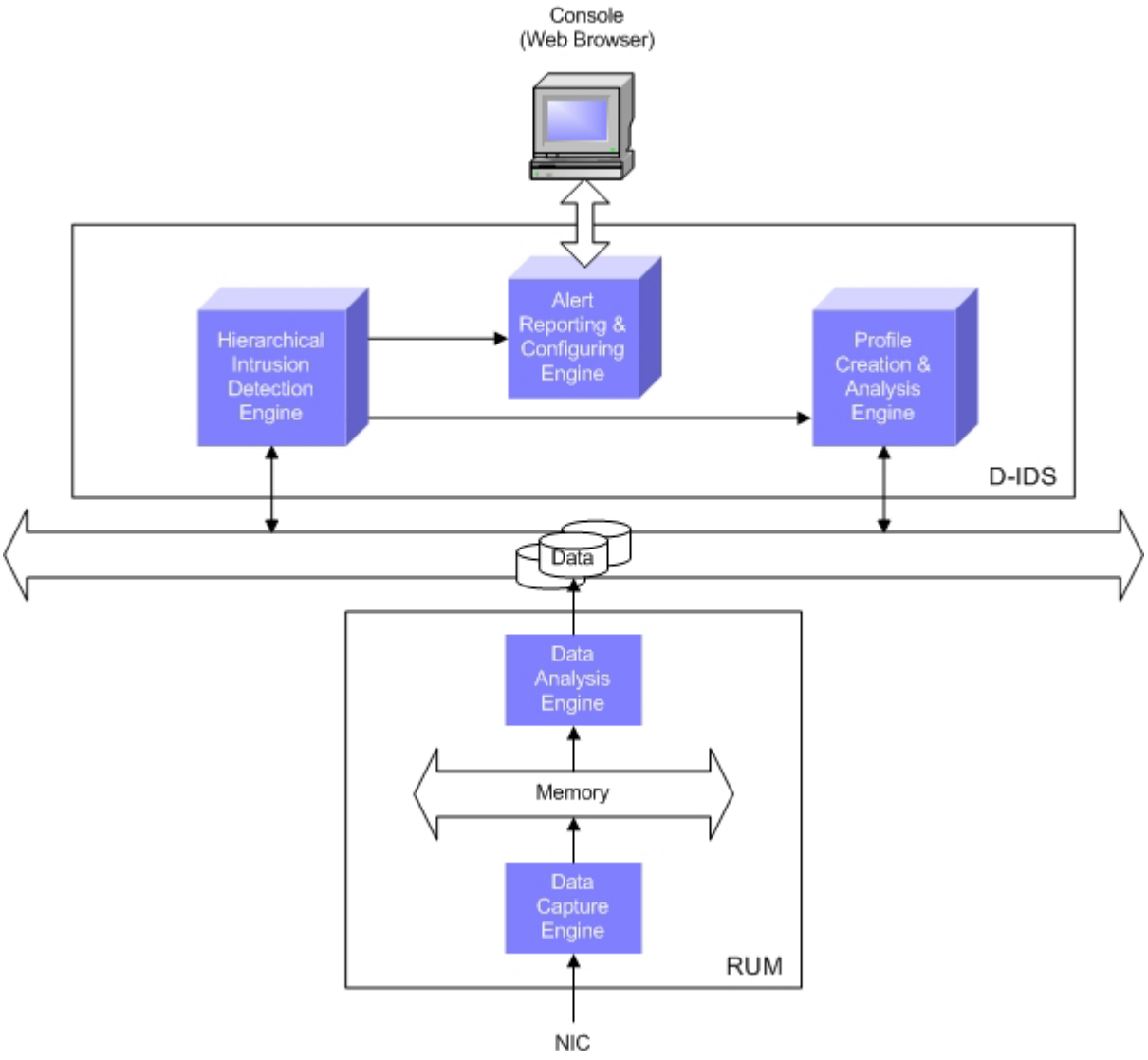


Figure 4.1 – Architecture

The system uses RUM to capture network traffic from the Network Interface Card (NIC) and stores it first in memory. Traffic data is then filtered, sorted, and classified in a way that the IDS will understand, and is then stored on disks. The IDS uses the

sorted information, generates dynamic profiles and performance information, and effects further analyses using a hierarchical intrusion detection engine. This detection engine is first deductive, and then, in combination with the human operator, inductive. It includes the five major components introduced in Chapter 3 – host database, dynamic policy, community-change and self-change statistical analyses, input/output ratio analysis, and host status as one additional component. Index values are generated during each of the six steps and are summed based on a linear additive model. It is worth re-iterating that the additive model is an experimental one, and it is intended to capture information that may otherwise be "in the noise.". The results are displayed to a web-based user interface through a reporting and configuring engine.

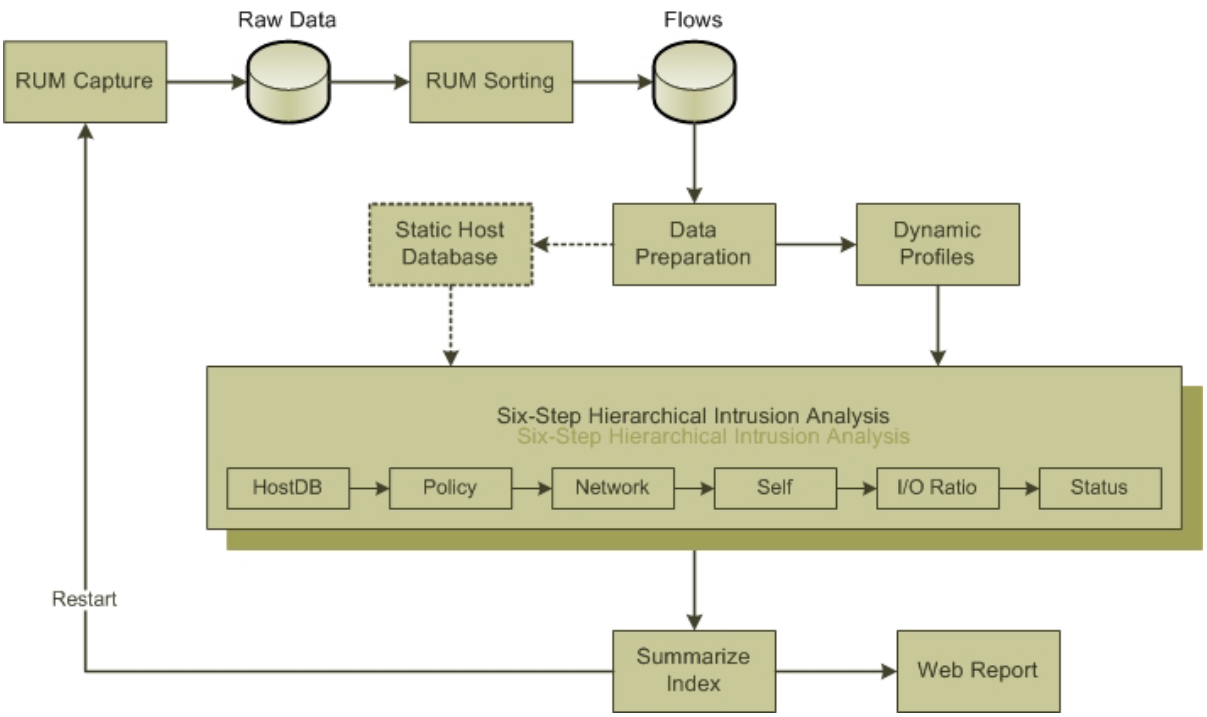


Figure 4.2 – Dynamic IDS Architecture Diagram

4.2 Data Preparation

There are two major things that need to be done before an actual analysis can take place. First, the system needs to capture network data, and second, an end-host profile database needs to be created.

There already are many open-source tools that can be used to capture network traffic in real time. One example is Tcpdump that uses the libpcap packet capture library. But capturing the data is far from enough. The data needs to be classified and stored in a format useful for further analyses. This is especially important if real-time analysis is desired since the data capture and analysis retrieval overhead needs to be minimized. Speed is one of the reasons why RUM [31] does not use Tcpdump directly, but the RUM author has written a RUM specific capture code (also based on libpcap) that provided a considerable speed-up. Because current work extends RUM capabilities this work also uses the data capturing engine. RUM captures the network traffic and stores it in a flow format, where flow is defined as "a unique IP protocol, source IP, source port, destination IP, and destination port combination." RUM implements Perl Storable [49] to store all data in binary format, including network traffic and related information. The current version of IDS focuses on the flow information that RUM creates, and does most of the analyses based on the flow data.

The end-host (or the profiling) database contains the static information for each host – the services that are allowed to run on the host and the roles that the host plays in the network (server or client or both) – and the dynamic profile of the host. However, this is only part of the story. One has to remember that the basic idea of the current

approach is that it is not possible to fully quantitatively capture neither the operational profile of a host, nor the inductive algorithms and experiences a good network security administrator uses to detect intrusions. Hence, the intention of the end-host database is to capture as much information as possible, but that in operation that information is seamlessly augmented by profiling information available in the mind (knowledge) of a security administrator. Similarly, analyses provided by the Dynamic IDS module are intended not to augment the deductive capabilities of packages such as SNORT, experiences used and cognitive induction processes expected from by a good security administrator. In other words, the IDS is intended to be a "security assistant" not a fully automated system. In the current implementation, the end-host database is stored as Perl Storable binary files.

4.2.1 Database – Static Information

Static information of the host database is simply the registered ports of each host (i.e. legal services that the host can run) and the role of each host (i.e. client, server, or both). This information is only gathered once on the first run. Since it is not dynamic, the system will not attempt to update the information on consecutive executions. However, the administrator can manually tune the information to be more accurate.

4.2.2 Database – Dynamic Information

Dynamic information of the host database is in fact the collection of dynamic profiles of all hosts. To generate this dynamic profile, the system first scans RUM data to find out a list of all the active hosts for the current sample as it starts up, and gathers their corresponding information from the host database. For each of these hosts, a few

operations are performed. First, host traffic data are extracted and concatenated from flow format into host format. In other words, the system searches for pieces of information and puts them together for each host. The results look like the following:

```
$VAR1 = {
          'Network1' => {
                          'IP1' => {
                                     'S' => {
                                              'packets' => [
                                                             1
                                                           ],
                                              'bits' => [
                                                          480
                                                        ],
                                              'proto' => [
                                                           '6'
                                                         ],
                                              'ports' => [
                                                           '4462'
                                                         ]
                                            },
                                     'D' => {
                                              'packets' => [
                                                             1
                                                           ],
                                              'bits' => [
                                                          480
                                                        ],
                                              'proto' => [
                                                           '6'
                                                         ],
                                              'ports' => [
                                                           '4462'
                                                         ]
                                            },
                                   },
                          'IP2' => {
                                     ... ...
                                   },
                          ... ...
                          },
           'Network2' => {
                           ... ...
                          },
           ... ...
         }
```

Figure 4.3 – Traffic Database Format

This information is then used in the next operation to perform statistical calculations for each host. Specifically, the following (Table 4.1) is calculated for each host:

Num	*Variables*	*Description*
1.	flows	Total number of flows in this sample
2.	packets	Total number of packets in this sample
3.	avg_ppf	Average number of packets per flow
4.	min_ppf	Minimum number of packets per flow
5.	med_ppf	Medium number of packets per flow
6.	max_ppf	Maximum number of packets per flow
7.	range_ppf	Range for number of packets per flow
8.	var_ppf	Variance for number of packets per flow
9.	std_ppf	Standard Deviation for number of packets per flow
10.	skew_ppf	Skew for number of packets per flow
11.	kurt_ppf	Kurt for number of packets per flow
12.	bits	Total number of bits in this sample
13.	avg_bpp	Average number of bits per packet
14.	avg_bpf	Average number of bits per flow
15.	min_bpf	Minimum number of bits per flow
16.	med_bpf	Medium number of bits per flow
17.	max_bpf	Maximum number of bits per flow
18.	range_bpf	Range for number of bits per flow
19.	var_bpf	Variance for number of bits per flow
20.	std_bpf	Standard Deviation for number of bits per flow
21.	skew_bpf	Skew for number of bits per flow
22.	kurt_bpf	Kurt for number of bits per flow
23.	ports	Total number of communicating ports in this sample
24.	pro_tcp	Percentage of TCP protocol packets in this sample
25.	pro_udp	Percentage of UDP protocol packets in this sample
26.	pro_icmp	Percentage of ICMP protocol packets in this sample
27.	pro_other	Percentage of other protocol packets in this sample

Table 4.1 – Host Profile Variables

The above values serve as the profile for each corresponding host for the current sample. The profile is a preparation for further analysis, and for creation of the host index. In addition to individual host profiles, network (or community) profiles are generated for Main Campus, Centennial Campus, Dorms, and the Combined Overall Network. A finer granularity of "communities," e.g., at the subnet level, is possible. To do so, the system checks which network a host belongs to while it scans and generate statistical profile for individual hosts, and at the end of the scan, an average

value for each of the above variables is calculated based on the data of hosts that belong to the same network. The Combined Overall covers data from all available hosts, regardless of their network property. The networks are further divided into subcategories, categorized by roles and traffic directions – server/client and source/destination.

Architecture is illustrated in Figure 4.2.

4.3 Analysis Algorithms

RUM runs in a statistical mode, which means it samples the network every certain amount of time (typically as sample is 60 seconds long, and samples may be taken every 5 minutes). For each interval, the intrusion detection system is started right after RUM finishes (in the prototype discussed in this thesis, due to processing time needed for IDS operations – i.e., hardware limitations, samples are about 30 minutes apart).

After the first stage of processing, the system is ready to perform the second and final analysis. For each of the active hosts during the current interval, the system performs a six-step analysis to ensure the thoroughness and accuracy. During each step, an index number is generated for the host. At the end all indices are pooled in the final model (right now an additive equal-weight one). The bigger the host index number, the more "unusual" host behavior appears to be .

The six steps are:

- End-host database analysis
- Thresholds measuring
- Network Statistical Analysis
- Self Statistical Analysis
- Input/Output Ratio
- Network Status Determination

Figure 4.4 – Six-Step Hierarchical Intrusion Detection

During the analyses, index values are generated into one of multiple of the following six categories, as described in section 3.8. The final index value is the sum of these six individual values.

- Index{ports} – generated for each unregistered communicating port
- Index{alerts} – generated when an alert is raised, such as virus or host down
- Index{intrusions} – generated when possible intrusion identified (e.g. DOS)
- Index{network} – generated from network (or community) statistical analysis
- Index{self} – generated from self statistical analysis
- Index{ratio} – generated from input/output ratio

4.3.1 End-host Profiling Database

End-host database analysis is the first step that the system takes towards a thorough analysis. It serves to determin some obvious problems and it provides enough knowledge about the host to help perform analyses that follow. The system first tries to determine the subnet of the host using its IP address and subnet mask, and then the network it resides on by looking this up in a subnet-to-network mapping databased. The information is used in all further analyses as the host's basic property. If the host is not found in any subnet or network, an alert will be generated and it contributes 10,000 points to the host index. The system then tries to find the host in the traffic data analyses done previously, retrieves the communicating ports used during the sample interval, compares those to information stored in the host database, and determines if the host is communicating on a port that is not specified in the host database.

For every port that the host communicates on, and that is not in the host database, the system generates an index of magnitude 10. If the host communicates on more than 30 unique ports in a single sample interval, the system alerts "*Host running large number of services*" ands adds 300 additional index points. In the case of 1000 or more ports, it alerts "Host *running extremely large number of services*" with index of 1000.

The ports are also compared against a list of Trojan/virus ports to determine possible port activity causes [40]. When at least one possible Trojan port is found, the system generates a "*Possible Trojan or Virus Detected*" alert, and adds another 1000 points to the index. In addition, each possible Trojan port contributes 100 points to the total index itself, so the more ports there are, the higher index value will be. The role information of the host is also retrieved,. The last index increment in this step, for 500 points, is added if host has no defined role in the host database. Another alert is also raised to warn the administrator, so he/she can investigate this and perhaps define a new entry for it in the database.

The pseudo code for this step is shown below:

```
For each host in the network
      Get registered ports and host role
      If host role not defined
            Raise "undefined role" alert
            Increase alert index by 500
            Increase total index by 500
      End If

      Get communicating ports in current sample
      For each unregistered port
            Increase ports index by 10
```

```
            Increase total index by 10
      End For

      If possible virus/Trojan port found
            Raise "virus/Trojan" alert
            Increase alert index by 1000 + 100 x number of virus ports
            Increase total index by 1000 + 100 x number of virus ports
      End If

      If number of unregistered ports > 1000
            Raise "extremely large number of communicating ports" alert
            Increase alert index by 1000
            Increase total index by 1000
      Else If number of unregistered ports > 300
            Increase alert index by 300
            Increase total index by 300
      End If
End For
```

4.3.2 Thresholds/Policy Enforcement

Threshold measuring is a used to determine the most obvious, first sign, of possible problems. In most systems, a threshold is usually a fixed pre-set value that defines a specific limit. However, a fixed value that is set for high traffic occasions may cause many false alarms during low traffic hours, and a value that is good for low traffic hours will certainly be too small when the traffic comes up. A trade-off points is often selected to set the value somewhere in between the two, so that it will not cause too many false alarms and will not loses many true intrusions. But that is not optimal. Therefore, in a dynamic intrusion detection system, this threshold should also be dynamic. A total of three thresholds are defined – number of flows, packets, and bits. Each of the three is defined as a percentage of the total current network (community) value, instead of a fixed value. And the values differ for server hosts and client hosts, because in general, it is more likely for network servers to consume more bandwidth and have higher values than clients. By default, the thresholds are set to 1% of the corresponding network (community) data for client hosts, and 3% for server hosts.

For example, if the total number of flows for the entire network is 100,000, the flow threshold is 1,000 for clients and 3,000 for servers. In such cases, a "*Traffic flows exceed threshold*" alert is raised, and 1000 points added to the index. Please note that in this example threshold is (in our experience) very generous since in most cases hosts that exceed about 300 flows (in a 60 second sample) need scrutiny unless they are registered servers.

The details are as follow:

```
For each host in the network
      For traffic flows, packets, and bits
            If host is client only
                  Threshold = 1% of total network
            Else If host is server or both
                  Threshold = 1% of total network
            End If

            If host Actual > Threshold
                  %Over = Actual / Threshold
                  Raise "threshold exceeded" alert
                  Increase alert index by 1000 * %Over
                  Increase total index by 1000 * %Over
            End If
      End For
End For
```

4.3.3 Network Statistical Analysis

Network (community) statistical analysis is done by comparing the host's statistical profile against the average network profile for the network which the host resides in. This operation determines how differently this particular host behaves to the network as a whole. If the host is under attack, or engaged in an attack, or has a Trojan or virus, it is believed that it should behave significantly differently than a normal host. The system also assumes that at any give point of time, the majority of a network should be operating normally (an interesting assumption that may not hold in environments

where a large number of hosts has been compromised). Based on the difference, an index value is calculated and added to the host's index total.

```
For each host in the network
      For each statistical parameter
            Index = host data – network data / network data
            Increase network index and total index
      End For
End For
```

If the host is not found in any one of the three major networks (Main Campus, Centennial, and Dorms), its data is compared against that of the entire network consisting all hosts from all three networks.

4.3.4 Self Statistical Analysis

Self statistical analysis, similar to the network statistical analysis, is done by comparing the host's current statistical profile against its own profile from the last sample, to determine if there is a significant change in its own behavior. The function that calculates the difference in percentage and result is considered an index for the host.

```
For each host in the network
      For each statistical parameter
            Index = host current data / host last data
            Increase network index and total index
      End For
End For
```

4.3.5 Input/Output Ratio

Input/Output Ratio is defined for flows, packets, and bits, as follow:

```
Flow I/O Ratio = Incoming Flows / Outgoing Flows

Packet I/O Ratio = Incoming Packets / Outgoing Packets

Bit I/O Ratio = Incoming Bits / Outgoing Bits
```

All of these three ratio value will be used as index value accumulated directly to Index{ratio} and total index.

Flow I/O Ratio is used to determine DOS (Denial of Service), DDOS (Distributed Denial of Service) attacks and port/host scans. Although Denial of Service attacks come in a number of varieties [4][41], they have one similar characteristic. That is, the attacker generates extremely large number of request to a particular server, making it impossible to provide service to legitimate users. Ideally, in a given period of time, a host communicates using TCP protocol should have identical number of incoming flows and outgoing flows, because TCP is a synchronized protocol [47]. Host/port scans have similar characteristics. A scanning host transmits a large amount of small packets to all ports on a particular target host or to multiple hosts targeting a particular port. To further distinguish between DOS/DDOS attacks and host/port scans, the number of unique communicating hosts is used. The following implementation is used, which is based on the algorithm discussed in section 3.6.

```
For each host in the network
      If Flow I/O Ratio >> 1
            If Many Unique Communicating Hosts
                  If Many Unique Communicating Ports
                        ➔ Possible Distributed Scan/DOS Victim
                  Else
                        ➔ Possible DDOS Victim
                  End If
            Else
                  If Many Unique Communicating Ports
                        ➔ Possible Port Scan Victim
                  Else
                        ➔ Possible DOS Victim
                  End If
            End If
      Else If Flow I/O Ratio << 1 AND > 0
            If Many Unique Communicating Hosts
                  If Many Unique Communicating Ports
                        ➔ Possible Host/Port Scan Attacker
```

```
                  Else
                        ➔ Possible Host Scan Attacker
                  End If
            Else
                  If Many Unique Communicating Ports
                        ➔ Possible Port Scan Attacker
                  Else
                        ➔ Possible DOS/DDOS Attacker
                  End If
            End If
      End If
End For
```

In both the victim case and the attacker case, the probability gets higher if the flow I/O ratio is further different than the optimal value – one. Therefore, index values are generated differently according to the different probability levels. The table below exhibits the level of severity based on flow I/O ratio and the corresponding index value generated.

Flow I/O Ratio	System Alert	Index Number
< 0.5 and > 0	Possible Attacker	1000
< 0.1 and > 0	Possible Attacker	2000
< 0.01 and > 0	Attacker	3000
> 2	Possible Victim	3000
> 10	Possible Victim	6000
> 100	Victim	9000

Table 4.2 – Flow I/O Related Index Values

As shown in the above table, 1 is the prefect number that shows the host is behaving perfectly. For larger than 1, greater ratio represents a higher chance being an attack or scan victim, because it means there is more incoming traffic than outgoing, thus a higher index number. In contrast, a smaller ratio that is less than 1 but greater than 0 represents a higher probability for an attacker. From a network administrator's perspective, a host under attack deserves more immediate attention than a host performing an attack. Therefore, the index numbers for victims are three times higher than that of the attacker.

4.3.6 Host Status Determination

Host status determination is the last analysis that the system does, where it tries to find the host in both the current and the last traffic data files, in order to determine a host's up/down status. There is no change in its network status if the system sees the host in both files, which means the host was up and still is. However, if the system only sees the host in the last sample, it knows that the host went down between now and then. Oppositely, if it only sees the host in current sample, the host's status changed from down to up. Corresponding, the alerts and index points generated are shown in the following table.

	Server		**Client**		**Unknown Host**	
	Alert	***Index***	***Alert***	***Index***	***Alert***	***Index***
Up	"Server Up"	2,000	"Client Up"	1,000	"Unknown Host Up"	3,000
Down	"Server Down"	3,000*	"Client Down"	1,000*	"Unknown Host Down"	2,000*
* *Plus index from last sample*						

Table 4.3 – Host Status Determination

Notice that in the above table, there are three different types of hosts defined, and their index value differs from each other for the same up/down event. The reason is that they have importance. For a freshly started host, a host with unknown status is potentially more dangerous than the other two types. Because we have very limited knowledge of the unknown host, it has the capability to do more harm than the known hosts. And then a faulty server can be more serious than a faulty client. For a host that goes down, it probably requires immediate attention and investigation if the host is a server. A host with unknown status should deserve some attention too. Again, a common client goes up and down regularly and thus is least important. In addition to the fixed index values, all of the down hosts accumulate their previous indexes onto the current one. By including the index value from the previous sample instead of

simply assigning a static value, the IDS system clearly distinguish the level of importance among all the different hosts that went down in the current sample period, and the placement of each individual down host became dynamic due to previous behavior. A more suspicious host that went down deserves more attention from the system administrator.

4.3.7 Host Indexing

Finally, after all the analysis have taken place and index generated (based on the formulas show in section 3.7.2), the system analysis engine stores all the alerts, intrusions, and index values to binary database files in Perl Storable format. The data is now ready to use for reporting, monitoring, or further human analysis, which will be explained in greater detail in the Web Interface Section of the same chapter.

4.3.8 Other Features

Other features of the system include optional port scanning for a particular host and integration with RUM. In particular, port scanning is done using Nmap, a powerful open-source tool, as the scanning engine. The scanning can be invoked by system administrator from the web monitoring interface. The details will be addressed in section 4.5.

4.4 Tuning the System

The Dynamic Intrusion Detection System does not require any manual configuration in order to work. These roles and services are defined and end-host database is built when Dynamic IDS is first initialized. During the initialization process, the system

analyzes each host available in that particular sample and automatically defines every open port ranging from 1 to 1023 as a legitimate service. Hosts with at least one legitimate service was defined as a server, otherwise a host. The automation is the entirely accurate. Since the end-host database is created during the initial automatic system set up against some sample data, it is very likely that the end-host database may contain inaccurate data. For example, any host that is suppose to be a client can be mistakenly logged as a server if a "legitimate" server is running at the time of initial setup and vice versa. Some simple tuning of the system can be done to obtain more accurate results. The inaccurate information can be easily correctly by configuring the host database from the system configuration web interface (see section 4.5.2). Once all the hosts are correctly logged in the database, the system can perform analysis more accurately.

4.5 Web Interface

Data displaying and results reporting can be as important as the system analysis itself, since the information is only useful when presented correctly and efficiently to the users. Dynamic intrusion detection system uses web interface for monitoring and configuration, for the same reason that RUM uses web interface – that it is platform-independent; it offers conveniences such as access from any location and no specialized client-side software required [38]. The section will detail the system's monitoring and configuration interfaces, as well as the integration with RUM.

4.5.1 Monitoring Interface

The monitoring interface is made up with three main sections. They are Monitoring Home, Main Console, and Host Details. The Console and Host Details page can be invoked from Monitoring Home.

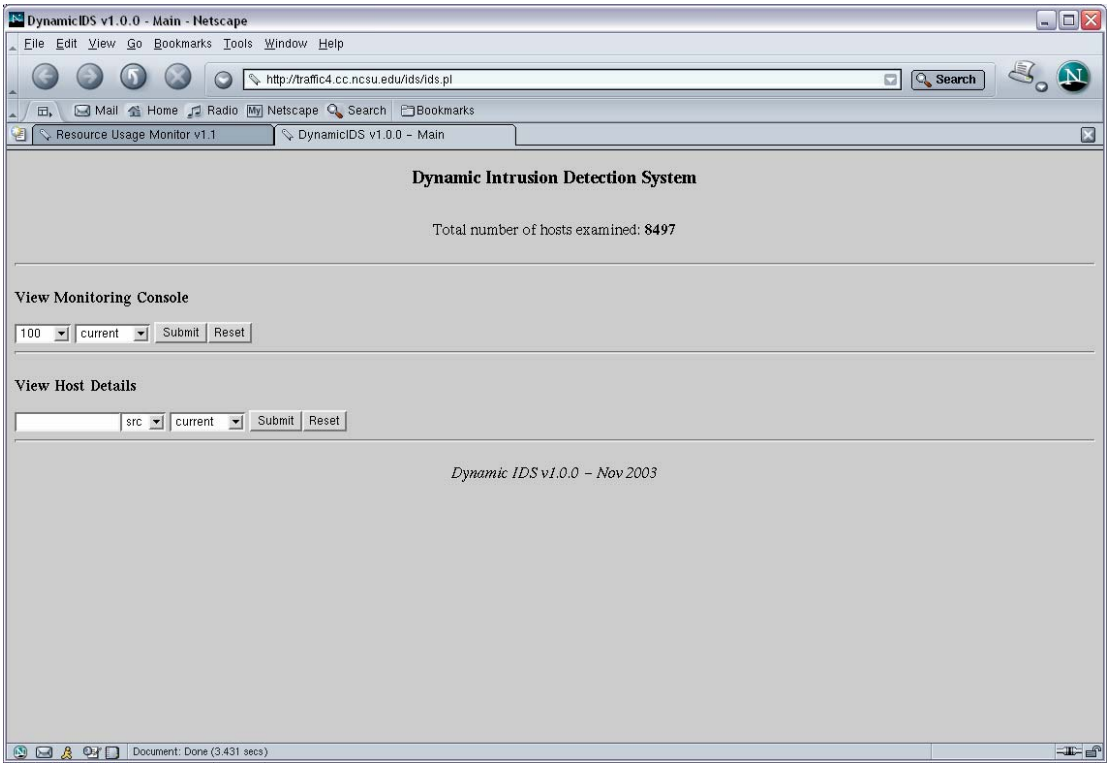


Figure 4.5 – Dynamic IDS Monitoring Home Screenshot

The Main Monitoring Console page displays the selected number of hosts in a table format. For each host, brief information such as Index, Status, Role, Services, Network, and Subnet are displayed in the columns. The hosts are sorted based on their index value, with the highest ones on top. This way, the hosts that are considered more serious and problematic attracts more attention from the administrator. It conveniently provides the administrator an investigation list.

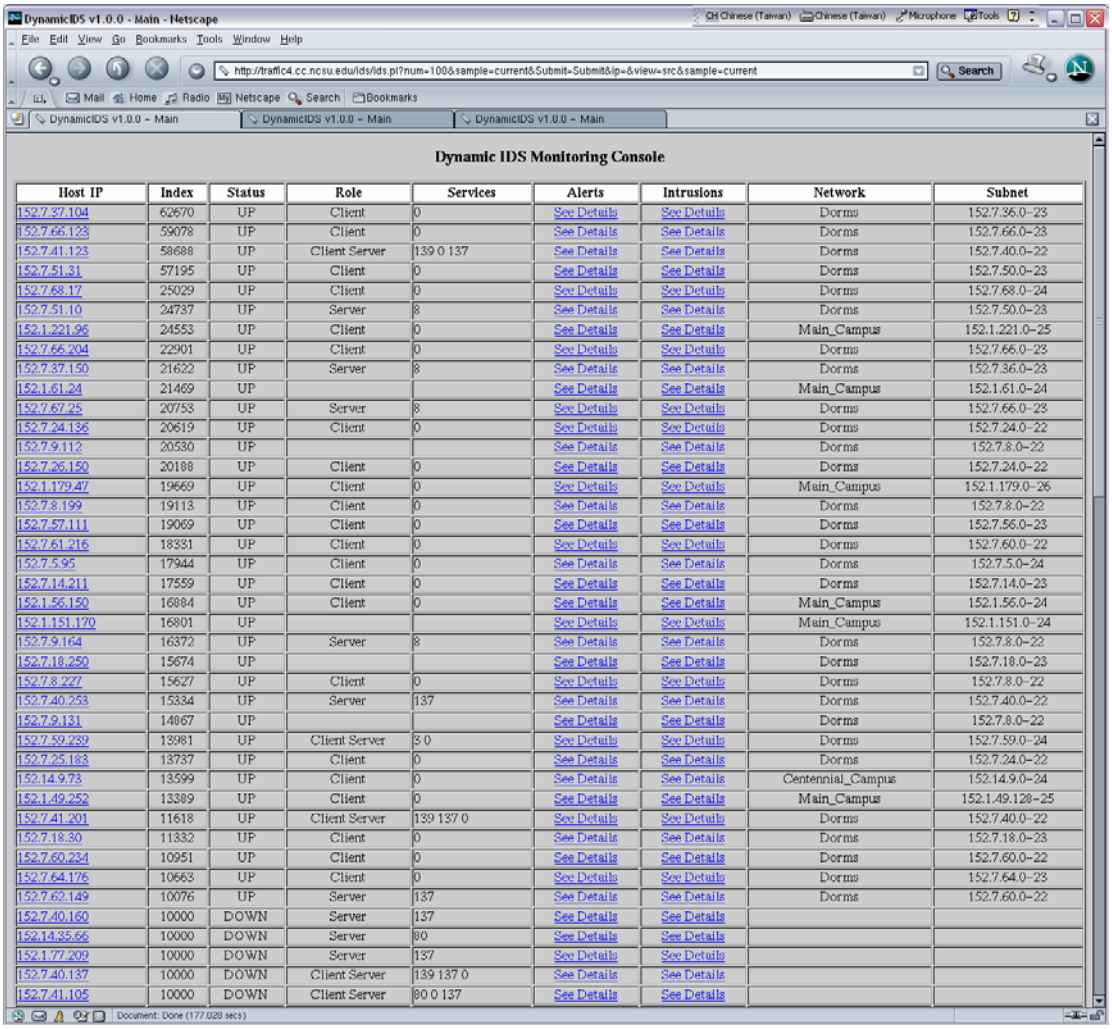


Figure 4.6 – Dynamic IDS Monitoring Console Screenshot

The Host Details page can be reached by entering its IP address from the Monitoring Home page, or by clicking on the link from the Console Table as shown above. The details page contains all the information that the administrator needs to know about this host. It first displays the total index value, as well as the index values generated corresponding to different categories – Ports, Alerts, Intrusions, Network Analysis, Self Analysis, and Input/Output Ratio. Then lists of communicating ports, alerts, and intrusions are shown. If any of the ports matches the ones defined in the Trojan Ports database, that particular port will have a red link, so the administrator can simply

click on it to bring up a description for that Trojan port. Finally, the host's status, role, allowed services, network, subnet, and input/output ratios are displayed in the middle of the page. On its left the page shows details for network analysis, whereas details for self analysis are shown on the right. In addition to the host information, the host also presents several different viewing options and administrative options. The viewing options include View Host as Source or Destination, and View Current or Previous Sample. The administrative options include modifying the host's information in end-host database, and actively probing the host to find open ports. The modifying option is explained in detail in the configuration section below. Probing of the host is done by implementing Nmap [34] in the execution, and presenting Nmap output onto the browser.

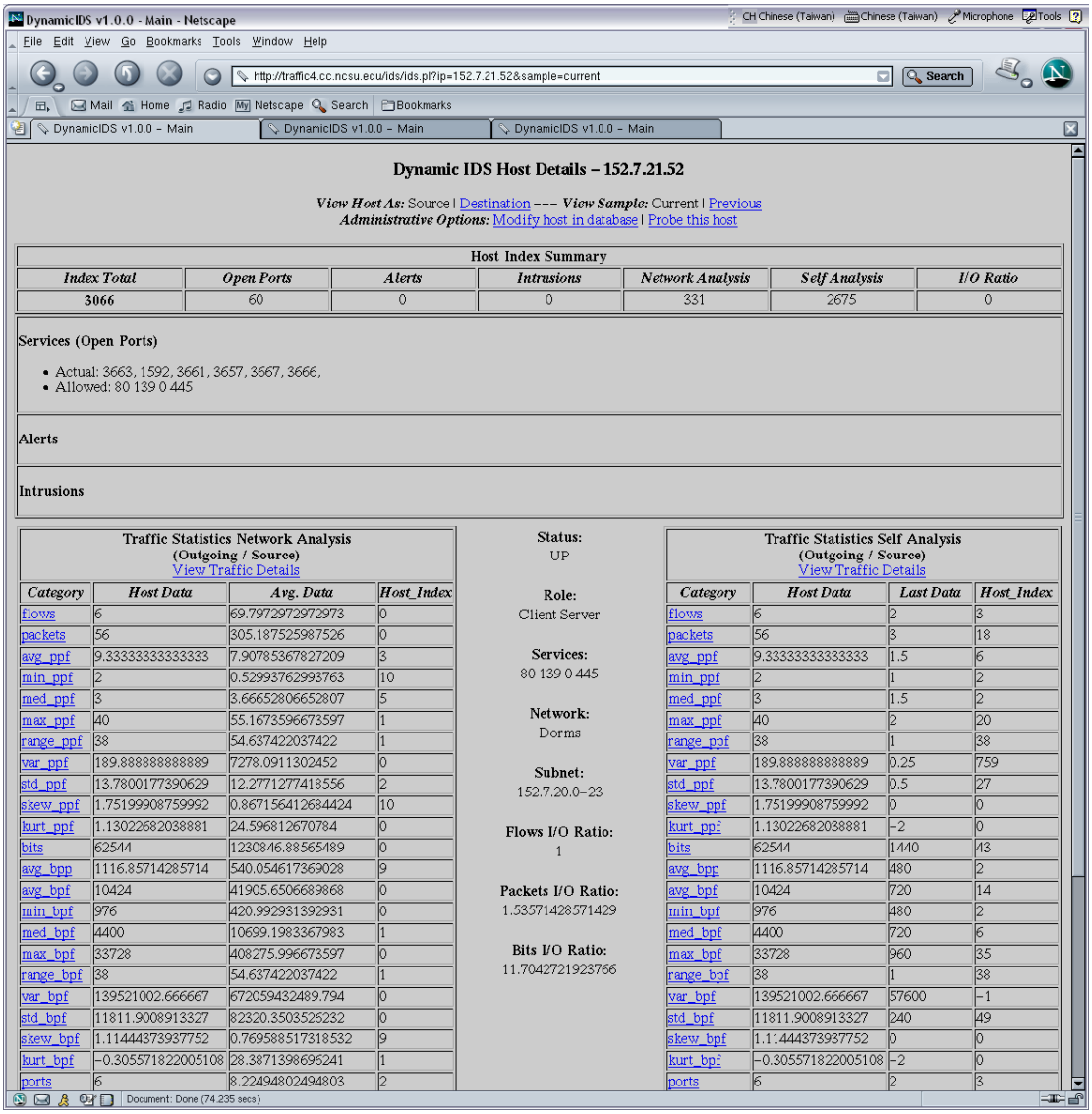


Figure 4.7 – Dynamic IDS Host Details Screenshot

4.5.2 Configuration Interface

The system configuration page currently offers only two configuration options. One is end-host database viewing and configuring, the other is Trojan ports database. Similar to monitoring, both options can be invoked from the configuration home page.

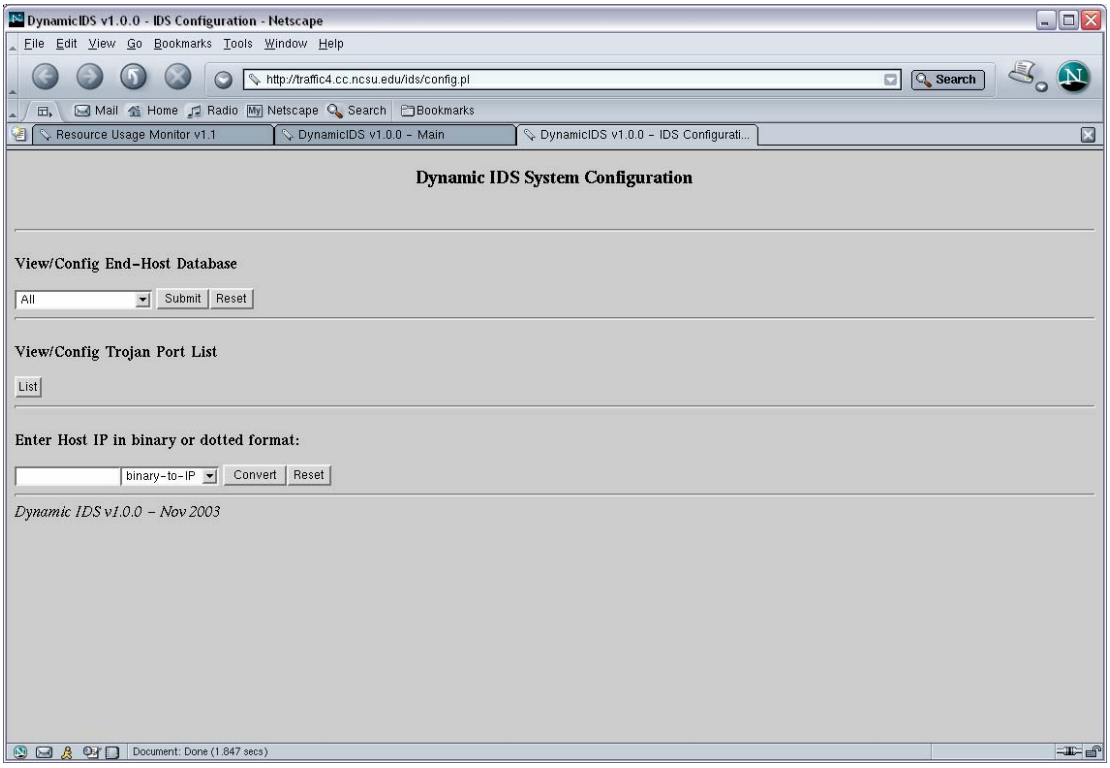


Figure 4.8 – Dynamic IDS Configuration Screenshot

In addition to the configuration options, an IP address utility is also provided. The utility can convert back and forth between the dotted decimal IP format and the binary format used in Perl Storable database files. It allows the administrator to view individual entries of the database files from command line and be able to figure out the corresponding IP address easily.

The host database configuration option lists the currently hosts in the end-hosts database by network. It lists the IP addresses and host names, as well as each host's allowed services and pre-defined client/server roles. It also gives options to modify any existing host, or add new host to the database. The Trojan Port List is presented in a similar fashion, where the page lists all the defined Trojan ports and corresponding description in a table.

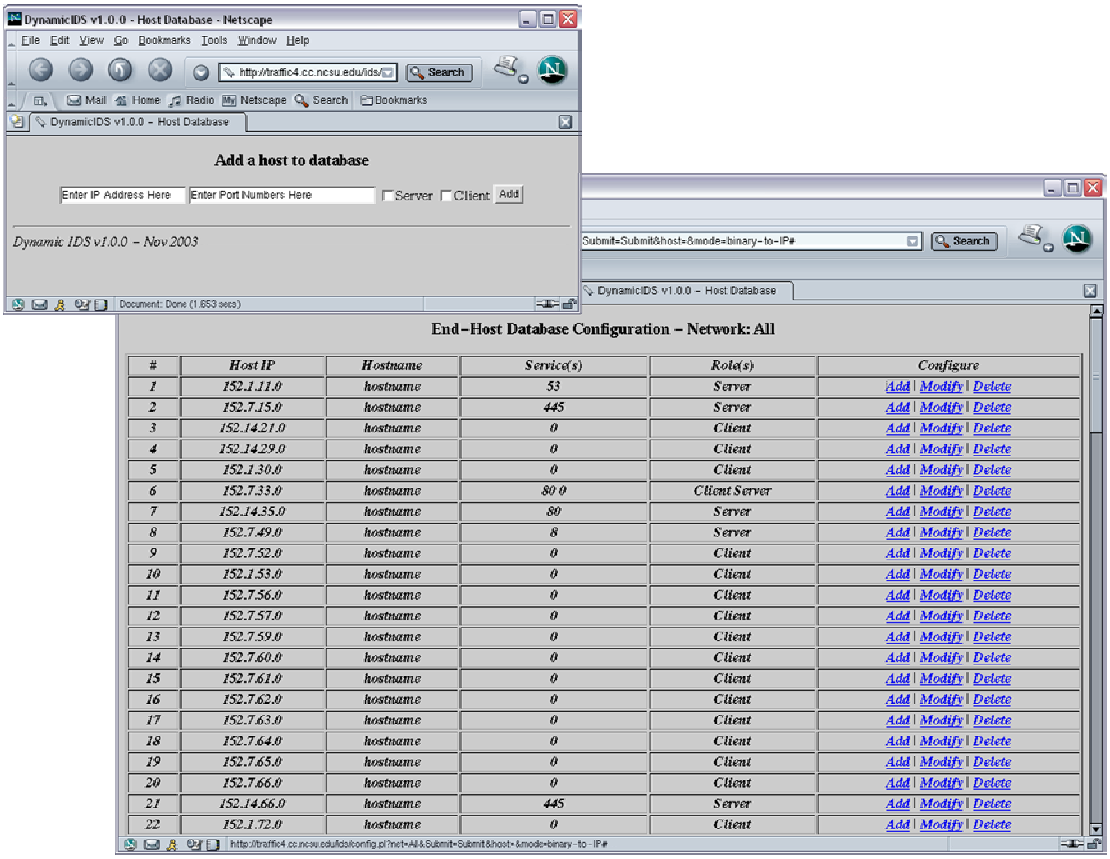


Figure 4.9 – End-Host Database Configuration Screenshot

4.5.3 Integration with RUM

The dynamic intrusion detection system integrates with RUM bi-directionally. In other words, it is possible to invoke IDS from within RUM and vice versa. To achieve this, links to IDS pages are place inside RUM's main menu, with both links to IDS monitoring and IDS configuration. On the other hand, from the IDS host details page, there is a link for users to view the traffic details captured by RUM for the current host.

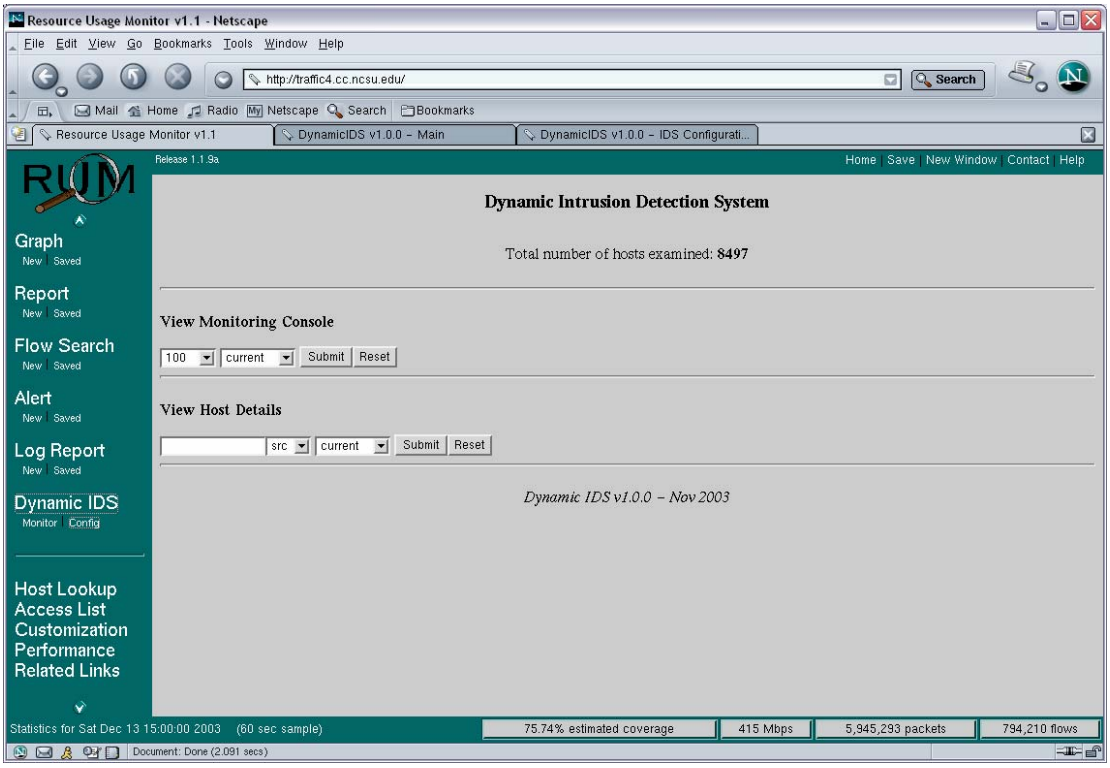


Figure 4.10 – Dynamic IDS & RUM Integration Screenshot

4.6 Setup and Configuration

Setting up the dynamic intrusion detection system involves choice of the right hardware and operating system, placing it at the right spot in the network (so the right information can be collected), installing dependent applications, and finally installing and configuring the main system itself.

4.6.1 System Hardware

The hardware requirements to run the dynamic intrusion detection system depend on the scale of network and the amount of traffic that is to be analyzed. The table below lists the minimum and recommended requirements for both small and large scale networks. A small scale means a single subnet network with several hundred hosts or

less and less than 100Mbps total network traffic, whereas a large scale means network with multiple subnets, thousands of hosts and up to 1Gbps total network traffic.

	Minimum	Recommended
Processor	1 x Pentium III 700MHz+	1 x Pentium III 1GHz+
Memory	256 MB or more	512 MB or more
Hard Drives	2 x 9GB IDE or SCSI	2 x 18GB+ SCSI
Network Cards	2 x 10/100Mbps	2 x 10/100Mbps

Table 4.4 – System Requirements for Small Scale Networks

	Minimum	Recommended
Processor	2 x Pentium III 1GHz+	2 x Pentium 4 Xeon, 2.8+ GHz per processor
Memory	1GB or more	2GB or more
Hard Drives	2 x 18GB Ultra SCSI	2 x 18GB+ Ultra160 SCSI
Network Cards	1 x 10/100Mbps and 1 x 10/100/1000Mbps	1 x 10/100Mbps and 1 x 10/100/1000Mbps

Table 4.5 – System Requirements for Large Scale Networks

4.6.2 Operating Environment

Dynamic IDS was developed and tested on Red Hat Linux 7.1 with 2.4.x kernel. All its codes are currently written in Perl and the system does not contain operating system specific components. Therefore, it should be able to run on other Linux and UNIX operating systems. Since dynamic IDS requires RUM as the data capturing engine and RUM is not design to run in Windows environment, dynamic IDS currently cannot run in Windows. However, with a substituted capturing engine such as WinDump [29], and certain modifications to the codes, the IDS can easily be ported to run on any version of Windows with Perl installed.

4.6.3 Placement

Since the dynamic intrusion detection system is a network-based IDS, it should be placed exactly the same way as a NIDS. If the network is behind a firewall, the best placing scheme is both in front of and behind the firewall [43].

4.6.4 Dependencies

Dependent applications include RUM, Nmap, and a web server. RUM is the system's packet capturing and sorting engine, while Nmap is used for optional host probing. Detailed installation procedures for the two applications can be found in the documentation of its own [31][34]. A web server such as Apache [50] is also required, since it uses web interface for monitoring and configuration.

4.6.5 Deployment

Installation of the main system can be done manually or with a make file and install script. Both ways are very straightforward. To install manually, a compressed file containing all the source codes need to be un-tarred into the same directory where RUM is installed. It creates three other directories:

- **`ids-src`** – contains main source codes
- **`ids-data`** – contains database files
- **`scripts`** – contains useful administrative scripts

In addition, the original **`menu.html`** file in **`www`** and **`www-src`** directories and **`RUMengine.pl`** file in **`bin`** and **`perl-src`** directories are replaced. The make file and install script performs the exact same actions.

After installation, the system needs to be initialized by running the **`IDSinit.pl`** from **`ids-src`** directory. This allows the system to run a few initial scans on the network to gather enough information as preparation and to create a host database as well as a network-to-subnet mapping file. Finally, a cron job should be added to ensure regular monitoring of the network data.

Chapter 5 – Evaluation

This chapter presents an initial exploratory evaluation of the dynamic intrusion detection metrics under investigation using a set of case studies. We compare the effectiveness of these metrics with respect to what human administrator investigation, Snort, and the original RUM are able to detect. The goals of this part of work are not to provide a statistical validity assessment of the IDS metric suite, but to provide sufficient amount of grounding (exploratory) information so that hypotheses regarding possible effectiveness can be formed. Actual statistical assessment of the effectiveness is beyond the scope of this work.

5.1 Case Studies

The evaluation case studies entailed direct monitoring of NC State University's incoming and outgoing networking traffic. Traffic sampling points are the same as those used for NC State RUM [31].

The first set of examined cases is based on a network traffic snapshot taken on March 15, 2004. The dataset contains 5,407,215 packets and 269,384 flows representing 488 Mbps of two-way NC State traffic. The snapshot is 60 seconds long. A total of 10,288 hosts were analyzed using regular RUM screens, and using Dynamic IDS extensions developed as part of this work[11].

[11] Individual IP numbers may have been modified to protect privacy of source and destination nodes.

Selected hosts from the IDS report were first examined, and then compared with the corresponding data and reports provided by RUM as we now discuss.

5.1.1 First Level Analysis – Ranking of Hosts

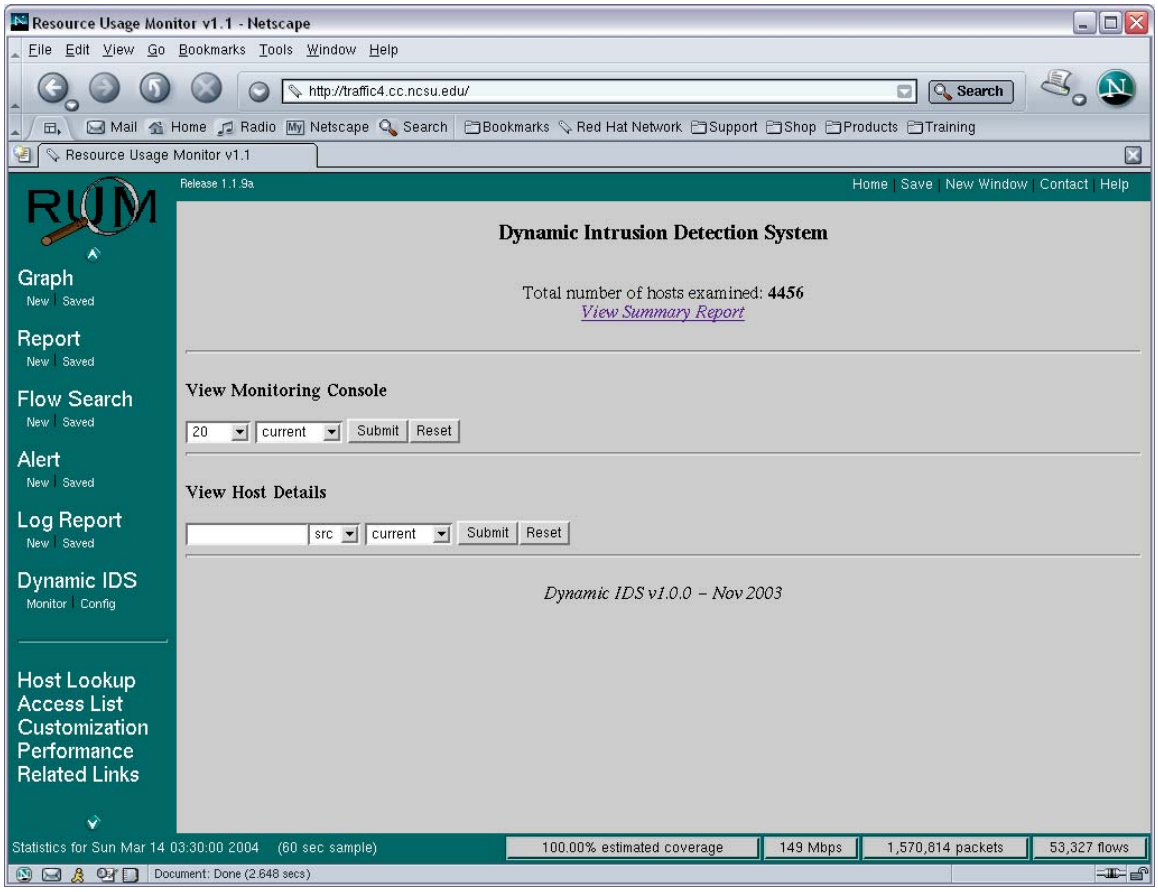


Figure 5.1 – Dynamic IDS Monitoring Main Page

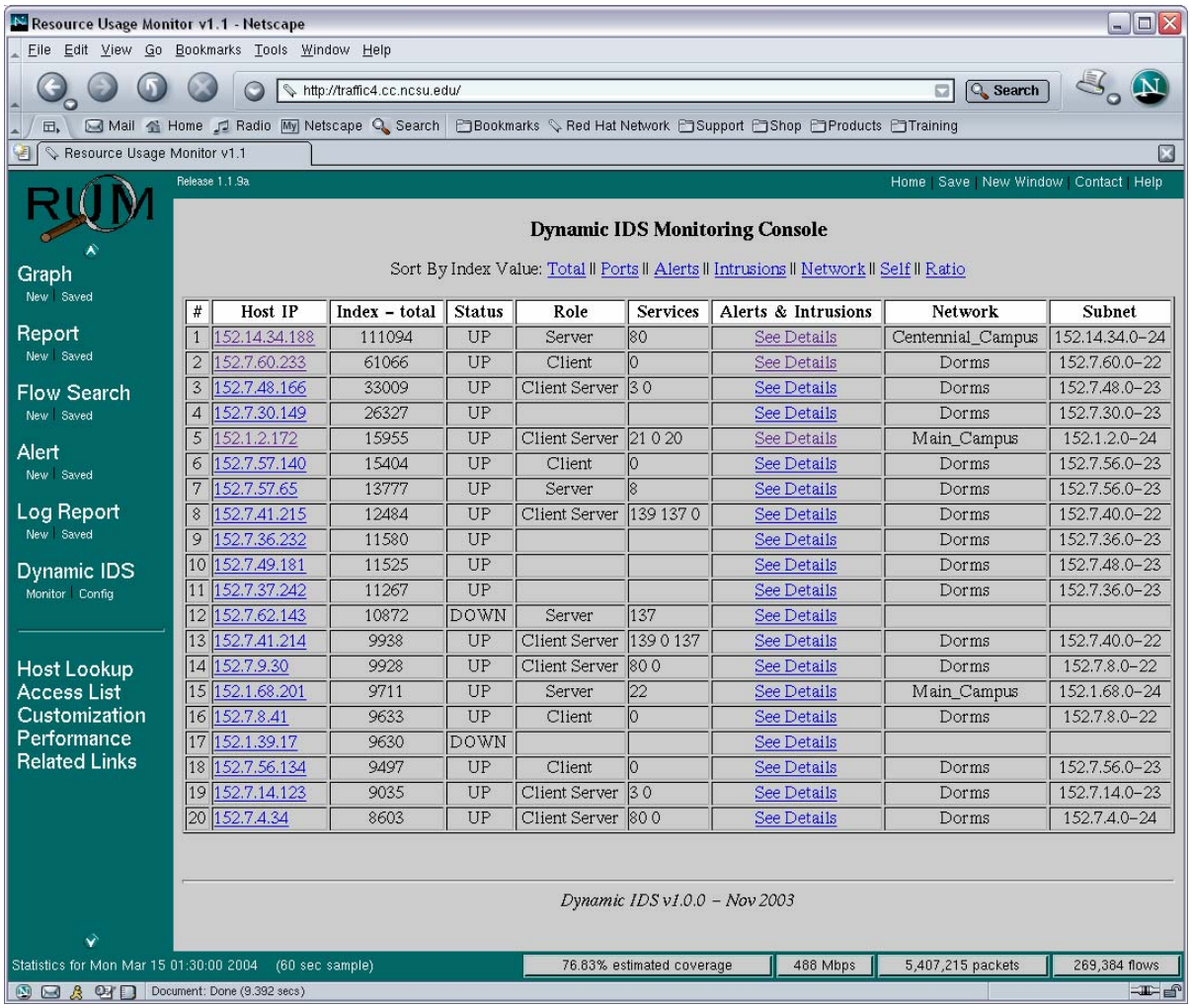


Figure 5.2 – Dynamic IDS Monitoring Console - Top 20 Hosts

Figure 5.2 (which is retrieved by selecting 20 hosts to view from the IDS main monitoring page in Figure 5.1) shows the top 20 hosts ranked by the total anomaly Index (third column). Of the 20 hosts 18 have (in this pass) status UP (i.e. the host is up and running in the current session) and the rest were DOWN (i.e. the host was up in the previous sample, which was taken approximately 30 minutes ago, but did not show up in the sample in this session) – fourth column. Most of them have had their

role and services tentatively defined[12] (fifth and sixth column). Further information about the IP numbers is available by following the link in the Alerts & Intrusions column which leads to the host details page. The last two columns provide more information about the network and subnet to which the host belongs. A couple of them also lack the network and subnet information because such mapping is not found in the release of the RUM's subnet-to-host database used for this experiment.

The IDS console list allows the user to sort the list of hosts by the total anomaly index, or any one of the six sub-indexing categories – ports, alerts, intrusions, network, self, or ratio.

[12] These roles and services were defined when Dynamic IDS was first initialized. During the initialization process, the system analyzed each host available in that particular sample and automatically defined every open port ranging from 1 to 1023 as a legitimate service. Hosts with at least one legitimate service were defined as a server.. To ensure accuracy, the system provides a configuration web interface so that the network administrator can modify the host role and services. However, intervension was not available to this project, so roles and services are tentative only.

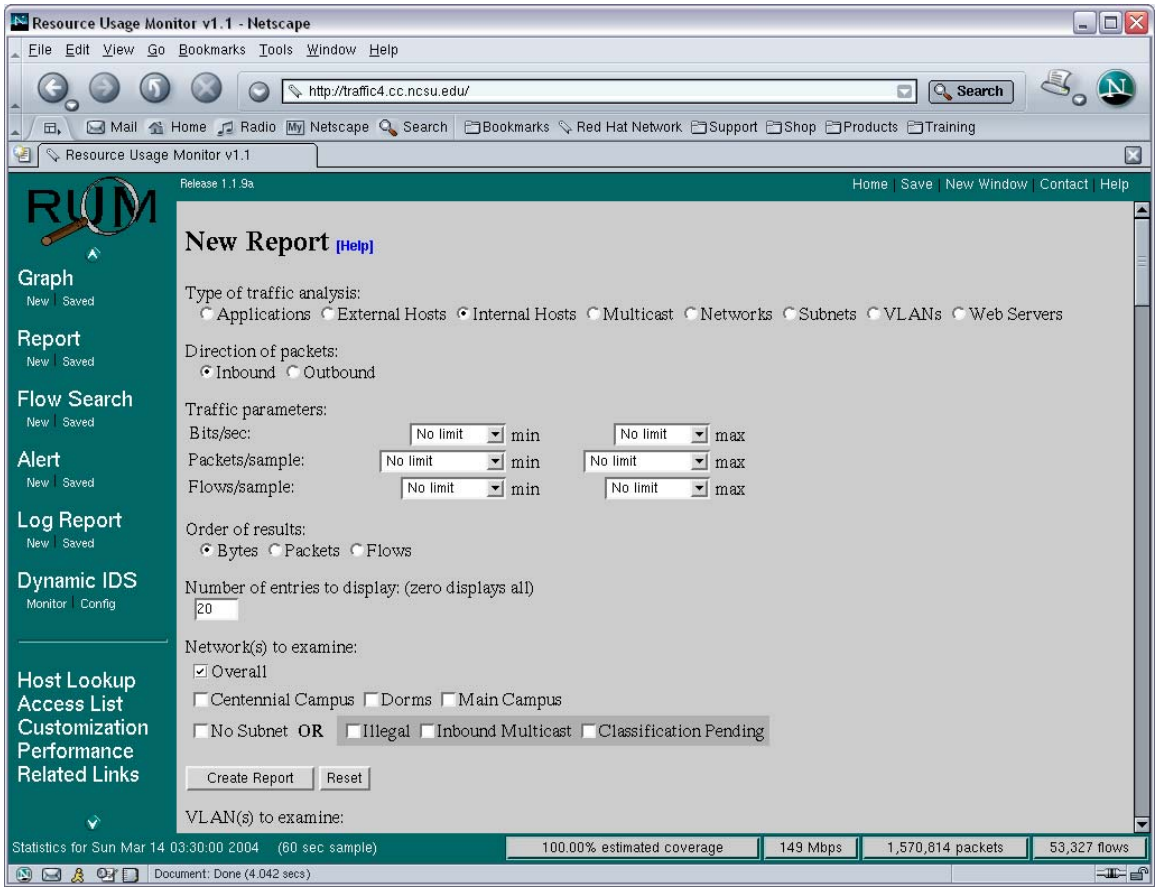


Figure 5.3 – RUM Report Main Configuration Page

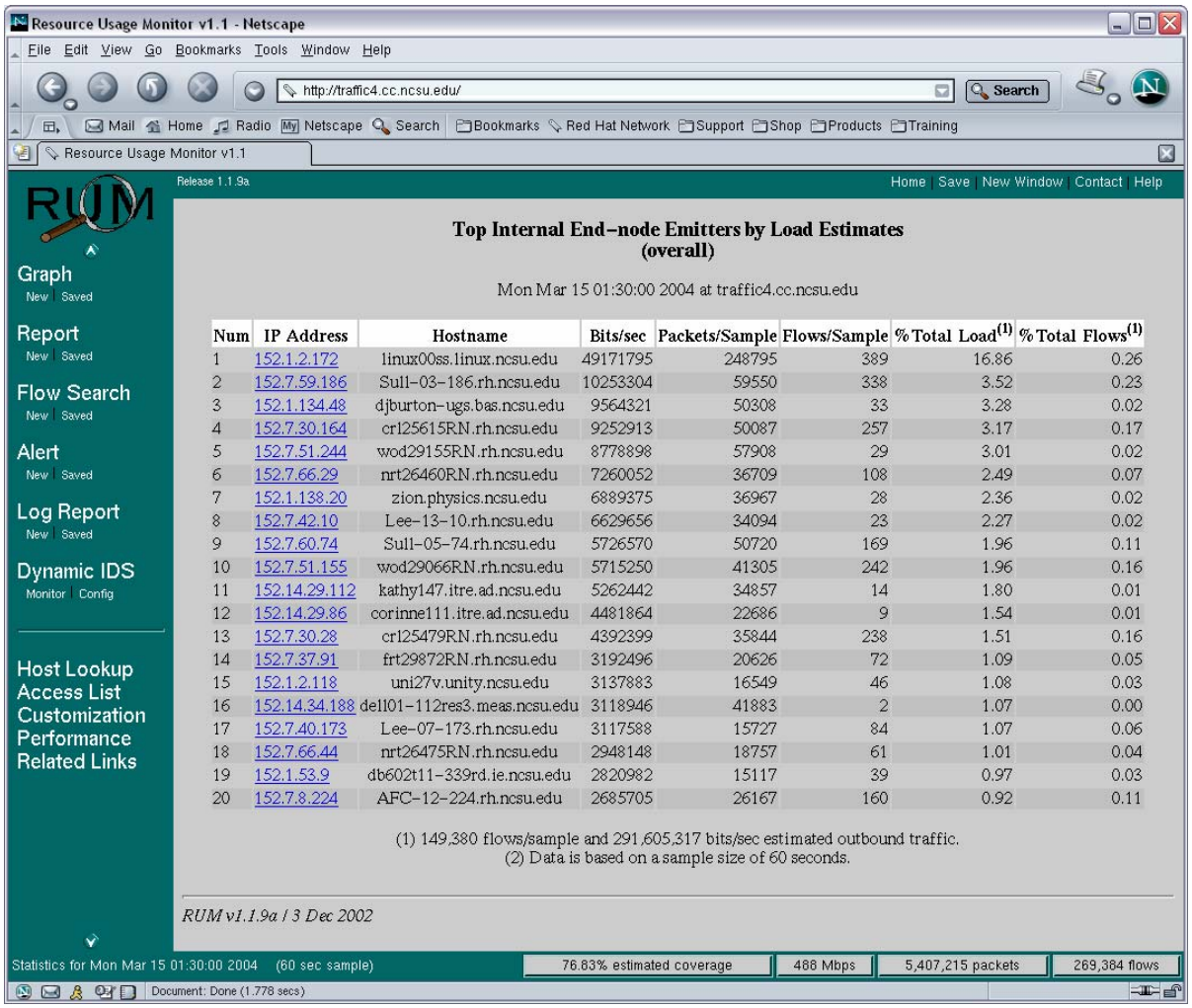


Figure 5.4 – RUM Top 20 Internal Load Emitters

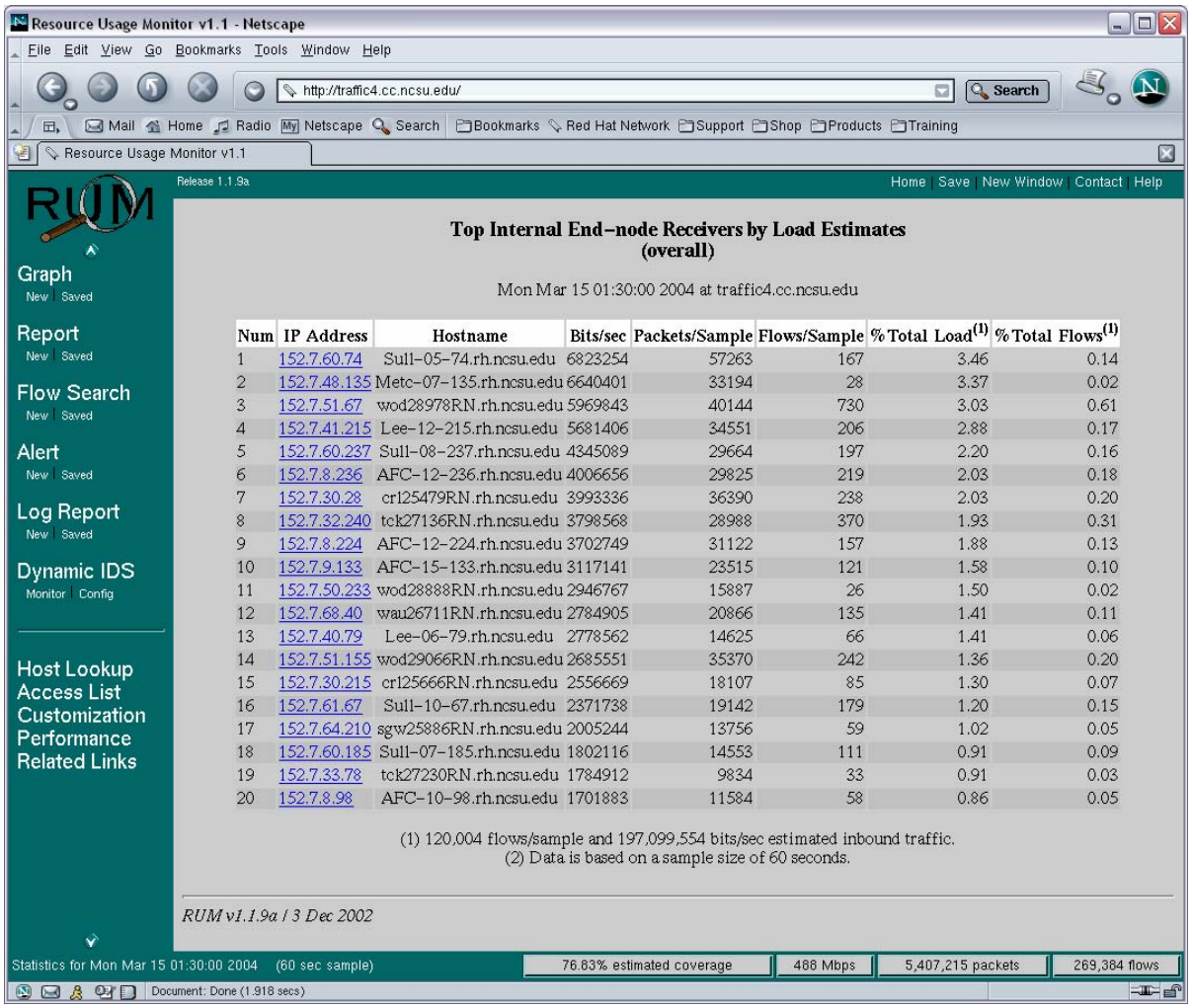


Figure 5.5 – RUM Top 20 Internal Load Receivers

The RUM console can sort the sample by flow intensity (flows per sample) load intensity (bits per second) and by the number of packets per sample. In addition it can select between incoming and outgoing flows. Figure 5.4 illustrates the sort of the top 20 hosts by flow load (outgoing). Figure 5.5 the sort by received load. Similarly Figure 5.6 through Figure 5.9 shows the sorts for packets and flow intensity. All of which are generated from the main report configuration page shown in Figure 5.3.

The first glance reveals that the ranking of DIDS and RUM is very different. For example, the number one host on the DIDS ranking (from Figure 5.2) ranked number 3816 on RUM's Load Receivers list, number 16 on Load Emitters list, number 3051 on Packet Receivers list, number 7 on Packet Emitters list, number 7444 on Flow Receivers list, and finally number 1983 on Flow Emitters list. This particular host ranked low on the Load and Packets Receivers list because it had a very small amount of incoming traffic (only 3 packets and 1.5 kilobits of load). But its outgoing traffic is considerably larger (over 30,000 packets and over 3 Mbps load), therefore the higher ranking as a Load and Packet Emitter. Again its flows were small for both ways – only 2 outgoing and 1 incoming flows. This host ranked number one on DIDS ranking mainly due to its significantly large difference in incoming and outgoing traffic amount, which generated over 10,000 in I/O ratio index. This host is explored in greater details in the next section (5.1.2).

The second host on the DIDS ranking had low rankings as a Load Receiver and Emitter, as well as a Packet Receiver and Emitter – number 366, 202, 84, and 33, respectfully – all of it due to relatively low amount of actual traffic associated with it. However, it had a very large number of flows (both inbound and outbound), therefore it ranked at the very top of both Flow Receivers and Flow Emitters lists. Again, this host is discussed in greater details in the next section.

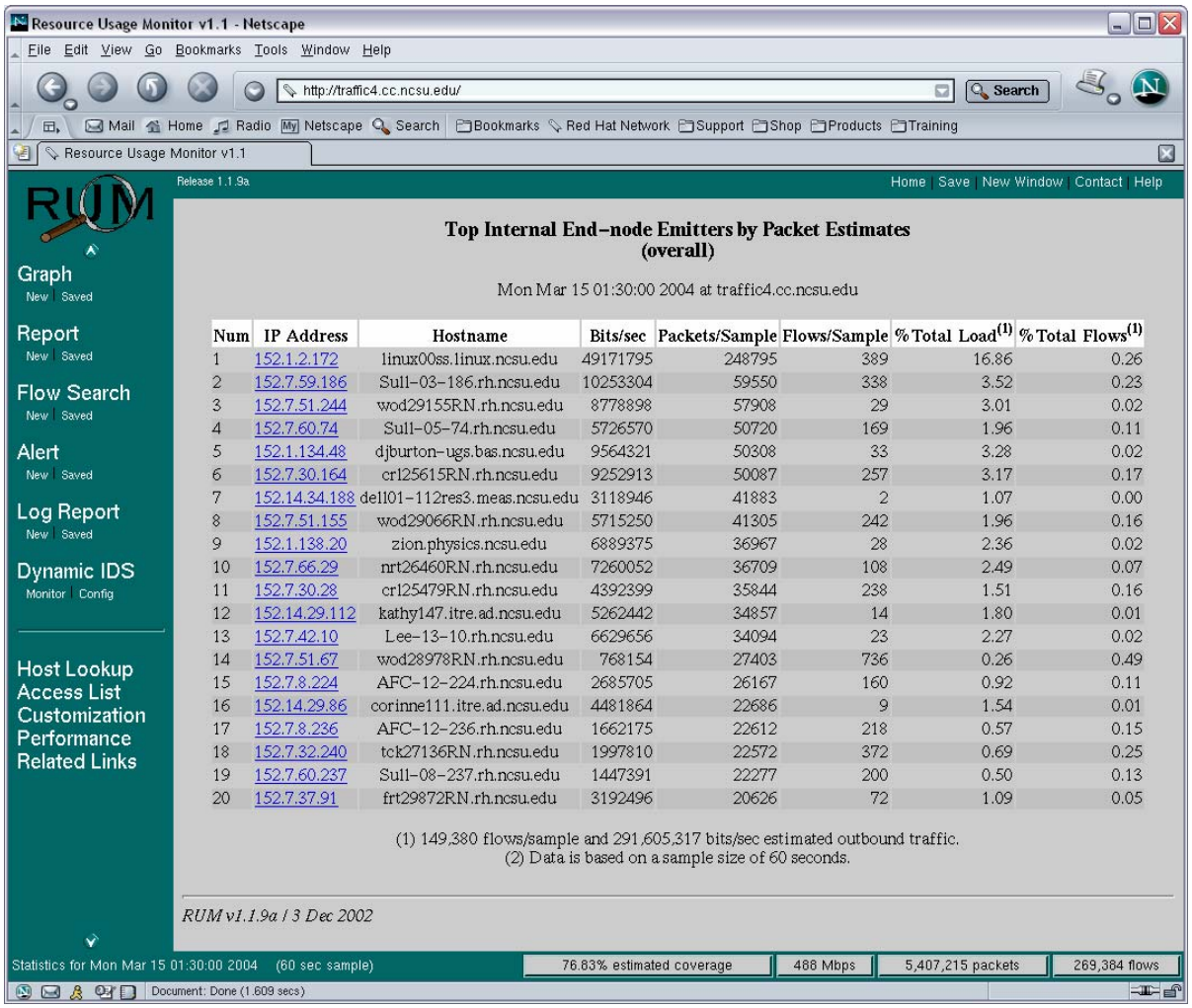


Figure 5.6 – RUM Top 20 Internal Packet Emitters

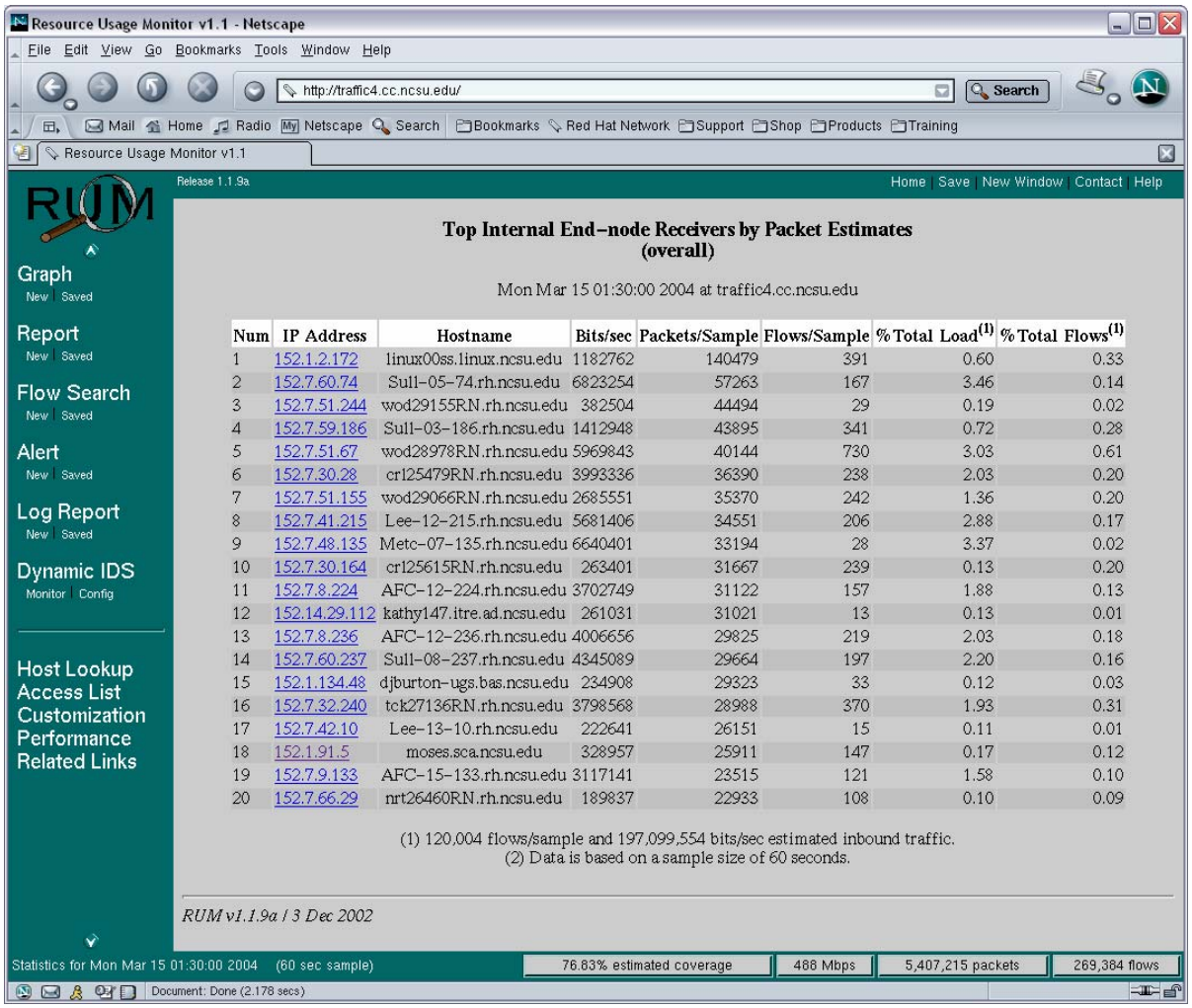


Figure 5.7 – RUM Top 20 Internal Packet Receivers

On the other hand, the host 152.1.2.172 that ranked number one on 3 of the 6 reports generated by RUM (Load Emitters – Figure 5.4, Packet Emitters – Figure 5.6, and Packet Receivers – Figure 5.7) was only placed number 5 on the DIDS ranking (Figure 5.2). This host had a lower index than the first four hosts on the DIDS ranking and because its traffic pattern was not as "unusual" as that of the first four.

In short, DIDS considers more parameters and assumes that an additive effect of several relatively normal or perhaps only slightly "unusual" events (that may

otherwise go unnoticed) may be an indicator of a host that needs further scrutiny. Hence the top hosts on RUM's ranking are not necessarily the top "unusual" hosts on the DIDS's list, and vice versa. This will become clear once we get into the detailed discussion of each individual hosts in the coming section.

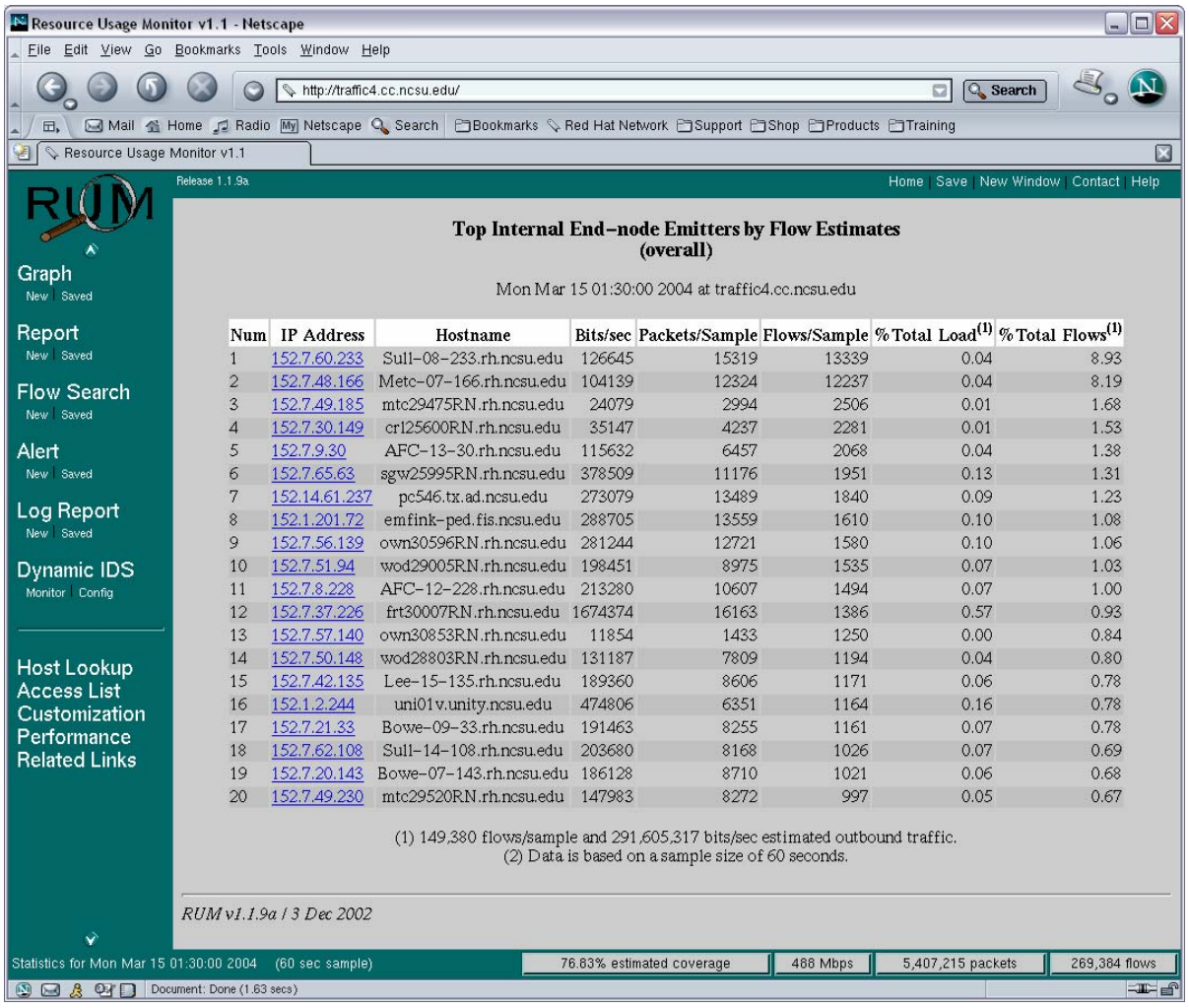


Figure 5.8 – RUM Top 20 Internal Flow Emitters

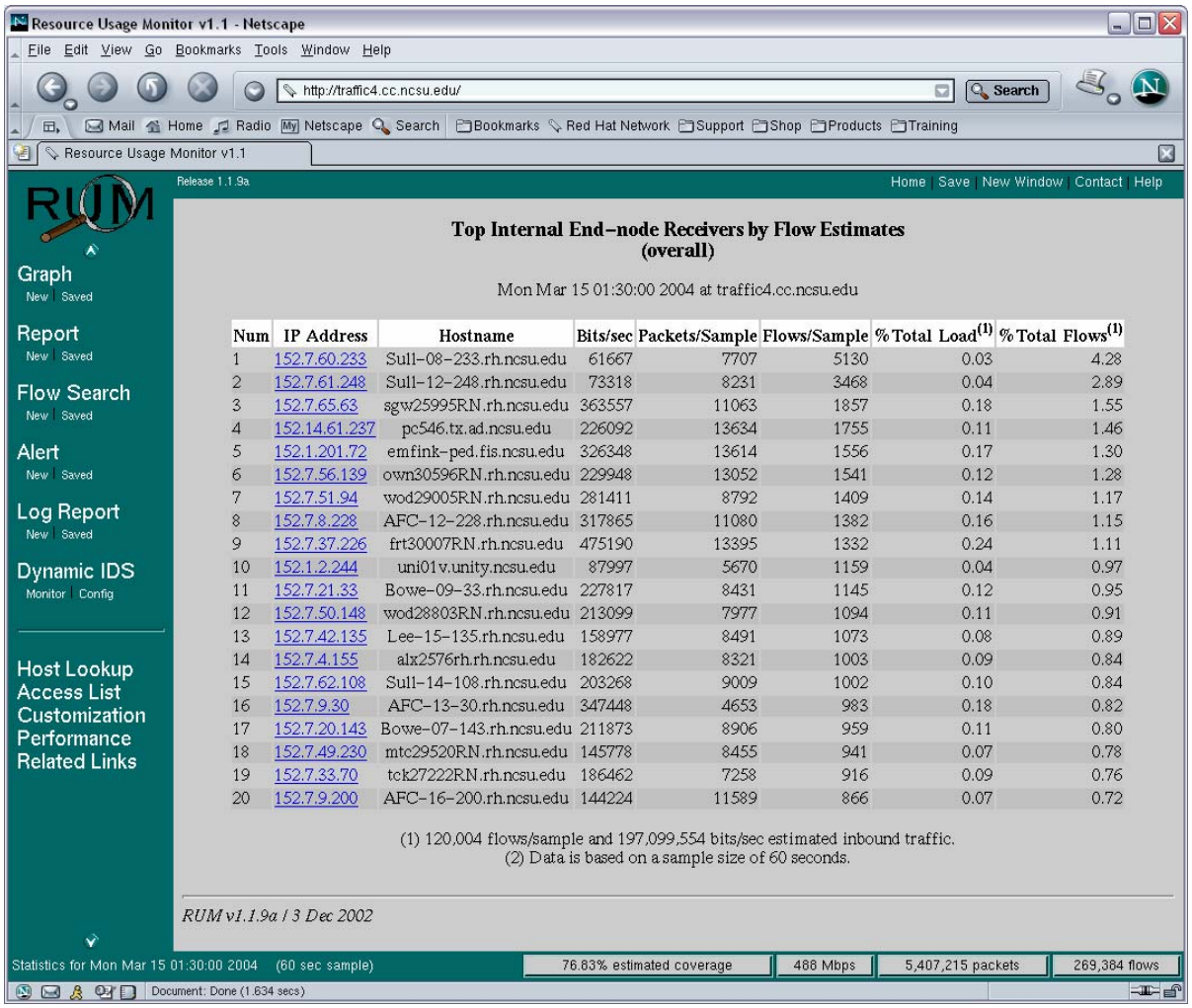


Figure 5.9 – RUM Top 20 Internal Flow Receivers

5.1.2 Second Level Analysis – Top DIDS Hosts

In both the DIDS Console panel and the RUM host-ranking panels, details of the hosts are presented by selecting the host's IP address or clicking on either the IP address or on the "See Details" link from the DIDS page. Figure 5.10 shows the snapshot of details for the first host on the DIDS list (Figure 5.2).

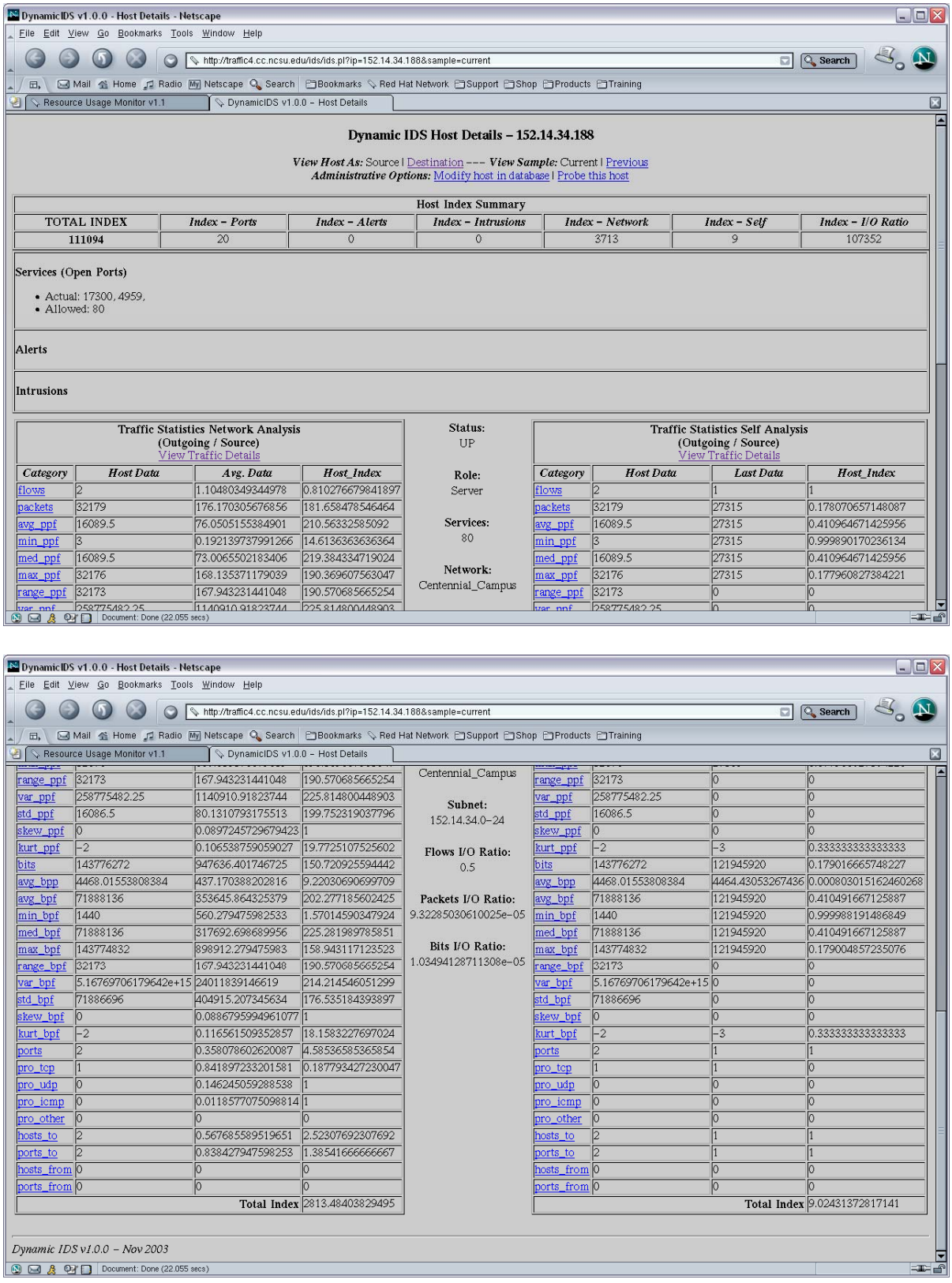


Figure 5.10 – DIDS Host Details (#1) – 152.14.34.188 as Source (Outbound)

At the very top of the host details page, users can choose to view the host as either a source (outbound) or as a destination (inbound) and to view details for the current or past sample. Currently, this host is viewed as a source in the current sample. Optionally, the user can also modify the host database to enter more accurate role and services information for the host or to probe the host for open ports by following the other two links on the top of the page (this is shown in section 4.5).

Inspection of this host's index summary details (in Figure 5.10, first table) shows that out of the six index categories (defined and justified in detail in section 3.7), the I/O Ratio index is the dominant contributor to this host's high total anomaly index (which, of course, is the sum of the individual indices in the top table row on that page). In fact, the I/O Ratio index contributes over 96% of the total index. Reasons that the other categories are low in index can be found by continuing onto the other sections of the page. The next table shows that the host was communicating on two ports (17300, and 4959), where the host's registered service was 80 (web server). The two unregistered ports were not considered to be significant enough, and thus the Ports index reflected merely an index of 20[13]. In addition, it had no alerts or intrusion warnings, which made the index for both of the two categories zero.

[13] See section 3.8.2.1 for details on index from communicating ports.

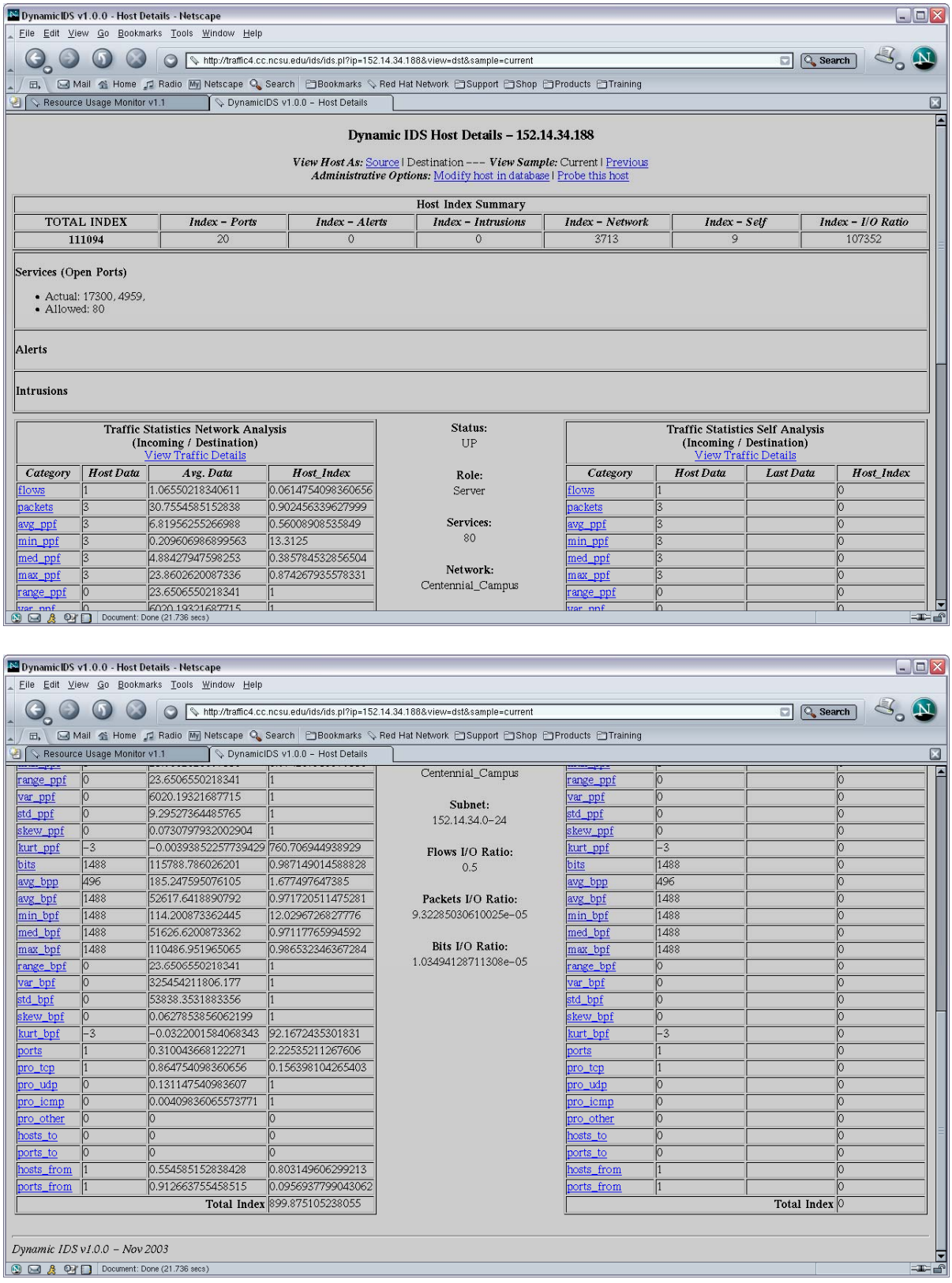


Figure 5.11 – DIDS Host Details (#1) – 152.14.34.188 as Destination (Inbound)

The second part of the page displays more details on network- and self-relative comparisons, actual ratio values, and other host information. By looking at the actual ratio figures (in the middle of the page between left and right tables), the host appears to have significantly more outbound packets than inbound packets and load – the outbound packets are over 10 thousand times that of the inbound packets, to be specific; and for load (bits), outbound is over 96 thousand times bigger than inbound. However, the inbound outbound difference in terms of number of flows is not as drastic (only twice as much outbound flows than inbound).

The network analysis table on the left confirms the results. This table presents in detail each individual parameter[14] (Category column) and the actual data (Host Data column) that the system uses to represent the host's traffic behavior, as well as the data for network average[15] (Avg. Data column), and the difference between the two (Host_Index column). The last row shows the total index generated by network comparison for the host as a source (Figure 5.10) and as a destination (Figure 5.11), and those two numbers together is the total index from network analysis as shown on the top of the page. Specifically, as a source, the index is relatively high for packets, bits, and average, median, maximum, range, variance and standard deviation for packets and bits per flow. These parameters all indicate issues in regards to the amount of traffic and traffic pattern. The same parameters did not generate much index at all for the host as a destination. This difference matches the high I/O ratio

[14] Clicking on the parameters will bring up a description message. Detailed discussion for each individual parameter can be found in section 3.6.

[15] Network average varies depend on the network a particular host resides on, the host's role, and traffic direction. Please refer to section 3.8.2.3 for complete details.

discussed in the previous paragraph. The parameters with high index (as destination) are the kurtosis for packets and bits per flow. A negative kurtosis indicates a relatively flat distribution (please refer to section 3.6), where in this case, the host only had 1 incoming flow.

Results for self-relative comparisons are displayed in the same way in the table to the right of the page. This particular host, when viewed as a destination, as shown in the table, had no previous data. This, at first glance, may suggest that the host was previously down and came up in this session, but the host had previous data as a source and there is not a "host up this session" alert shown. Therefore, the host was in fact up but its inbound traffic was inactive in the last sample. The previous data can be viewed by selecting previous sample on the top of the page (Figure 5.12 and Figure 5.13). From Figure 5.12 we can see that the host did have activities as a source, and from Figure 5.13 we can see that it was inactive as a destination host.

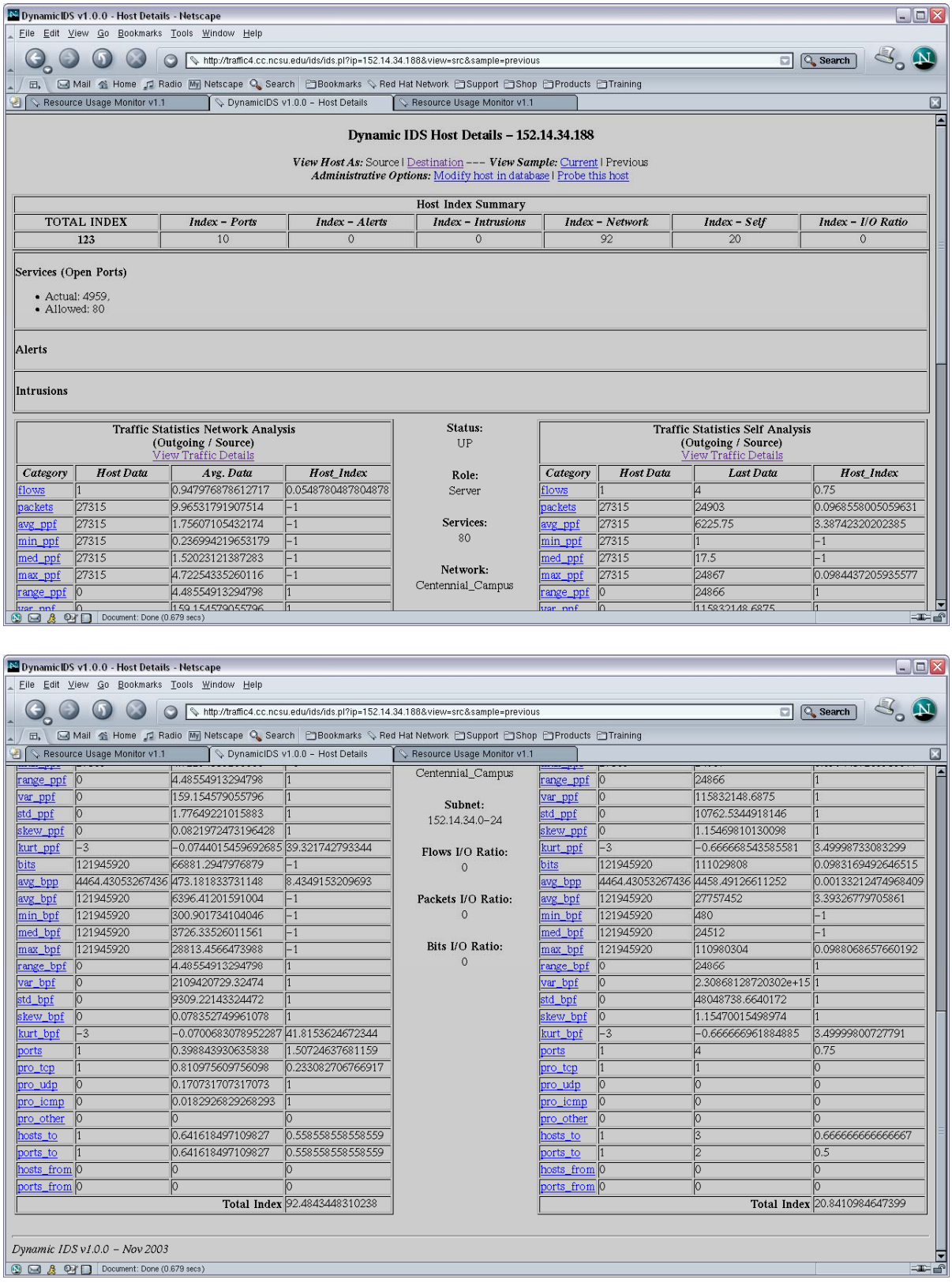


Figure 5.12 – DIDS Host Details (#1) – 152.14.34.188 as Source (Previous Sample)

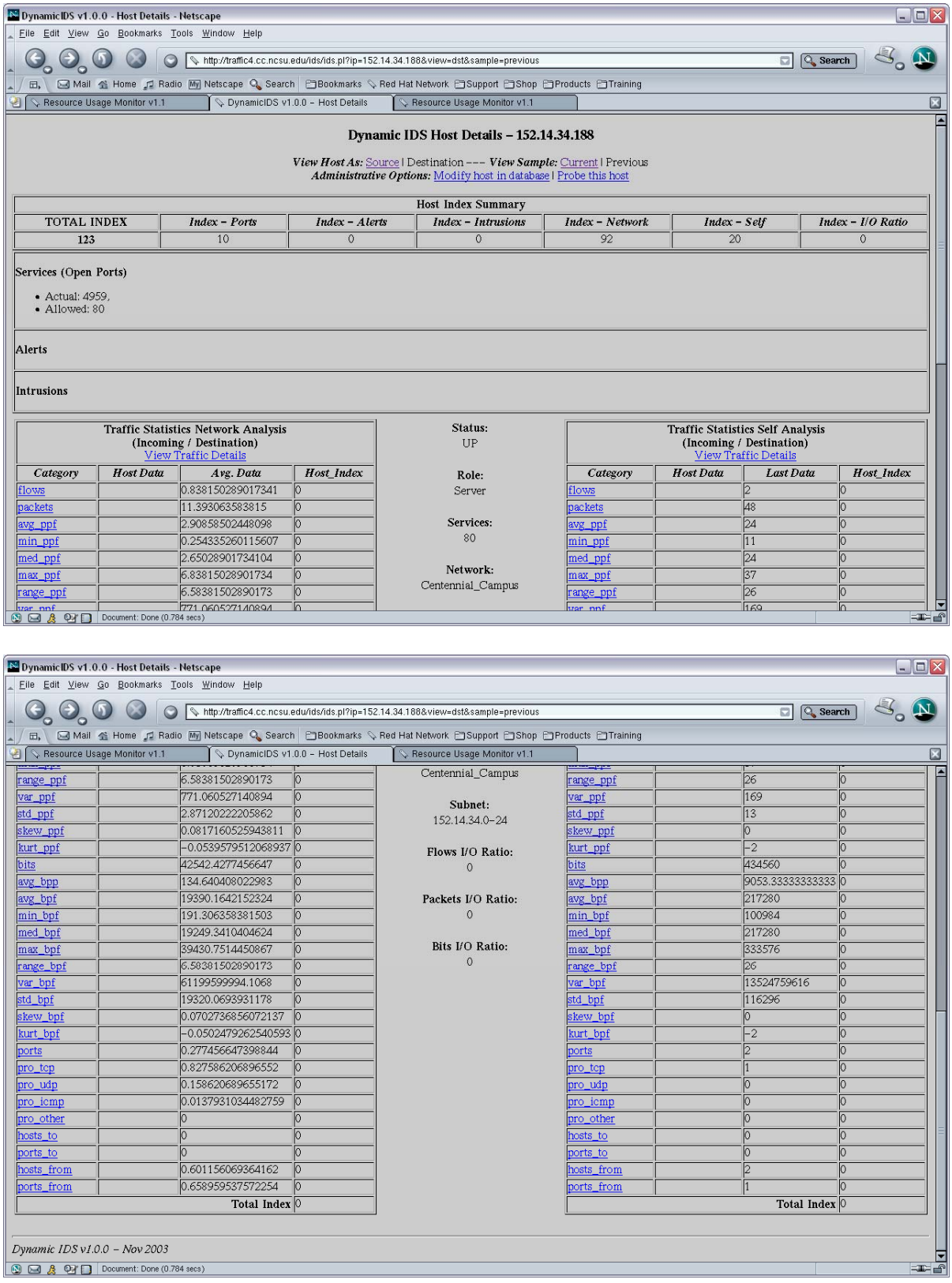


Figure 5.13 – DIDS Host Details (#1) – 152.14.34.188 as Destination (Previous Sample)

To further verify the analysis done by DIDS on this host, the actual traffic logs for this particular host are examined (Figure 5.14 and Figure 5.15). The large amount of data was sent in one flow to port 22, which is the default port for SSH (Secure Shell). Clearly, this infected host was performing a SSH buffer overflow attack[16]. Although DIDS did not specifically point out the type of attack[17], it still considered this host to be the most suspicious and put it on the very top of the list. On the other hand, this case has proven that DIDS is capable of detecting attacks that is not pre-defined.

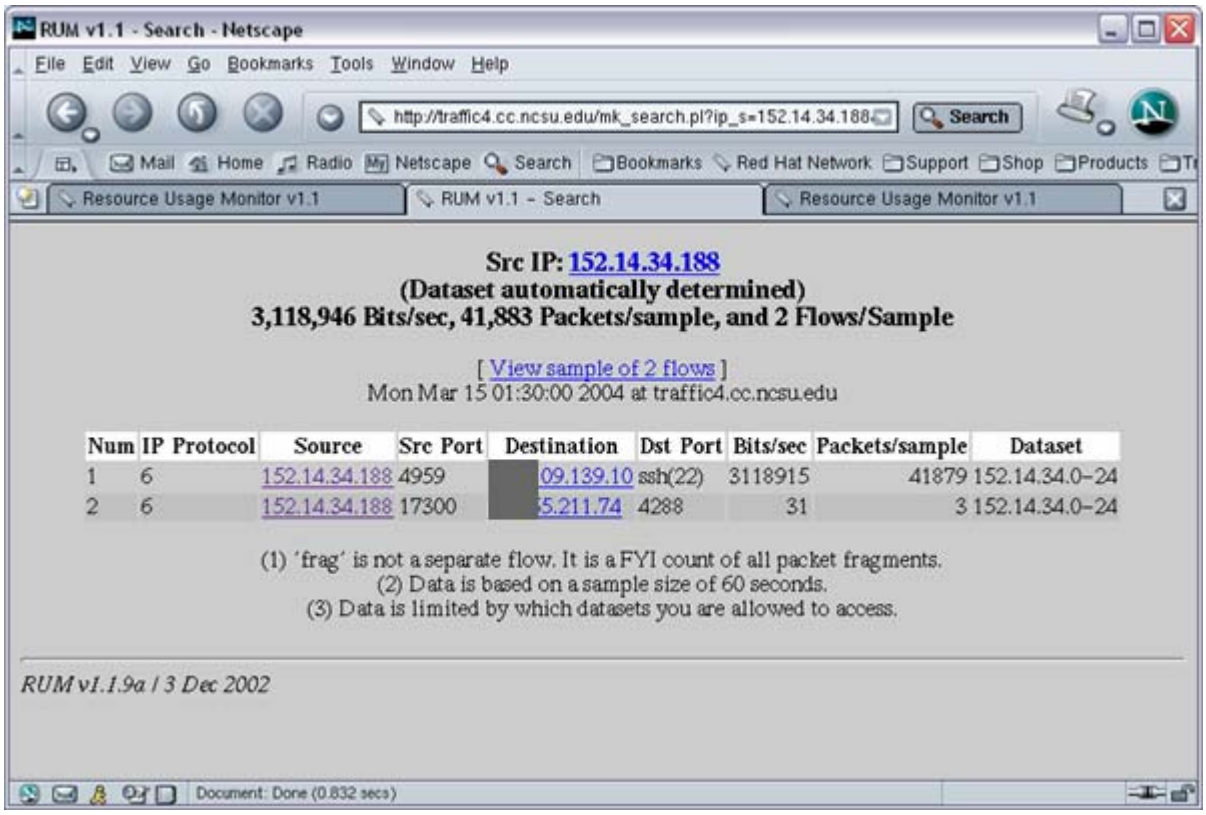


Figure 5.14 – RUM Host Outgoing Flows – 152.2.14.34.188

[16] Several implementations of the SSH tools have a buffer overflow in the SSH version 1 protocol code. The buffer overflow can be used by a remote attacker to execute arbitrary code, in most cases with the permissions of the root user. (http://www.linuxdevcenter.com/pub/a/linux/2001/11/19/insecurities.html)

[17] The main reason that DIDS did not point out specifically the type of this attack is that the current system only uses flow input/output ratio to identify the attack type. This is a potential enhancement that can be added to the system implementation in the future.

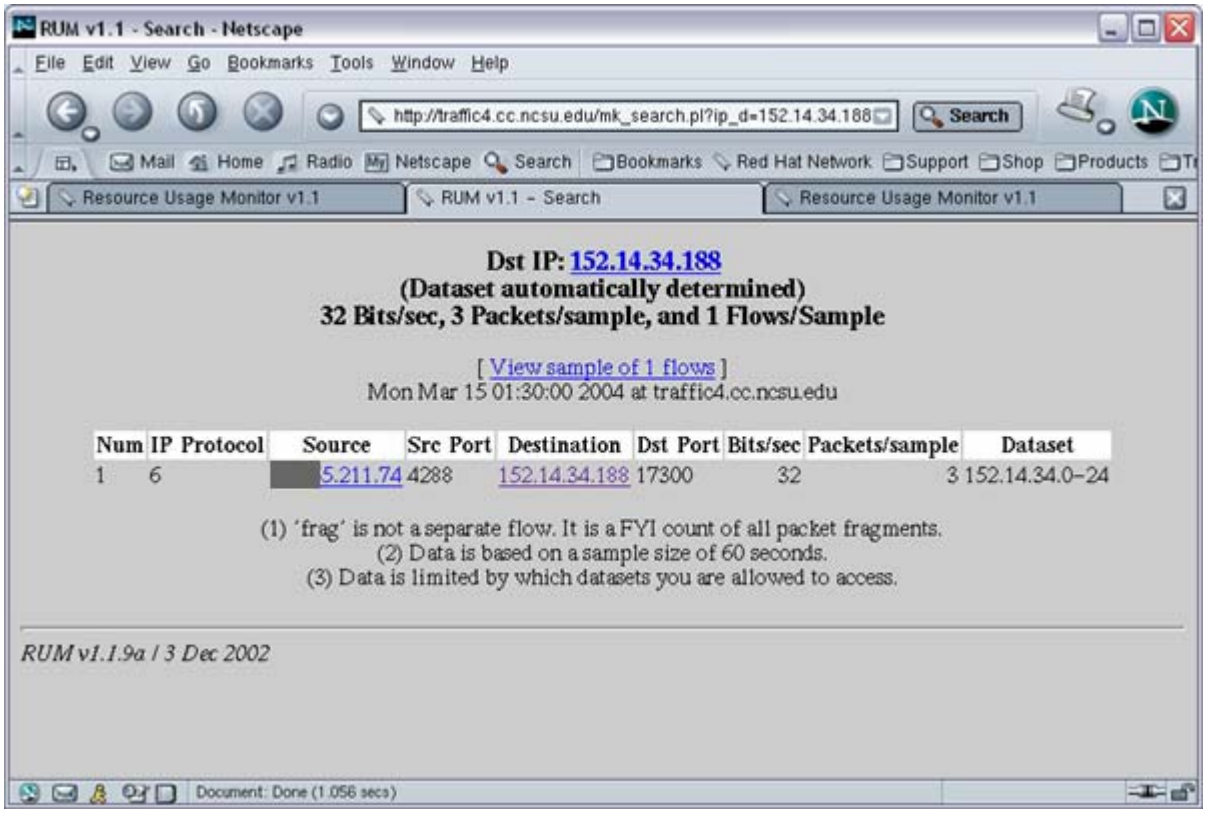
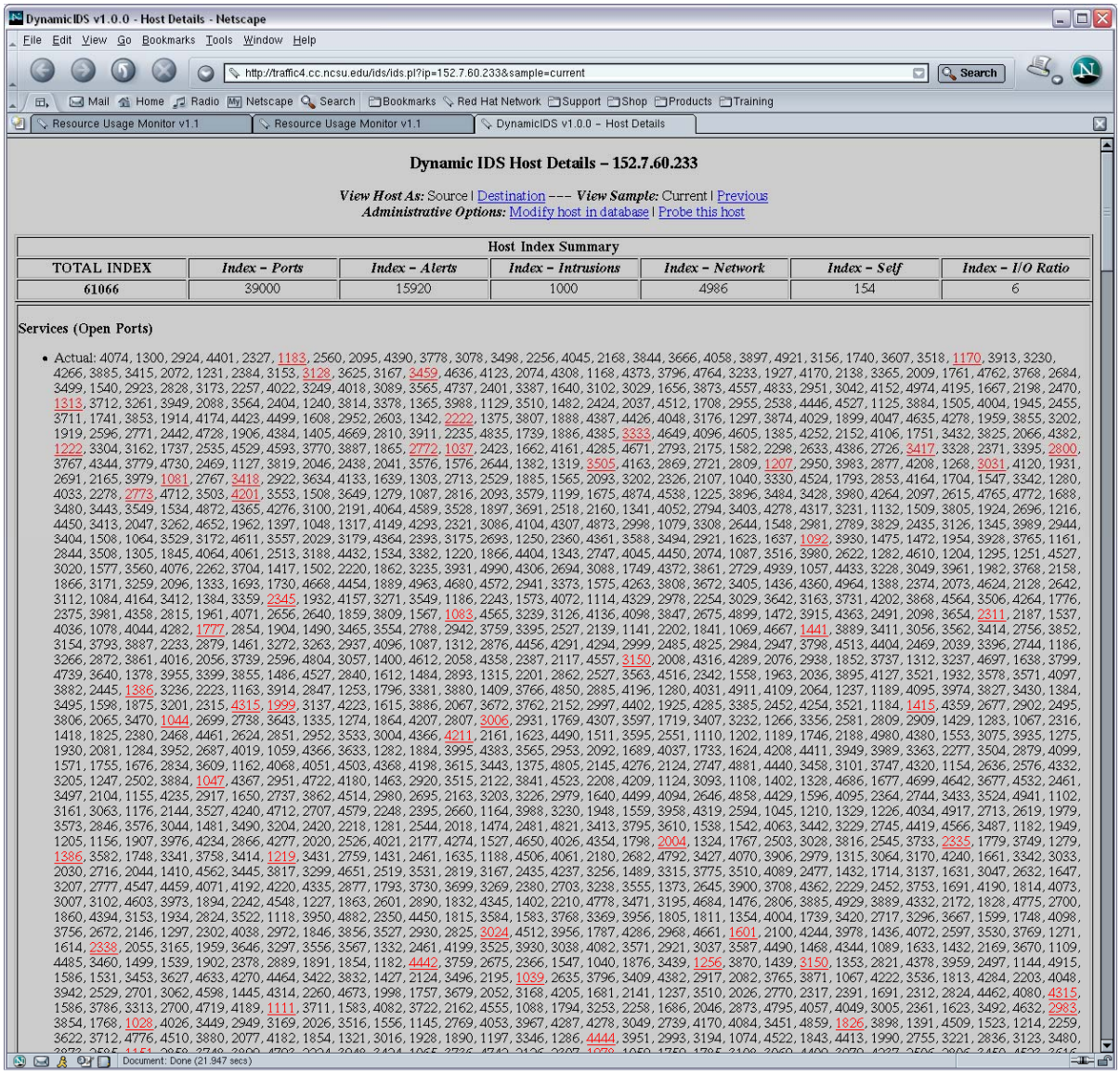


Figure 5.15 – RUM Host Incoming Flows – 152.2.14.34.188

Recall that in the RUM rankings, as discussed briefly in section 5.2.1, this particular host was placed very low as load and packet receiver (below 3000). It showed up as the 16th on RUM's top outbound load emitters and 7th on packet emitters list. Clearly, either one of the two was not significant enough individually to be on the very top of the list. Without taking consideration of both inbound and outbound traffic data at the same time, the real issue could not be identified. This is one place where DIDS is superior to RUM.

Moving down on the top list generated by DIDS (Figure 5.2), the next screenshot (Figure 5.16) shows the host with the second ranking.



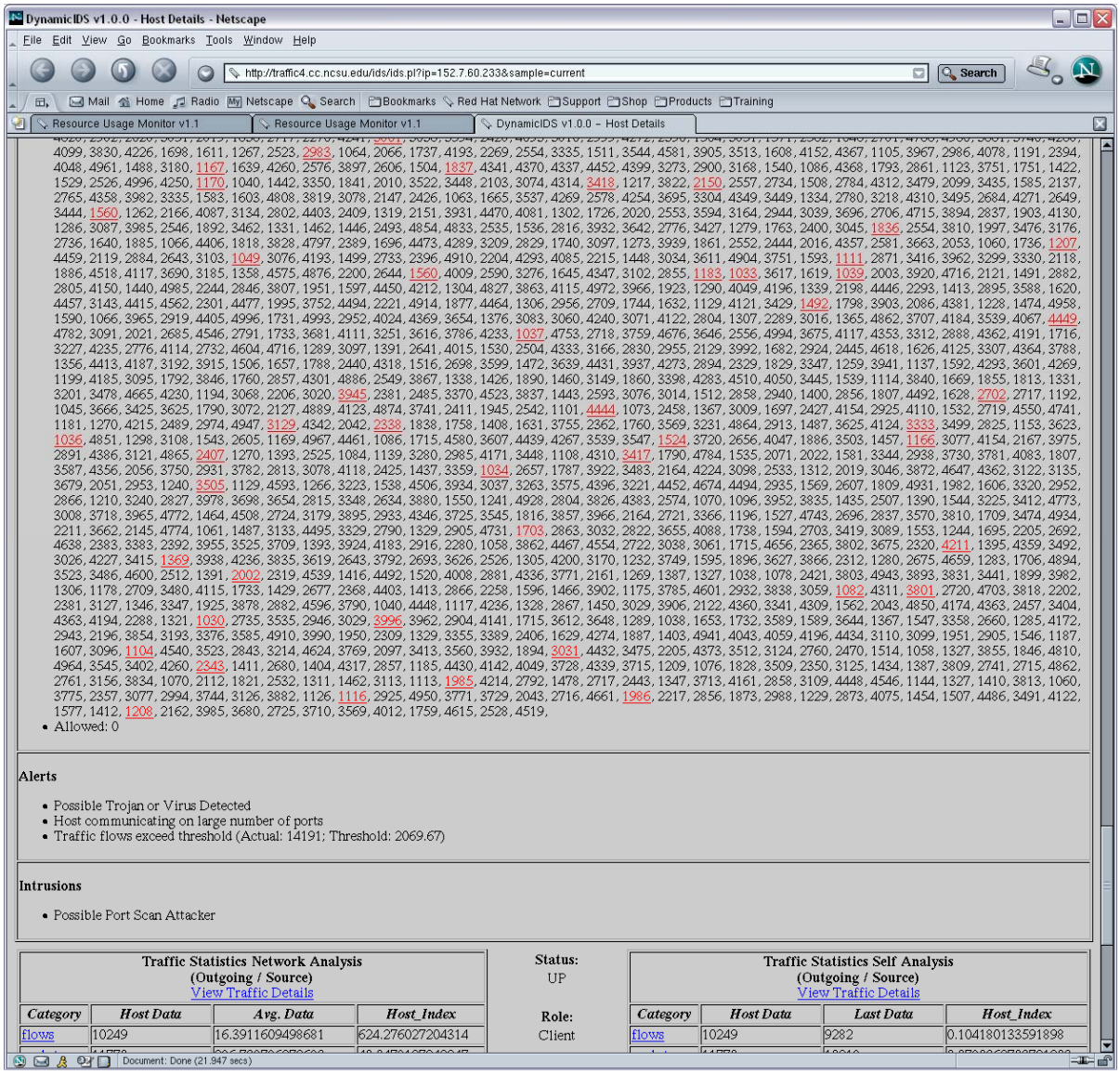
Dynamic IDS Host Details – 152.7.60.233

View Host As: Source | Destination --- ***View Sample:*** Current | Previous
Administrative Options: Modify host in database | Probe this host

Host Index Summary						
TOTAL INDEX	*Index – Ports*	*Index – Alerts*	*Index – Intrusions*	*Index – Network*	*Index – Self*	*Index – I/O Ratio*
61066	39000	15920	1000	4986	154	6

Services (Open Ports)

- Actual: 4074, 1300, 2924, 4401, 2327, 1183, 2560, 2095, 4390, 3778, 3078, 3498, 2256, 4045, 2168, 3844, 3666, 4058, 3897, 4921, 3156, 1740, 3607, 3518, 1170, 3913, 3230, 4266, 3885, 3415, 2072, 1231, 2384, 3153, 3128, 3625, 3167, 3459, 4636, 4123, 2074, 4308, 1168, 4373, 3796, 4764, 3233, 1927, 4170, 2138, 3365, 2009, 1761, 4762, 3768, 2684, 3499, 1540, 2923, 2828, 3173, 2257, 4022, 3249, 4018, 3089, 3565, 4737, 2401, 3387, 1640, 3102, 3029, 1656, 3873, 4557, 4833, 2951, 3042, 4152, 4974, 4195, 1667, 2198, 2470, 1313, 3712, 3261, 3949, 2088, 3564, 2404, 1240, 3814, 3378, 1365, 3988, 1129, 3510, 1482, 2424, 2037, 4512, 1708, 2955, 2538, 4446, 4527, 1125, 3884, 1505, 4004, 1945, 2455, 3711, 1741, 3853, 1914, 4174, 4423, 4499, 1608, 2952, 2603, 1342, 2222, 1375, 3807, 1888, 4387, 4426, 4048, 3176, 1297, 3874, 4029, 1899, 4047, 4635, 4278, 1959, 3855, 3202, 1919, 2596, 2771, 2442, 4728, 1906, 4384, 1405, 4669, 2810, 3911, 2235, 4835, 1739, 1886, 4385, 3333, 4649, 4096, 4605, 1385, 4252, 2152, 4106, 1751, 3432, 3825, 2066, 4382, 1222, 3304, 3162, 1737, 2535, 4529, 4593, 3770, 3887, 1865, 2772, 1037, 2423, 1662, 4161, 4285, 4671, 2793, 2175, 1582, 2298, 2633, 4386, 2726, 3417, 3328, 2371, 3395, 2800, 3767, 4344, 3779, 4730, 2469, 1127, 3819, 2046, 2438, 2041, 3576, 1576, 2644, 1382, 1319, 3505, 4163, 2869, 2721, 2809, 1207, 2950, 3983, 2877, 4208, 1268, 3031, 4120, 1931, 2691, 2165, 3979, 1081, 2767, 3418, 2922, 3634, 4133, 1639, 1303, 2713, 2529, 1885, 1565, 2093, 3202, 2326, 2107, 1040, 3330, 4524, 1793, 2853, 4164, 1704, 1547, 3342, 1280, 4033, 2278, 2773, 4712, 3503, 4201, 3553, 1508, 3649, 1279, 1087, 2816, 2093, 3579, 1199, 1675, 4874, 4538, 1225, 3896, 3484, 3428, 3980, 4264, 2097, 2615, 4765, 4772, 1688, 3480, 3443, 3549, 1534, 4872, 4365, 4276, 3100, 2191, 4064, 4589, 3528, 1897, 3691, 2518, 2160, 1341, 4052, 2794, 3403, 4278, 4317, 3231, 1132, 1509, 3805, 1924, 2696, 1216, 4450, 3413, 2047, 3262, 4652, 1962, 1397, 1048, 1317, 4149, 4293, 2321, 3086, 4104, 4307, 4873, 2998, 1079, 3308, 2644, 1548, 2981, 2789, 3829, 2435, 3126, 1345, 3989, 2944, 3404, 1508, 1064, 3529, 3172, 4611, 3557, 2029, 3179, 4364, 2393, 3175, 2693, 1250, 2360, 4361, 3588, 3494, 2921, 1623, 1637, 1092, 3930, 1475, 1472, 1954, 3928, 3765, 1161, 2844, 3508, 1305, 1845, 4064, 4061, 2513, 3188, 4432, 1534, 3382, 1220, 1866, 4404, 1343, 2747, 4045, 4450, 2074, 1087, 3516, 3980, 2622, 1282, 4610, 1204, 1295, 1251, 4527, 3020, 1577, 3560, 4076, 2262, 3704, 1417, 1502, 2220, 1862, 3235, 3931, 4990, 4306, 2694, 3088, 1749, 4372, 3861, 2729, 4939, 1057, 4433, 3228, 3049, 3961, 1982, 3768, 2158, 1866, 3171, 3259, 2096, 1333, 1693, 1730, 4668, 4454, 1889, 4963, 4680, 4572, 2941, 3373, 1575, 4263, 3808, 3672, 3405, 1436, 4360, 4964, 1388, 2374, 2073, 4624, 2128, 2642, 3112, 1084, 4164, 3412, 1384, 3359, 2345, 1932, 4157, 3271, 3549, 1186, 2243, 1573, 4072, 1114, 4329, 2978, 2254, 3029, 3642, 3163, 3731, 4202, 3868, 4564, 3506, 4264, 1776, 2375, 3981, 4358, 2815, 1961, 4071, 2656, 2640, 1859, 3809, 1567, 1083, 4565, 3239, 3126, 4136, 4098, 3847, 2675, 4899, 1472, 3915, 4363, 2491, 2098, 3654, 2311, 2187, 1537, 4036, 1078, 4044, 4282, 1777, 2854, 1904, 1490, 3465, 3554, 2788, 2942, 3759, 3395, 2527, 2139, 1141, 2202, 1841, 1069, 4667, 1441, 3889, 3411, 3056, 3562, 3414, 2756, 3852, 3154, 3793, 3887, 2233, 2879, 1461, 3272, 3263, 2937, 4096, 1087, 1312, 2876, 4456, 4291, 4294, 2999, 2485, 4825, 2984, 2947, 3798, 4513, 4404, 2469, 2039, 3396, 2744, 1186, 3266, 2872, 3861, 4016, 2056, 3739, 2596, 4804, 3057, 1400, 4612, 2058, 4358, 2387, 2117, 4557, 3150, 2008, 4316, 4289, 2076, 2938, 1852, 3737, 1312, 3237, 4697, 1638, 3799, 4739, 3640, 1378, 3955, 3399, 3855, 1486, 4527, 2840, 1612, 1484, 2893, 1315, 2201, 2862, 2527, 3563, 4516, 2342, 1558, 1963, 2036, 3895, 4127, 3521, 1932, 3578, 3571, 4097, 3882, 2445, 1386, 3236, 2223, 1163, 3914, 2847, 1253, 1796, 3381, 3880, 1409, 3766, 4850, 2885, 4196, 1280, 4031, 4911, 4109, 2064, 1237, 1189, 4095, 3974, 3827, 3430, 1384, 3495, 1598, 1875, 3201, 2315, 4315, 1999, 3137, 4223, 1615, 3886, 2067, 3672, 3762, 2152, 2997, 4402, 1925, 4285, 3385, 2452, 4254, 3521, 1184, 1415, 4359, 2677, 2902, 2495, 3806, 2065, 3470, 1044, 2699, 2738, 3643, 1335, 1274, 1864, 4207, 2807, 3006, 2931, 1769, 4307, 3597, 1719, 3407, 3232, 1266, 3356, 2581, 2809, 2909, 1429, 1283, 1067, 2316, 1418, 1825, 2380, 2468, 4461, 2624, 2851, 2952, 3533, 3004, 4366, 4211, 2161, 1623, 4490, 1511, 3595, 2551, 1110, 1202, 1189, 1746, 2188, 4980, 4380, 1553, 3075, 3935, 1275, 1930, 2081, 1284, 3952, 2687, 4019, 1059, 4366, 3633, 1282, 1884, 3995, 4383, 3565, 2953, 2092, 1689, 4037, 1733, 1624, 4208, 4411, 3949, 3989, 3363, 2277, 3504, 2879, 4099, 1571, 1755, 1676, 2834, 3609, 1162, 4068, 4051, 4503, 4368, 4198, 3615, 3443, 1375, 4805, 2145, 4276, 2124, 2747, 4881, 4440, 3458, 3101, 3747, 4320, 1154, 2636, 2576, 4332, 3205, 1247, 2502, 3884, 1047, 4367, 2951, 4722, 4180, 1463, 2920, 3515, 2122, 3841, 4523, 2208, 4209, 1124, 3093, 1108, 1402, 1328, 4686, 1677, 4699, 4642, 3677, 4532, 2461, 3497, 2104, 1155, 4235, 2917, 1650, 2737, 3862, 4514, 2980, 2695, 2163, 3203, 3226, 2979, 1640, 4499, 4094, 2646, 4858, 4429, 1596, 4095, 2364, 2744, 3433, 3524, 4941, 1102, 3161, 3063, 1176, 2144, 3527, 4240, 4712, 2707, 4579, 2248, 2395, 2660, 1164, 3988, 3230, 1948, 1559, 3958, 4319, 2594, 1045, 1210, 1329, 1226, 4034, 4917, 2713, 2619, 1979, 3573, 2846, 3576, 3044, 1481, 3490, 3204, 2420, 2218, 1281, 2544, 2018, 1474, 2481, 4821, 3413, 3795, 3610, 1538, 1542, 4063, 3442, 3229, 2745, 4419, 4566, 3487, 1182, 1949, 1205, 1156, 1907, 3976, 4234, 2866, 4277, 2020, 2526, 4021, 2177, 4274, 1527, 4650, 4026, 4354, 1798, 2004, 1324, 1767, 2503, 3028, 3816, 2545, 3733, 2335, 1779, 3749, 1279, 1386, 3582, 1748, 3341, 3758, 3414, 1219, 3431, 2759, 1431, 2461, 1635, 1188, 4506, 4061, 2180, 2682, 4792, 3427, 4070, 3906, 2979, 1315, 3064, 3170, 4240, 1661, 3342, 3033, 2030, 2716, 2044, 1410, 4562, 3445, 3817, 3299, 4651, 2519, 3531, 2819, 3167, 2435, 4237, 3256, 1489, 3315, 3775, 3510, 4089, 2477, 1432, 1714, 3137, 1631, 3047, 2632, 1647, 3207, 2777, 4547, 4459, 4071, 4192, 4220, 4335, 2877, 1793, 3730, 3699, 3269, 2380, 2703, 3238, 3555, 1373, 2645, 3900, 3708, 4362, 2229, 2452, 3753, 1691, 4190, 1814, 4073, 3007, 3102, 4603, 3973, 1894, 2242, 4548, 1227, 1863, 2601, 2890, 1832, 4345, 1402, 2210, 4778, 3471, 3195, 4684, 1476, 2806, 3885, 4929, 3889, 4332, 2172, 1828, 4775, 2700, 1860, 4394, 3153, 1934, 2824, 3522, 1118, 3950, 4882, 2350, 4450, 1815, 3584, 1583, 3768, 3369, 3956, 1805, 1811, 1354, 4004, 1739, 3420, 2717, 3296, 3667, 1599, 1748, 4098, 3756, 2672, 2146, 1297, 2302, 4038, 2972, 1846, 3856, 3527, 2930, 2825, 3024, 4512, 3956, 1787, 4286, 2968, 4661, 1601, 2100, 4244, 3978, 1436, 4072, 2597, 3530, 3769, 1271, 1614, 2338, 2055, 3165, 1959, 3646, 3297, 3556, 3567, 1332, 2461, 4199, 3525, 3930, 3038, 4082, 3571, 2921, 3037, 3587, 4490, 1468, 4344, 1089, 1633, 1432, 2169, 3670, 1109, 4485, 3460, 1499, 1539, 1902, 2378, 2889, 1891, 1854, 1182, 4442, 3759, 2675, 2366, 1547, 1040, 1876, 3439, 1256, 3870, 1439, 3150, 1353, 2821, 4378, 3959, 2497, 1144, 4915, 1586, 1531, 3453, 3627, 4633, 4270, 4464, 3422, 3832, 1427, 2124, 3496, 2195, 1039, 2635, 3796, 3409, 4382, 2917, 2082, 3765, 3871, 1067, 4222, 3536, 1813, 4284, 2203, 4048, 3942, 2529, 2701, 3062, 4598, 1445, 4314, 2260, 4673, 1998, 1757, 3679, 2052, 3168, 4205, 1681, 2141, 1237, 3510, 2026, 2770, 2317, 2391, 1691, 2312, 2824, 4462, 4080, 4315, 1586, 3786, 3313, 2700, 4719, 4189, 1111, 3711, 1583, 4082, 3722, 2162, 4555, 1088, 1794, 3253, 2258, 1686, 2046, 2873, 4795, 4057, 4049, 3005, 2361, 1623, 3492, 4632, 2983, 3854, 1768, 1028, 4026, 3449, 2949, 3169, 2026, 3516, 1556, 1145, 2769, 4053, 3967, 4287, 4278, 3049, 2739, 4170, 4084, 3451, 4859, 1826, 3898, 1391, 4509, 1523, 1214, 2259, 3622, 3712, 4776, 4510, 3880, 2077, 4182, 1854, 1321, 3016, 1928, 1890, 1197, 3346, 1286, 4444, 3951, 2993, 3194, 1074, 4522, 1843, 4413, 1990, 2755, 3221, 2836, 3123, 3480,


DynamicIDS v1.0.0 - Host Details - Netscape
File Edit View Go Bookmarks Tools Window Help
http://traffic4.cc.ncsu.edu/ids/ids.pl?ip=152.7.60.233&sample=current
Search
Mail Home Radio Netscape Search Bookmarks Red Hat Network Support Shop Products Training
Resource Usage Monitor v1.1
Resource Usage Monitor v1.1
DynamicIDS v1.0.0 - Host Details
4099, 3830, 4226, 1698, 1611, 1267, 2523, 2983, 1064, 2066, 1737, 4193, 2269, 2554, 3335, 1511, 3544, 4581, 3905, 3513, 1608, 4152, 4367, 1105, 3967, 2986, 4078, 1191, 2394,
4048, 4961, 1488, 3180, 1167, 1639, 4260, 2576, 3897, 2606, 1504, 1837, 4341, 4370, 4337, 4452, 4399, 3273, 2900, 3168, 1540, 1086, 4368, 1793, 2861, 1123, 3751, 1751, 1422,
1529, 2526, 4996, 4250, 1170, 1040, 1442, 3350, 1841, 2010, 3522, 3448, 2103, 3074, 4314, 3418, 1217, 3822, 2150, 2557, 2734, 1508, 2784, 4312, 3479, 2099, 3435, 1585, 2137,
2765, 4358, 3982, 3335, 1583, 1603, 4808, 3819, 3078, 2147, 2426, 1063, 1665, 3537, 4269, 2578, 4254, 3695, 3304, 4349, 3449, 1334, 2780, 3218, 4310, 3495, 2684, 4271, 2649,
3444, 1560, 1262, 2166, 4087, 3134, 2802, 4403, 2409, 1319, 2151, 3931, 4470, 4081, 1302, 1726, 2020, 2553, 3594, 3164, 2944, 3039, 3696, 2706, 4715, 3894, 2837, 1903, 4130,
1286, 3087, 3985, 2546, 1892, 3462, 1331, 1462, 1446, 2493, 4854, 4833, 2535, 1536, 2816, 3932, 3642, 2776, 3427, 1279, 1763, 2400, 3045, 1836, 2554, 3810, 1997, 3476, 3176,
2736, 1640, 1885, 1066, 4406, 1818, 3828, 4797, 2389, 1696, 4473, 4289, 3209, 2829, 1740, 3097, 1273, 3939, 1861, 2552, 2444, 2016, 4357, 2581, 3663, 2053, 1060, 1736, 1207,
4459, 2119, 2884, 2643, 3103, 1049, 3076, 4193, 1499, 2733, 2396, 4910, 2204, 4293, 4085, 2215, 1448, 3034, 3611, 4904, 3751, 1593, 1111, 2871, 3416, 3962, 3299, 3330, 2118,
1886, 4518, 4117, 3690, 3185, 1358, 4575, 4876, 2200, 2644, 1560, 4009, 2590, 3276, 1645, 4347, 3102, 2855, 1183, 1033, 3617, 1619, 1039, 2003, 3920, 4716, 2121, 1491, 2882,
2805, 4150, 1440, 4985, 2244, 2846, 3807, 1951, 1597, 4450, 4212, 1304, 4827, 3863, 4115, 4972, 3966, 1923, 1290, 4049, 4196, 1339, 2198, 4446, 2293, 1413, 2895, 3588, 1620,
4457, 3143, 4415, 4562, 2301, 4477, 1995, 3752, 4494, 2221, 4914, 1877, 4464, 1306, 2956, 2709, 1744, 1632, 1129, 4121, 3429, 1492, 1798, 3903, 2086, 4381, 1228, 1474, 4958,
1590, 1066, 3965, 2919, 4405, 4996, 1731, 4993, 2952, 4024, 4369, 3654, 1376, 3083, 3060, 4240, 3071, 4122, 2804, 1307, 2289, 3016, 1365, 4862, 3707, 4184, 3539, 4067, 4449,
4782, 3091, 2021, 2685, 4546, 2791, 1733, 3681, 4111, 3251, 3616, 3786, 4233, 1037, 4753, 2718, 3759, 4676, 3646, 2556, 4994, 3675, 4117, 4353, 3312, 2888, 4362, 4191, 1716,
3227, 4235, 2776, 4114, 2732, 4604, 4716, 1289, 3097, 1391, 2641, 4015, 1530, 2504, 4333, 3166, 2830, 2955, 2129, 3992, 1682, 2924, 2445, 4618, 1626, 4125, 3307, 4364, 3788,
1356, 4413, 4187, 3192, 3915, 1506, 1657, 1788, 2440, 4318, 1516, 2698, 3599, 1472, 3639, 4431, 3937, 4273, 2894, 2329, 1829, 3347, 1259, 3941, 1137, 1592, 4293, 3601, 4269,
1199, 4185, 3095, 1792, 3846, 1760, 2857, 4301, 4886, 2549, 3867, 1338, 1426, 1890, 1460, 3149, 1860, 3398, 4283, 4510, 4050, 3445, 1539, 1114, 3840, 1669, 1855, 1813, 1331,
3201, 3478, 4665, 4230, 1194, 3068, 2206, 3020, 3945, 2381, 2485, 3370, 4523, 3837, 1443, 2593, 3076, 3014, 1512, 2858, 2940, 1400, 2856, 1807, 4492, 1628, 2702, 2717, 1192,
1045, 3666, 3425, 3625, 1790, 3072, 2127, 4889, 4123, 4874, 3741, 2411, 1945, 2542, 1101, 4444, 1073, 2458, 1367, 3009, 1697, 2427, 4154, 2925, 4110, 1532, 2719, 4550, 4741,
1181, 1270, 4215, 2489, 2974, 4947, 3129, 4342, 2042, 2338, 1838, 1758, 1408, 1631, 3755, 2362, 1760, 3569, 3231, 4864, 2913, 1487, 3625, 4124, 3333, 3499, 2825, 1153, 3623,
1036, 4851, 1298, 3108, 1543, 2605, 1169, 4967, 4461, 1086, 1715, 4580, 3607, 4439, 4267, 3539, 3547, 1524, 3720, 2656, 4047, 1886, 3503, 1457, 1166, 3077, 4154, 2167, 3975,
2891, 4386, 3121, 4865, 2407, 1270, 1393, 2525, 1084, 1139, 3280, 2985, 4171, 3448, 1108, 4310, 3417, 1790, 4784, 1535, 2071, 2022, 1581, 3344, 2938, 3730, 3781, 4083, 1807,
3587, 4356, 2056, 3750, 2931, 3782, 2813, 3078, 4118, 2425, 1437, 3359, 1034, 2657, 1787, 3922, 3483, 2164, 4224, 3098, 2533, 1312, 2019, 3046, 3872, 4647, 4362, 3122, 3135,
3679, 2051, 2953, 1240, 3505, 1129, 4593, 1266, 3223, 1538, 4506, 3934, 3037, 3263, 3575, 4396, 3221, 4452, 4674, 4494, 2935, 1569, 2607, 1809, 4931, 1982, 1606, 3320, 2952,
2866, 1210, 3240, 2827, 3978, 3698, 3654, 2815, 3348, 2634, 3880, 1550, 1241, 4928, 2804, 3826, 4383, 2574, 1070, 1096, 3952, 3835, 1435, 2507, 1390, 1544, 3225, 3412, 4773,
3008, 3718, 3965, 4772, 1464, 4508, 2724, 3179, 3895, 2933, 4346, 3725, 3545, 1816, 3857, 3966, 2164, 2721, 3366, 1196, 1527, 4743, 2696, 2837, 3570, 3810, 1709, 3474, 4934,
2211, 3662, 2145, 4774, 1061, 1487, 3133, 4495, 3329, 2790, 1329, 2905, 4731, 1703, 2863, 3032, 2822, 3655, 4088, 1738, 1594, 2703, 3419, 3089, 1553, 1244, 1695, 2205, 2692,
4638, 2383, 3383, 2392, 3955, 3525, 3709, 1393, 3924, 4183, 2916, 2280, 1058, 3862, 4467, 4554, 2722, 3038, 3061, 1715, 4656, 2365, 3802, 3675, 2320, 4211, 1395, 4359, 3492,
3026, 4227, 3415, 1369, 3938, 4236, 3835, 3619, 2643, 3792, 2693, 3626, 2526, 1305, 4200, 3170, 1232, 3749, 1595, 1896, 3627, 3866, 2312, 1280, 2675, 4659, 1283, 1706, 4894,
3523, 3486, 4600, 2512, 1391, 2002, 2319, 4539, 1416, 4492, 1520, 4008, 2881, 4336, 3771, 2161, 1269, 1387, 1327, 1038, 1078, 2421, 3803, 4943, 3893, 3831, 3441, 1899, 3982,
1306, 1178, 2709, 3480, 4115, 1733, 1429, 2677, 2368, 4403, 1413, 2866, 2258, 1596, 1466, 3902, 1175, 3785, 4601, 2932, 3838, 3059, 1082, 4311, 3801, 2720, 4703, 3818, 2202,
2381, 3127, 1346, 3347, 1925, 3878, 2882, 4596, 3790, 1040, 4448, 1117, 4236, 1328, 2867, 1450, 3029, 3906, 2122, 4360, 3341, 4309, 1562, 2043, 4850, 4174, 4363, 2457, 3404,
4363, 4194, 2288, 1321, 1030, 2735, 3535, 2946, 3029, 3996, 3962, 2904, 4141, 1715, 3612, 3648, 1289, 1038, 1653, 1732, 3589, 1589, 3644, 1367, 1547, 3358, 2660, 1285, 4172,
2943, 2196, 3854, 3193, 3376, 3585, 4910, 3990, 1950, 2309, 1329, 3355, 3389, 2406, 1629, 4274, 1887, 1403, 4941, 4043, 4059, 4196, 4434, 3110, 3099, 1951, 2905, 1546, 1187,
1607, 3096, 1104, 4540, 3523, 2843, 3214, 4624, 3769, 2097, 3413, 3560, 3932, 1894, 3031, 4432, 3475, 2205, 4373, 3512, 3124, 2760, 2470, 1514, 1058, 1327, 3855, 1846, 4810,
4964, 3545, 3402, 4260, 2343, 1411, 2680, 1404, 4317, 2857, 1185, 4430, 4142, 4049, 3728, 4339, 3715, 1209, 1076, 1828, 3509, 2350, 3125, 1434, 1387, 3809, 2741, 2715, 4862,
2761, 3156, 3834, 1070, 2112, 1821, 2532, 1311, 1462, 3113, 1113, 1985, 4214, 2792, 1478, 2717, 2443, 1347, 3713, 4161, 2858, 3109, 4448, 4546, 1144, 1327, 1410, 3813, 1060,
3775, 2357, 3077, 2994, 3744, 3126, 3882, 1126, 1116, 2925, 4950, 3771, 3729, 2043, 2716, 4661, 1986, 2217, 2856, 1873, 2988, 1229, 2873, 4075, 1454, 1507, 4486, 3491, 4122,
1577, 1412, 1208, 2162, 3985, 3680, 2725, 3710, 3569, 4012, 1759, 4615, 2528, 4519,
Allowed: 0
Alerts
Possible Trojan or Virus Detected
Host communicating on large number of ports
Traffic flows exceed threshold (Actual: 14191; Threshold: 2069.67)
Intrusions
Possible Port Scan Attacker
Traffic Statistics Network Analysis (Outgoing / Source)
View Traffic Details
Category | Host Data | Avg. Data | Host_Index
flows | 10249 | 16.3911609498681 | 624.276027204314
Status: UP
Role: Client
Traffic Statistics Self Analysis (Outgoing / Source)
View Traffic Details
Category | Host Data | Last Data | Host_Index
flows | 10249 | 9282 | 0.104180133591898
Document: Done (21.947 secs)

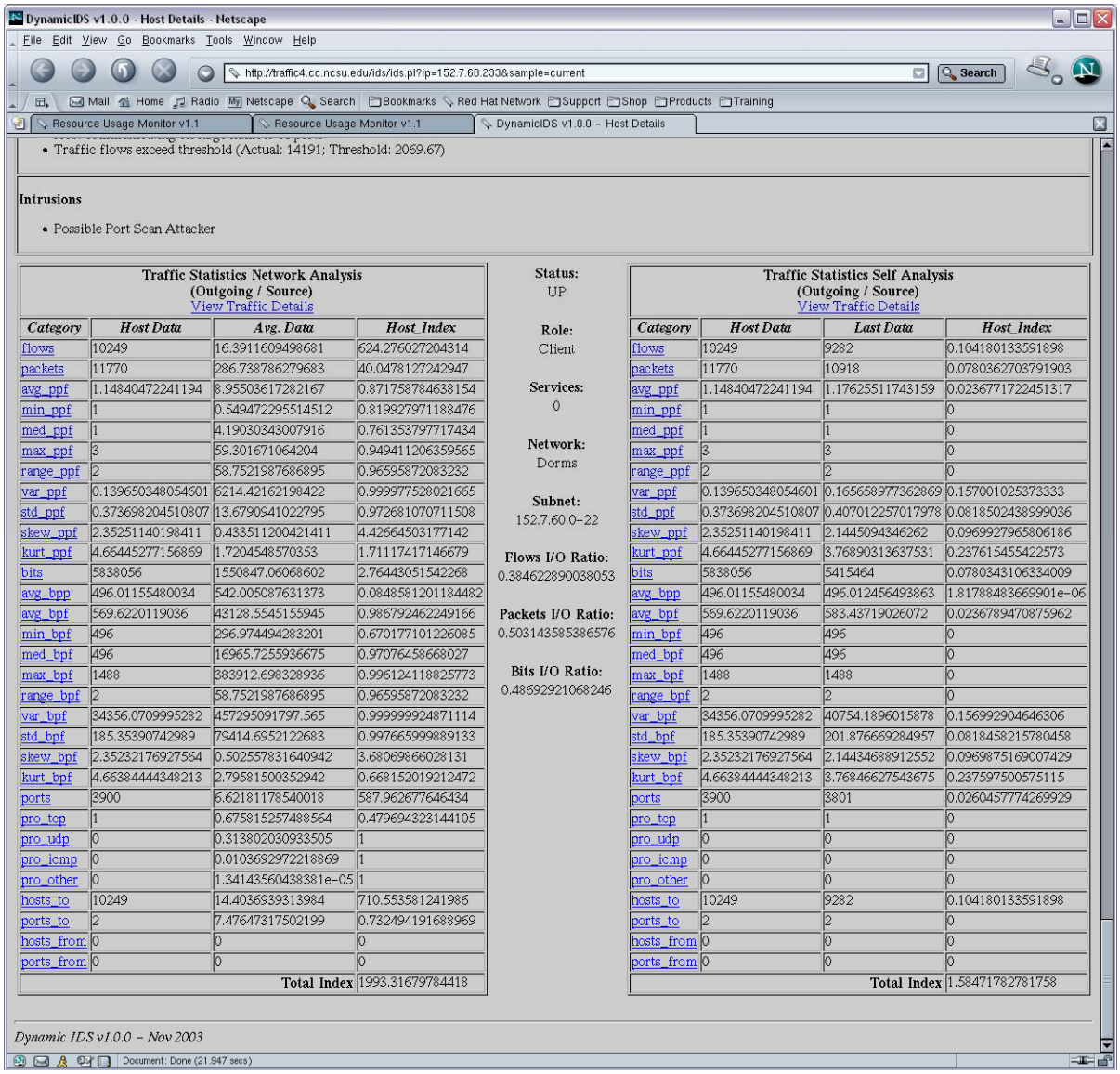


Figure 5.16 – DIDS Host Details (#2) – 152.7.60.233 as Source (Outbound)

The first obvious observation for this particular host is the large number of communicating ports it had. From the top of the page, the host is shown to have 39,000 index numbers from ports (over 60% of the total). The only allowed (registered) service is 0 – in other words, this host is a pure client. Where every unregistered communicating port generates an index of 10, this means that the host

had 3,900 unregistered communicating ports[18]. This can also be seen from the network analysis table (third portion of the figure), where it shows 3,900 for the total number of ports. Obviously, 3,900 ports connections that this host was using were excessive by any standards and this was a genuine.

In addition, this host also had several alerts and an intrusion warning (second portion of the figure). The first alert indicates possible Trojan and virus. In fact, all the communicating ports were scanned against a Trojan/virus database for matching ports, and each matching port is shown in red as a link. More information can be obtained by following that link (e.g. Figure 5.17 and Figure 5.18). The second alert indicates that the host is communicating on large number of ports, which we have already seen. And the third and last alert indicates the host has exceeded the flows threshold[19]. Together the alerts put up an index of 15920, calculated based on the algorithms discussed in section 3.8.2.2.

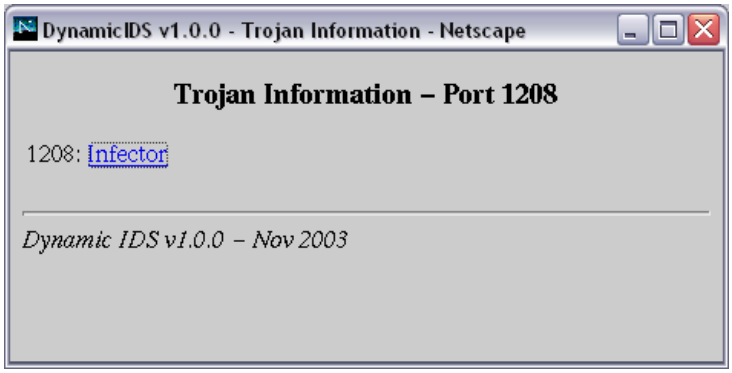


Figure 5.17 – Example of Trojan/Virus Port Information (By following the port link from host details page)

[18] Please refer to section 3.8.2.1.
[19] For more information on thresholds, please refer to section 3.4 and 3.8.2.2.

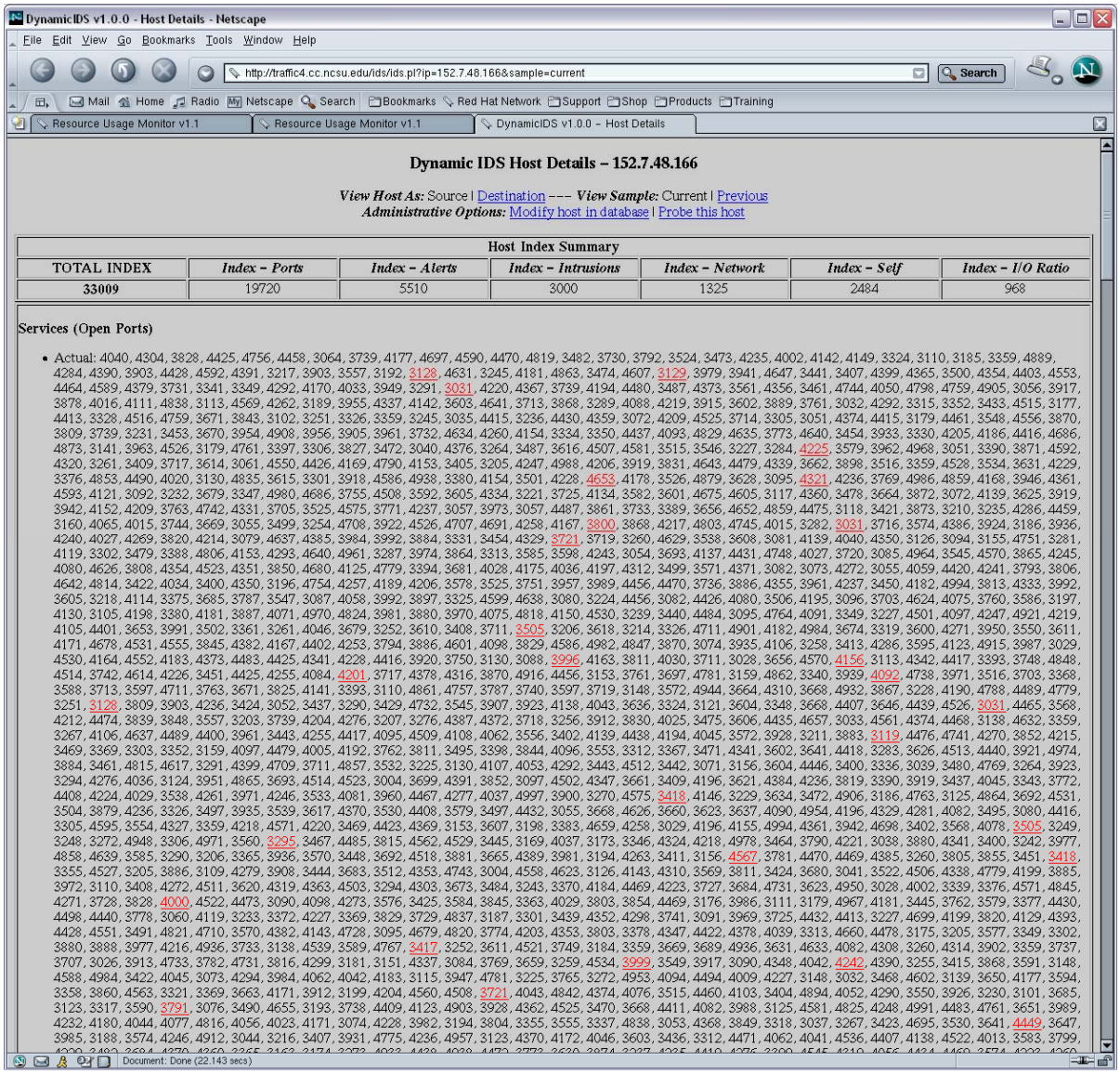
Figure 5.18 – Example of Trojan/Virus Details
(By following the port link from Figure 5.17)

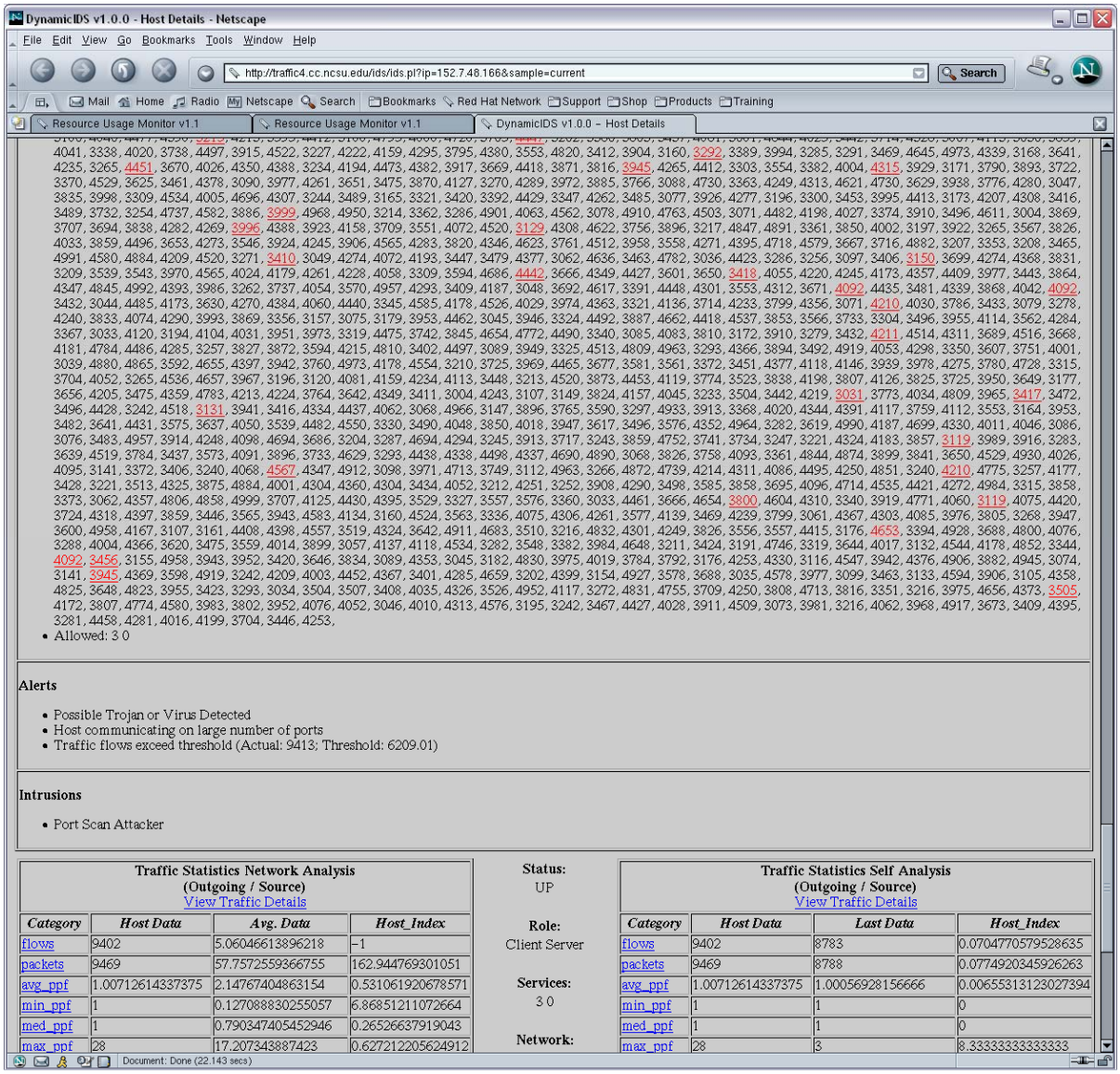
With such an excessive number of communicating ports and a large number of ports that matches the port used by Trojans and viruses, this host is most definitely infected.

Further, the intrusion warning shows that the host is a possible port scan attacker. This can be confirmed by further examining the flow ratio, which shows almost 3 times more outbound flows than inbound. This indicates a possible scan or DoS attacker. To distinguish, the unique number of target hosts and target ports that the host is generating traffic to is examined. From near the end of the network analysis table, one can see that the host is sending packets to 10,259 unique hosts but to only 2 unique ports. This is the obvious behavior or a port scan. (More discussions on this can be found in section 3.7 and 4.3.5).

RUM, on the other hand, ranked this host very low on its load and packet emitters and receivers list, but put it up as number one on flow emitters and receivers list (again, a classic characteristics of port scanners – high flows and low packets/load). However, assume the host had not generated such a large number of flows, but kept the same behavior as a port scanner, RUM might have given this host a low ranking on flow emitters and receivers list too, but DIDS would have still detected it because of the large number of communicating ports and the flow input/output ratio.

The third host on DIDS's monitoring list presents a very similar example to the second host – again a port scanner with low amount of traffic packets and load but high number of ports, flows, and flow I/O ratio. See Figure 5.19 below.


DynamicIDS v1.0.0 - Host Details - Netscape
File Edit View Go Bookmarks Tools Window Help
http://traffic4.cc.ncsu.edu/ids/ids.pl?ip=152.7.48.166&sample=current
Search
Mail Home Radio Netscape Search Bookmarks Red Hat Network Support Shop Products Training
Resource Usage Monitor v1.1
Resource Usage Monitor v1.1
DynamicIDS v1.0.0 - Host Details
Dynamic IDS Host Details – 152.7.48.166
View Host As: Source | Destination --- View Sample: Current | Previous
Administrative Options: Modify host in database | Probe this host
Host Index Summary
TOTAL INDEX
Index – Ports
Index – Alerts
Index – Intrusions
Index – Network
Index – Self
Index – I/O Ratio
33009
19720
5510
3000
1325
2484
968
Services (Open Ports)
Actual: 4040, 4304, 3828, 4425, 4756, 4458, 3064, 3739, 4177, 4697, 4590, 4470, 4819, 3482, 3730, 3792, 3524, 3473, 4235, 4002, 4142, 4149, 3324, 3110, 3185, 3359, 4889, 4284, 4390, 3903, 4428, 4592, 4391, 3217, 3903, 3557, 3192, 3128, 4631, 3245, 4181, 4863, 3474, 4607, 3129, 3979, 3941, 4647, 3441, 3407, 4399, 4365, 3500, 4354, 4403, 4553, 4464, 4589, 4379, 3731, 3341, 3349, 4292, 4170, 4033, 3949, 3291, 3031, 4220, 4367, 3739, 4194, 4480, 3487, 4373, 3561, 4356, 3461, 4744, 4050, 4798, 4759, 4905, 3056, 3917, 3878, 4016, 4111, 4838, 3113, 4569, 4262, 3189, 3955, 4337, 4142, 3603, 4641, 3713, 3868, 3289, 4088, 4219, 3915, 3602, 3889, 3761, 3032, 4292, 3315, 3352, 3433, 4515, 3177, 4413, 3328, 4516, 4759, 3671, 3843, 3102, 3251, 3326, 3359, 3245, 3035, 4415, 3236, 4430, 4359, 3072, 4209, 4525, 3714, 3305, 3051, 4374, 4415, 3179, 4461, 3548, 4556, 3870, 3809, 3739, 3231, 3453, 3670, 3954, 4908, 3956, 3905, 3961, 3732, 4634, 4260, 4154, 3334, 3350, 4437, 4093, 4829, 4635, 3773, 4640, 3454, 3933, 3330, 4205, 4186, 4416, 4686, 4873, 3141, 3963, 4526, 3179, 4761, 3397, 3306, 3827, 3472, 3040, 4376, 3264, 3487, 3616, 4507, 4581, 3515, 3546, 3227, 3284, 4225, 3579, 3962, 4968, 3051, 3390, 3871, 4592, 4320, 3261, 3409, 3717, 3614, 3061, 4550, 4426, 4169, 4790, 4153, 3405, 3205, 4247, 4988, 4206, 3919, 3831, 4643, 4479, 4339, 3662, 3898, 3516, 3359, 4528, 3534, 3631, 4229, 3376, 4853, 4490, 4020, 3130, 4835, 3615, 3301, 3918, 4586, 4938, 3380, 4154, 3501, 4228, 4653, 4178, 3526, 4879, 3628, 3095, 4321, 4236, 3769, 4986, 4859, 4168, 3946, 4361, 4593, 4121, 3092, 3232, 3679, 3347, 4980, 4686, 3755, 4508, 3592, 3605, 4334, 3221, 3725, 4134, 3582, 3601, 4675, 4605, 3117, 4360, 3478, 3664, 3872, 3072, 4139, 3625, 3919, 3942, 4152, 4209, 3763, 4742, 4331, 3705, 3525, 4575, 3771, 4237, 3057, 3973, 3057, 4487, 3861, 3733, 3389, 3656, 4652, 4859, 4475, 3118, 3421, 3873, 3210, 3235, 4286, 4459, 3160, 4065, 4015, 3744, 3669, 3055, 3499, 3254, 4708, 3922, 4526, 4707, 4691, 4258, 4167, 3800, 3868, 4217, 4803, 4745, 4015, 3282, 3031, 3716, 3574, 4386, 3924, 3186, 3936, 4240, 4027, 4269, 3820, 4214, 3079, 4637, 4385, 3984, 3992, 3884, 3331, 3454, 4329, 3721, 3719, 3260, 4629, 3538, 3608, 3081, 4139, 4040, 4350, 3126, 3094, 3155, 4751, 3281, 4119, 3302, 3479, 3388, 4806, 4153, 4293, 4640, 4961, 3287, 3974, 3864, 3313, 3585, 3598, 4243, 3054, 3693, 4137, 4431, 4748, 4027, 3720, 3085, 4964, 3545, 4570, 3865, 4245, 4080, 4626, 3808, 4354, 4523, 4351, 3850, 4680, 4125, 4779, 3394, 3681, 4028, 4175, 4036, 4197, 4312, 3499, 3571, 4371, 3082, 3073, 4272, 3055, 4059, 4420, 4241, 3793, 3806, 4642, 4814, 3422, 4034, 3400, 4350, 3196, 4754, 4257, 4189, 4206, 3578, 3525, 3751, 3957, 3989, 4456, 4470, 3736, 3886, 4355, 3961, 4237, 3450, 4182, 4994, 3813, 4333, 3992, 3605, 3218, 4114, 3375, 3685, 3787, 3547, 3087, 4058, 3992, 3897, 3325, 4599, 4638, 3080, 3224, 4456, 3082, 4426, 4080, 3506, 4195, 3096, 3703, 4624, 4075, 3760, 3586, 3197, 4130, 3105, 4198, 3380, 4181, 3887, 4071, 4970, 4824, 3981, 3880, 3970, 4075, 4818, 4150, 4530, 3239, 3440, 4484, 3095, 4764, 4091, 3349, 3227, 4501, 4097, 4247, 4921, 4219, 4105, 4401, 3653, 3991, 3502, 3361, 3261, 4046, 3679, 3252, 3610, 3408, 3711, 3505, 3206, 3618, 3214, 3326, 4711, 4901, 4182, 4984, 3674, 3319, 3600, 4271, 3950, 3550, 3611, 4171, 4678, 4531, 4555, 3845, 4382, 4167, 4402, 4253, 3794, 3886, 4601, 4098, 3829, 4586, 4982, 4847, 3870, 3074, 3935, 4106, 3258, 3413, 4286, 3595, 4123, 4915, 3987, 3029, 4530, 4164, 4552, 4183, 4373, 4483, 4425, 4341, 4228, 4416, 3920, 3750, 3130, 3088, 3996, 4163, 3811, 4030, 3711, 3028, 3656, 4570, 4156, 3113, 4342, 4417, 3393, 3748, 4848, 4514, 3742, 4614, 4226, 3451, 4425, 4255, 4084, 4201, 3717, 4378, 4316, 3870, 4916, 4456, 3153, 3761, 3697, 4781, 3159, 4862, 3340, 3939, 4092, 4738, 3971, 3516, 3703, 3368, 3588, 3713, 3597, 4711, 3763, 3671, 3825, 4141, 3393, 3110, 4861, 4757, 3787, 3740, 3597, 3719, 3148, 3572, 4944, 3664, 4310, 3668, 4932, 3867, 3228, 4190, 4788, 4489, 4779, 3251, 3128, 3809, 3903, 4236, 3424, 3052, 3437, 3290, 3429, 4732, 3545, 3907, 3923, 4138, 4043, 3636, 3324, 3121, 3604, 3348, 3668, 4407, 3646, 4439, 4526, 3031, 4465, 3568, 4212, 4474, 3839, 3848, 3557, 3203, 3739, 4204, 4276, 3207, 3276, 4387, 4372, 3718, 3256, 3912, 3830, 4025, 3475, 3606, 4435, 4657, 3033, 4561, 4374, 4468, 3138, 4632, 3359, 3267, 4106, 4637, 4489, 4400, 3961, 3443, 4255, 4417, 4095, 4509, 4108, 4062, 3556, 3402, 4139, 4438, 4194, 4045, 3572, 3928, 3211, 3883, 3119, 4476, 4741, 4270, 3852, 4215, 3469, 3369, 3303, 3352, 3159, 4097, 4479, 4005, 4192, 3762, 3811, 3495, 3398, 3844, 4096, 3553, 3312, 3367, 3471, 4341, 3602, 3641, 4418, 3283, 3626, 4513, 4440, 3921, 4974, 3884, 3461, 4815, 4617, 3291, 4399, 4709, 3711, 4857, 3532, 3225, 3130, 4107, 4053, 4292, 3443, 4512, 3442, 3071, 3156, 3604, 4446, 3400, 3336, 3039, 3480, 4769, 3264, 3923, 3294, 4276, 4036, 3124, 3951, 4865, 3693, 4514, 4523, 3004, 3699, 4391, 3852, 3097, 4502, 4347, 3661, 3409, 4196, 3621, 4384, 4236, 3819, 3390, 3919, 3437, 4045, 3343, 3772, 4408, 4224, 4029, 3538, 4261, 3971, 4246, 3533, 4081, 3960, 4467, 4277, 4037, 4997, 3900, 3270, 4575, 3418, 4146, 3229, 3634, 3472, 4906, 3186, 4763, 3125, 4864, 3692, 4531, 3504, 3879, 4236, 3326, 3497, 3935, 3539, 3617, 4370, 3530, 4408, 3579, 3497, 4432, 3055, 3668, 4626, 3660, 3623, 3637, 4090, 4954, 4196, 4329, 4281, 4082, 3495, 3080, 4416, 3305, 4595, 3554, 4327, 3359, 4218, 4571, 4220, 3469, 4423, 4369, 3153, 3607, 3198, 3383, 4659, 4258, 3029, 4196, 4155, 4994, 4361, 3942, 4698, 3402, 3568, 4078, 3505, 3249, 3248, 3272, 4948, 3306, 4971, 3560, 3295, 3467, 4485, 3815, 4562, 4529, 3445, 3169, 4037, 3173, 3346, 4324, 4218, 4978, 3464, 3790, 4221, 3038, 3880, 4341, 3400, 3242, 3977, 4858, 4639, 3585, 3290, 3206, 3365, 3936, 3570, 3448, 3692, 4518, 3881, 3665, 4389, 3981, 3194, 4263, 3411, 3156, 4567, 3781, 4470, 4469, 4385, 3260, 3805, 3855, 3451, 3418, 3355, 4527, 3205, 3886, 3109, 4279, 3908, 3444, 3683, 3512, 4353, 4743, 3004, 4558, 4623, 3126, 4143, 4310, 3569, 3811, 3424, 3680, 3041, 3522, 4506, 4338, 4779, 4199, 3885, 3972, 3110, 3408, 4272, 4511, 3620, 4319, 4363, 4503, 3294, 4303, 3673, 3484, 3243, 3370, 4184, 4469, 4223, 3727, 3684, 4731, 3623, 4950, 3028, 4002, 3339, 3376, 4571, 4845, 4271, 3728, 3828, 4000, 4522, 4473, 3090, 4098, 4273, 3576, 3425, 3584, 3845, 3363, 4029, 3803, 3854, 4469, 3176, 3986, 3111, 3179, 4967, 4181, 3445, 3762, 3579, 3377, 4430, 4498, 4440, 3778, 3060, 4119, 3233, 3372, 4227, 3369, 3829, 3729, 4837, 3187, 3301, 3439, 4352, 4298, 3741, 3091, 3969, 3725, 4432, 4413, 3227, 4699, 4199, 3820, 4129, 4393, 4428, 4551, 3491, 4821, 4710, 3570, 4382, 4143, 4728, 3095, 4679, 4820, 3774, 4203, 4353, 3803, 3378, 4347, 4422, 4378, 4039, 3313, 4660, 4478, 3175, 3205, 3577, 3349, 3302, 3880, 3888, 3977, 4216, 4936, 3733, 3138, 4539, 3589, 4767, 3417, 3252, 3611, 4521, 3749, 3184, 3359, 3669, 3689, 4936, 3631, 4633, 4082, 4308, 3260, 4314, 3902, 3359, 3737, 3707, 3026, 3913, 4733, 3782, 4731, 3816, 4299, 3181, 3151, 4337, 3084, 3769, 3659, 3259, 4534, 3999, 3549, 3917, 3090, 4348, 4042, 4242, 4390, 3255, 3415, 3868, 3591, 3148, 4588, 4984, 3422, 4045, 3073, 4294, 3984, 4062, 4042, 4183, 3115, 3947, 4781, 3225, 3765, 3272, 4953, 4094, 4494, 4009, 4227, 3148, 3032, 3468, 4602, 3139, 3650, 4177, 3594, 3358, 3860, 4563, 3321, 3369, 3663, 4171, 3912, 3199, 4204, 4560, 4508, 3721, 4043, 4842, 4374, 4076, 3515, 4460, 4103, 3404, 4894, 4052, 4290, 3550, 3926, 3230, 3101, 3685, 3123, 3317, 3590, 3791, 3076, 3490, 4655, 3193, 3738, 4409, 4123, 4903, 3928, 4362, 4525, 3470, 3668, 4411, 4082, 3988, 3125, 4581, 4825, 4248, 4991, 4483, 4761, 3651, 3989, 4232, 4180, 4044, 4077, 4816, 4056, 4023, 4171, 3074, 4228, 3982, 3194, 3804, 3355, 3555, 3337, 4838, 3053, 4368, 3849, 3318, 3037, 3267, 3423, 4695, 3530, 3641, 4449, 3647, 3985, 3188, 3574, 4246, 4912, 3044, 3216, 3407, 3931, 4775, 4236, 4957, 3123, 4370, 4172, 4046, 3603, 3436, 3312, 4471, 4062, 4041, 4536, 4407, 4138, 4522, 4013, 3583, 3799,
Document: Done (22.143 secs)


DynamicIDS v1.0.0 - Host Details - Netscape
File Edit View Go Bookmarks Tools Window Help
http://traffic4.cc.ncsu.edu/ids/ids.pl?ip=152.7.48.166&sample=current
Search
Mail Home Radio Netscape Search Bookmarks Red Hat Network Support Shop Products Training
Resource Usage Monitor v1.1
Resource Usage Monitor v1.1
DynamicIDS v1.0.0 - Host Details
4041, 3338, 4020, 3738, 4497, 3915, 4522, 3227, 4222, 4159, 4295, 3795, 4380, 3553, 4820, 3412, 3904, 3160, 3292, 3389, 3994, 3285, 3291, 3469, 4645, 4973, 4339, 3168, 3641, 4235, 3265, 4451, 3670, 4026, 4350, 4388, 3234, 4194, 4473, 4382, 3917, 3669, 4418, 3871, 3816, 3945, 4265, 4412, 3303, 3554, 3382, 4004, 4315, 3929, 3171, 3790, 3893, 3722, 3370, 4529, 3625, 3461, 4378, 3090, 3977, 4261, 3651, 3475, 3870, 4127, 3270, 4289, 3972, 3885, 3766, 3088, 4730, 3363, 4249, 4313, 4621, 4730, 3629, 3938, 3776, 4280, 3047, 3835, 3998, 3309, 4534, 4005, 4696, 4307, 3244, 3489, 3165, 3321, 3420, 3392, 4429, 3347, 4262, 3485, 3077, 3926, 4277, 3196, 3300, 3453, 3995, 4413, 3173, 4207, 4308, 3416, 3489, 3732, 3254, 4737, 4582, 3886, 3999, 4968, 4950, 3214, 3362, 3286, 4901, 4063, 4562, 3078, 4910, 4763, 4503, 3071, 4482, 4198, 4027, 3374, 3910, 3496, 4611, 3004, 3869, 3707, 3694, 3838, 4282, 4269, 3996, 4388, 3923, 4158, 3709, 3551, 4072, 4520, 3129, 4308, 4622, 3756, 3896, 3217, 4847, 4891, 3361, 3850, 4002, 3197, 3922, 3265, 3567, 3826, 4033, 3859, 4496, 3653, 4273, 3546, 3924, 4245, 3906, 4565, 4283, 3820, 4346, 4623, 3761, 4512, 3958, 3558, 4271, 4395, 4718, 4579, 3667, 3716, 4882, 3207, 3353, 3208, 3465, 4991, 4580, 4884, 4209, 4520, 3271, 3410, 3049, 4274, 4072, 4193, 3447, 3479, 4377, 3062, 4636, 3463, 4782, 3036, 4423, 3286, 3256, 3097, 3406, 3150, 3699, 4274, 4368, 3831, 3209, 3539, 3543, 3970, 4565, 4024, 4179, 4261, 4228, 4058, 3309, 3594, 4686, 4442, 3666, 4349, 4427, 3601, 3650, 3418, 4055, 4220, 4245, 4173, 4357, 4409, 3977, 3443, 3864, 4347, 4845, 4992, 4393, 3986, 3262, 3737, 4054, 3570, 4957, 4293, 3409, 4187, 3048, 3692, 4617, 3391, 4448, 4301, 3553, 4312, 3671, 4092, 4435, 3481, 4339, 3868, 4042, 4092, 3432, 3044, 4485, 4173, 3630, 4270, 4384, 4060, 4440, 3345, 4585, 4178, 4526, 4029, 3974, 4363, 3321, 4136, 3714, 4233, 3799, 4356, 3071, 4210, 4030, 3786, 3433, 3079, 3278, 4240, 3833, 4074, 4290, 3993, 3869, 3356, 3157, 3075, 3179, 3953, 4462, 3045, 3946, 3324, 4492, 3887, 4662, 4418, 4537, 3853, 3566, 3733, 3304, 3496, 3955, 4114, 3562, 4284, 3367, 3033, 4120, 3194, 4104, 4031, 3951, 3973, 3319, 4475, 3742, 3845, 4654, 4772, 4490, 3340, 3085, 4083, 3810, 3172, 3910, 3279, 3432, 4211, 4514, 4311, 3689, 4516, 3668, 4181, 4784, 4486, 4285, 3257, 3827, 3872, 3594, 4215, 4810, 3402, 4497, 3089, 3949, 3325, 4513, 4809, 4963, 3293, 4366, 3894, 3492, 4919, 4053, 4298, 3350, 3607, 3751, 4001, 3039, 4880, 4865, 3592, 4655, 4397, 3942, 3760, 4973, 4178, 4554, 3210, 3725, 3969, 4465, 3677, 3581, 3561, 3372, 3451, 4377, 4118, 4146, 3939, 3978, 4275, 3780, 4728, 3315, 3704, 4052, 3265, 4536, 4657, 3967, 3196, 3120, 4081, 4159, 4234, 4113, 3448, 3213, 4520, 3873, 4453, 4119, 3774, 3523, 3838, 4198, 3807, 4126, 3825, 3725, 3950, 3649, 3177, 3656, 4205, 3475, 4359, 4783, 4213, 4224, 3764, 3642, 4349, 3411, 3004, 4243, 3107, 3149, 3824, 4157, 4045, 3233, 3504, 3442, 4219, 3031, 3773, 4034, 4809, 3965, 3417, 3472, 3496, 4428, 3242, 4518, 3131, 3941, 3416, 4334, 4437, 4062, 3068, 4966, 3147, 3896, 3765, 3590, 3297, 4933, 3913, 3368, 4020, 4344, 4391, 4117, 3759, 4112, 3553, 3164, 3953, 3482, 3641, 4431, 3575, 3637, 4050, 3539, 4482, 4550, 3330, 3490, 4048, 3850, 4018, 3947, 3617, 3496, 3576, 4352, 4964, 3282, 3619, 4990, 4187, 4699, 4330, 4011, 4046, 3086, 3076, 3483, 4957, 3914, 4248, 4098, 4694, 3686, 3204, 3287, 4694, 4294, 3245, 3913, 3717, 3243, 3859, 4752, 3741, 3734, 3247, 3221, 4324, 4183, 3857, 3119, 3989, 3916, 3283, 3639, 4519, 3784, 3437, 3573, 4091, 3896, 3733, 4629, 3293, 4438, 4338, 4498, 4337, 4690, 4890, 3068, 3826, 3758, 4093, 3361, 4844, 4874, 3899, 3841, 3650, 4529, 4930, 4026, 4095, 3141, 3372, 3406, 3240, 4068, 4567, 4347, 4912, 3098, 3971, 4713, 3749, 3112, 4963, 3266, 4872, 4739, 4214, 4311, 4086, 4495, 4250, 4851, 3240, 4210, 4775, 3257, 4177, 3428, 3221, 3513, 4325, 3875, 4884, 4001, 4304, 4360, 4304, 3434, 4052, 3212, 4251, 3252, 3908, 4290, 3498, 3585, 3858, 3695, 4096, 4714, 4535, 4421, 4272, 4984, 3315, 3858, 3373, 3062, 4357, 4806, 4858, 4999, 3707, 4125, 4430, 4395, 3529, 3327, 3557, 3576, 3360, 3033, 4461, 3666, 4654, 3800, 4604, 4310, 3340, 3919, 4771, 4060, 3119, 4075, 4420, 3724, 4318, 4397, 3859, 3446, 3565, 3943, 4583, 4134, 3160, 4524, 3563, 3336, 4075, 4306, 4261, 3577, 4139, 3469, 4239, 3799, 3061, 4367, 4303, 4085, 3976, 3805, 3268, 3947, 3600, 4958, 4167, 3107, 3161, 4408, 4398, 4557, 3519, 4324, 3642, 4911, 4683, 3510, 3216, 4832, 4301, 4249, 3826, 3556, 3557, 4415, 3176, 4653, 3394, 4928, 3688, 4800, 4076, 3288, 4004, 4366, 3620, 3475, 3559, 4014, 3899, 3057, 4137, 4118, 4534, 3282, 3548, 3382, 3984, 4648, 3211, 3424, 3191, 4746, 3319, 3644, 4017, 3132, 4544, 4178, 4852, 3344, 4092, 3456, 3155, 4958, 3943, 3952, 3420, 3646, 3834, 3089, 4353, 3045, 3182, 4830, 3975, 4019, 3784, 3792, 3176, 4253, 4330, 3116, 4547, 3942, 4376, 4906, 3882, 4945, 3074, 3141, 3945, 4369, 3598, 4919, 3242, 4209, 4003, 4452, 4367, 3401, 4285, 4659, 3202, 4399, 3154, 4927, 3578, 3688, 3035, 4578, 3977, 3099, 3463, 3133, 4594, 3906, 3105, 4358, 4825, 3648, 4823, 3955, 3423, 3293, 3034, 3504, 3507, 3408, 4035, 4326, 3526, 4952, 4117, 3272, 4831, 4755, 3709, 4250, 3808, 4713, 3816, 3351, 3216, 3975, 4656, 4373, 3505, 4172, 3807, 4774, 4580, 3983, 3802, 3952, 4076, 4052, 3046, 4010, 4313, 4576, 3195, 3242, 3467, 4427, 4028, 3911, 4509, 3073, 3981, 3216, 4062, 3968, 4917, 3673, 3409, 4395, 3281, 4458, 4281, 4016, 4199, 3704, 3446, 4253,
Allowed: 3 0
Alerts
Possible Trojan or Virus Detected
Host communicating on large number of ports
Traffic flows exceed threshold (Actual: 9413; Threshold: 6209.01)
Intrusions
Port Scan Attacker
Traffic Statistics Network Analysis (Outgoing / Source)
View Traffic Details
Category | Host Data | Avg. Data | Host_Index
flows | 9402 | 5.06046613896218 | -1
packets | 9469 | 57.7572559366755 | 162.944769301051
avg_ppf | 1.00712614337375 | 2.14767404863154 | 0.531061920678571
min_ppf | 1 | 0.127088830255057 | 6.86851211072664
med_ppf | 1 | 0.790347405452946 | 0.26526637919043
max_ppf | 28 | 17.207343887423 | 0.627212205624912
Status:
UP
Role:
Client Server
Services:
3 0
Network:
Traffic Statistics Self Analysis (Outgoing / Source)
View Traffic Details
Category | Host Data | Last Data | Host_Index
flows | 9402 | 8783 | 0.0704770579528635
packets | 9469 | 8788 | 0.0774920345926263
avg_ppf | 1.00712614337375 | 1.00056928156666 | 0.00655313123027394
min_ppf | 1 | 1 | 0
med_ppf | 1 | 1 | 0
max_ppf | 28 | 3 | 8.33333333333333
Document: Done (22.143 secs)

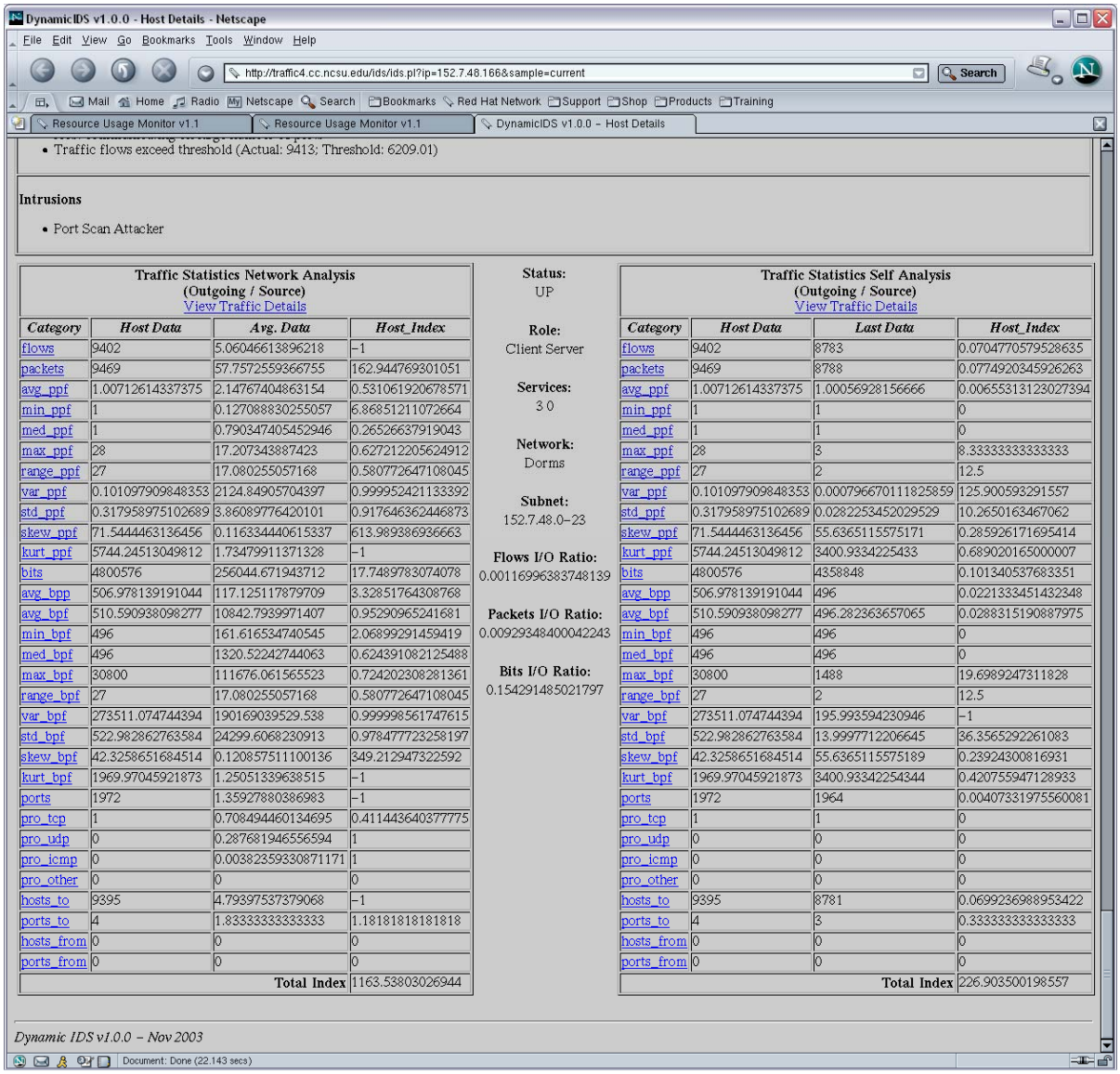


Figure 5.19 – DIDS Host Details (#3) – 152.7.48.166 as Source (Outbound)

Another interesting example to look at is the 12th host on the DIDS's ranking list. This particular host had a DOWN status while most others had an UP. The screenshot for the host is displayed below as Figure 5.20.

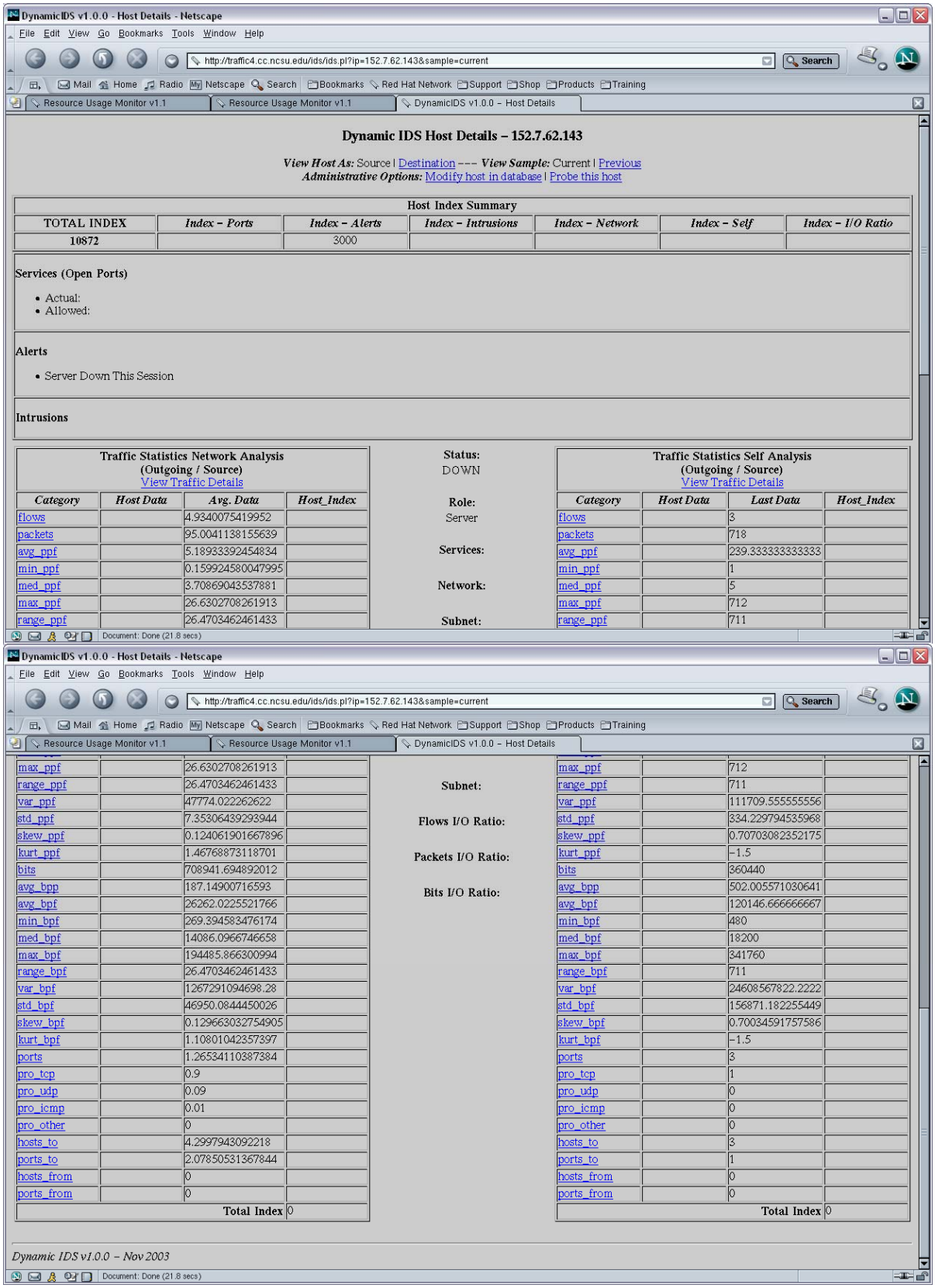


Figure 5.20 – DIDS Host Details (#12) – 152.7.62.143 (DOWN Host)

The DOWN status simply means that this particular host was present in the last sample, but was not present in the current one. It is important to include down hosts in the analysis because it helps in detection of intermittent effects such as network failures, computation of network reliability and availability, and detection of victims affected by attacks. From the host details page, one can clearly see that the host has a total index of 10,872. However, the only sub-category with an index value is the index from alerts (which has 3,000). And from the alerts message table we see "Server went down this session). The question is where the rest of the index came from. In fact, the rest of it (7,872) was carried over from the previous session (Figure 5.21). By including the index value from the previous sample instead of simply assigning a static value, the IDS system clearly distinguish the level of importance among all the different hosts that went down in the current sample period, and the placement of each individual down host became dynamic due to previous behavior. A more suspicious host that went down deserves more attention from the system administrator.

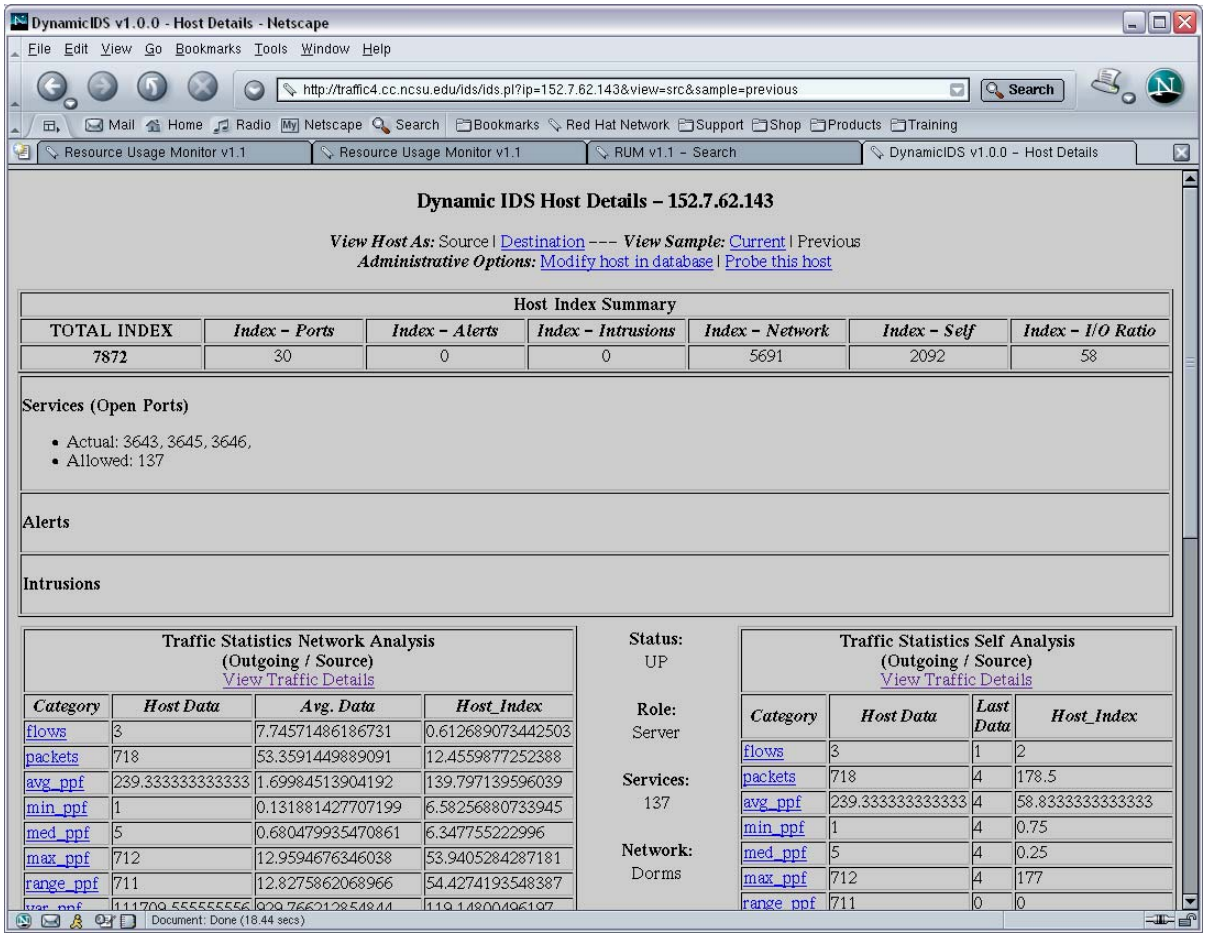
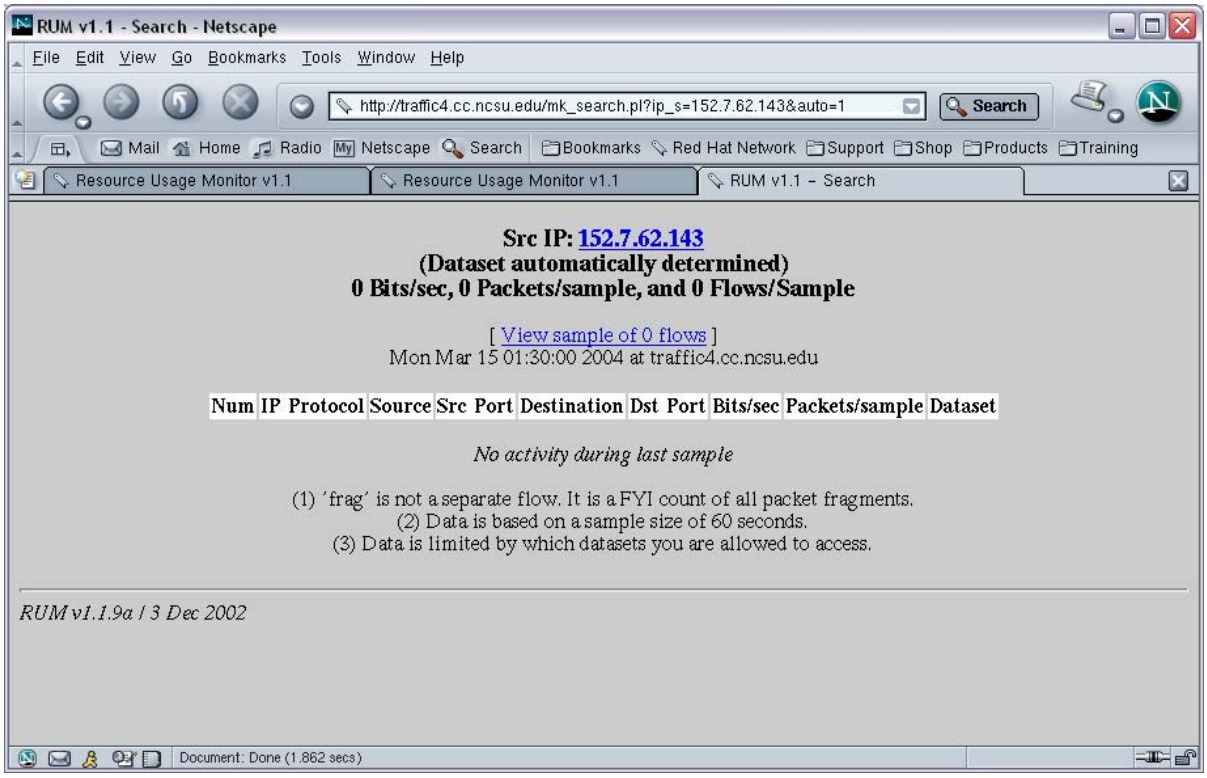


Figure 5.21 – DIDS Host Details (#12) – 152.7.62.143 (Previous Data)

RUM, in this case, does not give much information on the down hosts at all. First of all, none of the down hosts would show up on any of RUM's reports. In addition, when a manual search is performed using the down host's IP address, the host details provided by RUM simply stated that there was "no activity during last sample" (Figure 5.22).



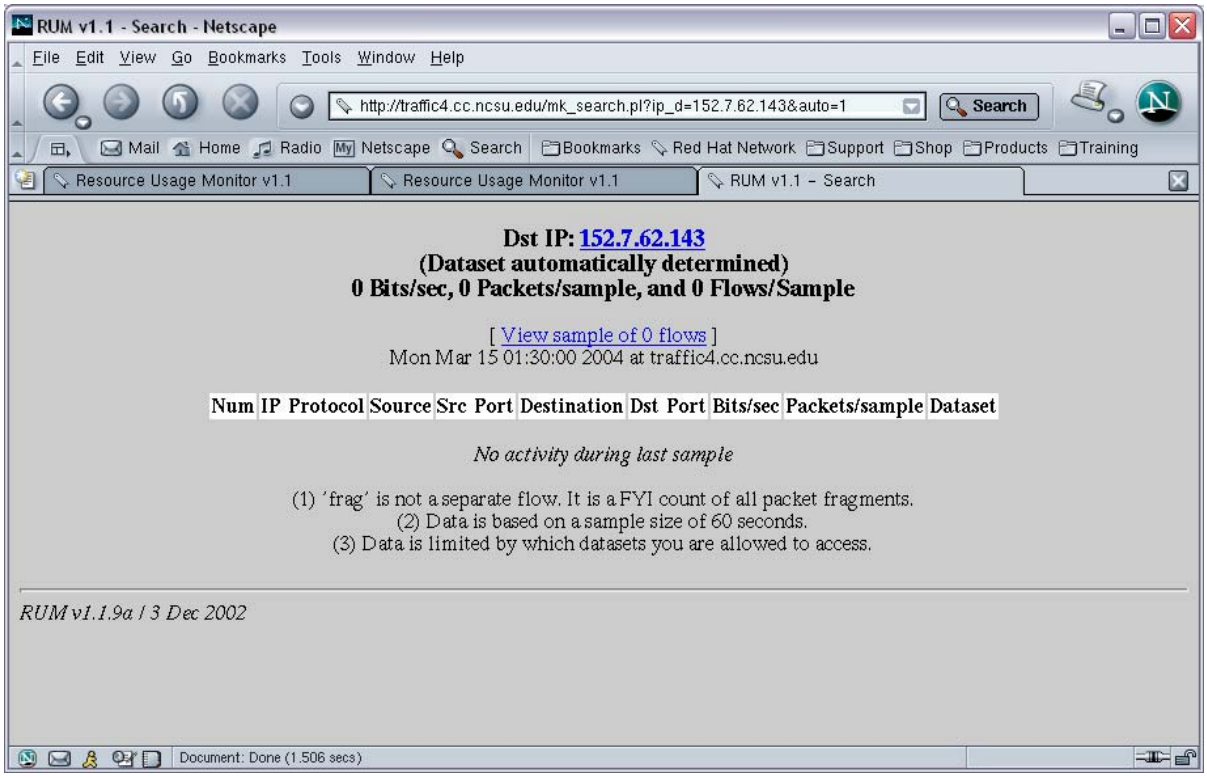


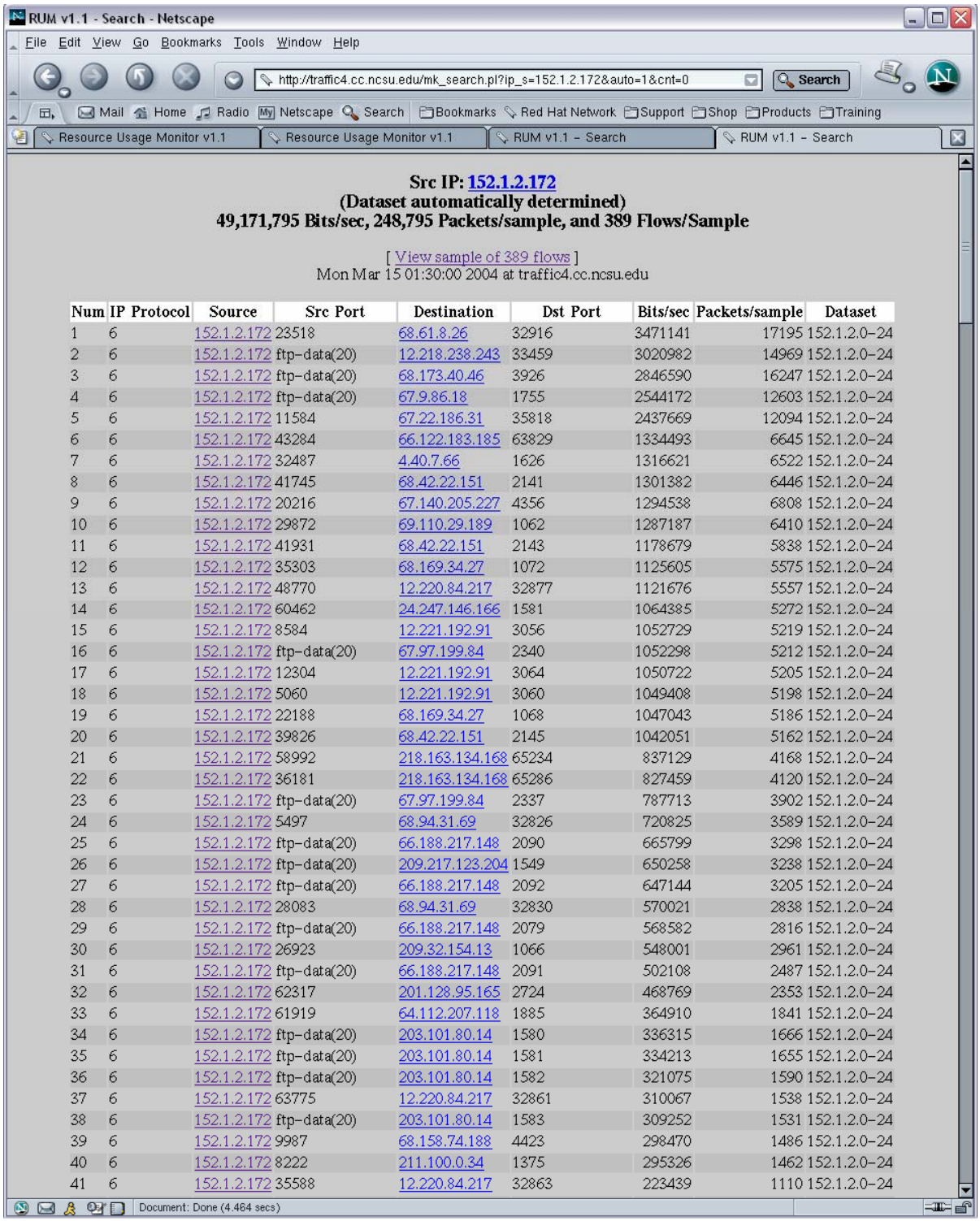
Figure 5.22 – RUM Host Details – 152.7.62.143 (No activity)

5.1.3 Second Level Analysis – Top RUM Hosts

Next, to get a view from the different perspective, a number of hosts from the RUM reports were selected, and the details were examined to see the reason why they did not make it on the top list of the intrusion detection system. For instance, the first host on RUM's top traffic load emitters list (Figure 5.4), which is also number one on both packets emitters and receivers list (Figure 5.6 and Figure 5.7, respectfully), accumulated a total index of 15,955 in Dynamic IDS and ranked number 5 (see Figure 5.2). The details of this host give by RUM are shown in Figure 5.23 and Figure 5.24, and details give by DIDS are shown in Figure 5.25 and Figure 5.26.

This host ranked number one on RUM's lists due to its high amount of traffic load and packets, and as a matter of fact, the index numbers it generated in DIDS were for the same reason too. More specifically, most the of host's total index were from Alerts and Network categories (6040 and 7837, respectfully). A closer look at the alerts messages reveals that the host exceeded both load and packets threshold, by as much as 240% in the first case and more than 340% in the latter. 5,000 of the alert index came from these two[20]. The rest of the alert index was the results of the four possible Trojan/virus ports detected.

[20] See section 3.8.2.2 for more on index from alerts.

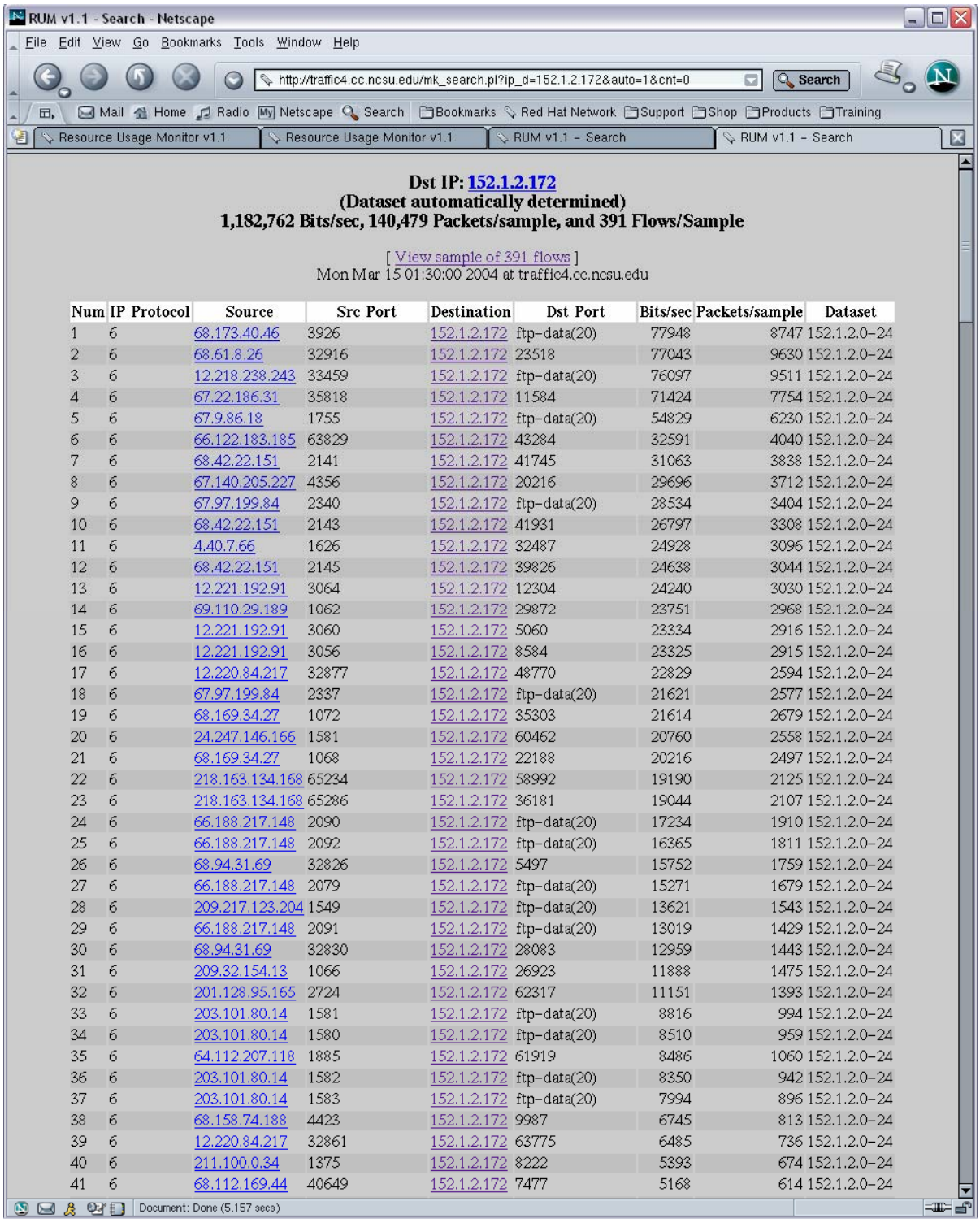


Src IP: 152.1.2.172
(Dataset automatically determined)
49,171,795 Bits/sec, 248,795 Packets/sample, and 389 Flows/Sample

[View sample of 389 flows]
Mon Mar 15 01:30:00 2004 at traffic4.cc.ncsu.edu

Num	IP Protocol	Source	Src Port	Destination	Dst Port	Bits/sec	Packets/sample	Dataset
1	6	152.1.2.172	23518	68.61.8.26	32916	3471141	17195	152.1.2.0–24
2	6	152.1.2.172	ftp-data(20)	12.218.238.243	33459	3020982	14969	152.1.2.0–24
3	6	152.1.2.172	ftp-data(20)	68.173.40.46	3926	2846590	16247	152.1.2.0–24
4	6	152.1.2.172	ftp-data(20)	67.9.86.18	1755	2544172	12603	152.1.2.0–24
5	6	152.1.2.172	11584	67.22.186.31	35818	2437669	12094	152.1.2.0–24
6	6	152.1.2.172	43284	66.122.183.185	63829	1334493	6645	152.1.2.0–24
7	6	152.1.2.172	32487	4.40.7.66	1626	1316621	6522	152.1.2.0–24
8	6	152.1.2.172	41745	68.42.22.151	2141	1301382	6446	152.1.2.0–24
9	6	152.1.2.172	20216	67.140.205.227	4356	1294538	6808	152.1.2.0–24
10	6	152.1.2.172	29872	69.110.29.189	1062	1287187	6410	152.1.2.0–24
11	6	152.1.2.172	41931	68.42.22.151	2143	1178679	5838	152.1.2.0–24
12	6	152.1.2.172	35303	68.169.34.27	1072	1125605	5575	152.1.2.0–24
13	6	152.1.2.172	48770	12.220.84.217	32877	1121676	5557	152.1.2.0–24
14	6	152.1.2.172	60462	24.247.146.166	1581	1064385	5272	152.1.2.0–24
15	6	152.1.2.172	8584	12.221.192.91	3056	1052729	5219	152.1.2.0–24
16	6	152.1.2.172	ftp-data(20)	67.97.199.84	2340	1052298	5212	152.1.2.0–24
17	6	152.1.2.172	12304	12.221.192.91	3064	1050722	5205	152.1.2.0–24
18	6	152.1.2.172	5060	12.221.192.91	3060	1049408	5198	152.1.2.0–24
19	6	152.1.2.172	22188	68.169.34.27	1068	1047043	5186	152.1.2.0–24
20	6	152.1.2.172	39826	68.42.22.151	2145	1042051	5162	152.1.2.0–24
21	6	152.1.2.172	58992	218.163.134.168	65234	837129	4168	152.1.2.0–24
22	6	152.1.2.172	36181	218.163.134.168	65286	827459	4120	152.1.2.0–24
23	6	152.1.2.172	ftp-data(20)	67.97.199.84	2337	787713	3902	152.1.2.0–24
24	6	152.1.2.172	5497	68.94.31.69	32826	720825	3589	152.1.2.0–24
25	6	152.1.2.172	ftp-data(20)	66.188.217.148	2090	665799	3298	152.1.2.0–24
26	6	152.1.2.172	ftp-data(20)	209.217.123.204	1549	650258	3238	152.1.2.0–24
27	6	152.1.2.172	ftp-data(20)	66.188.217.148	2092	647144	3205	152.1.2.0–24
28	6	152.1.2.172	28083	68.94.31.69	32830	570021	2838	152.1.2.0–24
29	6	152.1.2.172	ftp-data(20)	66.188.217.148	2079	568582	2816	152.1.2.0–24
30	6	152.1.2.172	26923	209.32.154.13	1066	548001	2961	152.1.2.0–24
31	6	152.1.2.172	ftp-data(20)	66.188.217.148	2091	502108	2487	152.1.2.0–24
32	6	152.1.2.172	62317	201.128.95.165	2724	468769	2353	152.1.2.0–24
33	6	152.1.2.172	61919	64.112.207.118	1885	364910	1841	152.1.2.0–24
34	6	152.1.2.172	ftp-data(20)	203.101.80.14	1580	336315	1666	152.1.2.0–24
35	6	152.1.2.172	ftp-data(20)	203.101.80.14	1581	334213	1655	152.1.2.0–24
36	6	152.1.2.172	ftp-data(20)	203.101.80.14	1582	321075	1590	152.1.2.0–24
37	6	152.1.2.172	63775	12.220.84.217	32861	310067	1538	152.1.2.0–24
38	6	152.1.2.172	ftp-data(20)	203.101.80.14	1583	309252	1531	152.1.2.0–24
39	6	152.1.2.172	9987	68.158.74.188	4423	298470	1486	152.1.2.0–24
40	6	152.1.2.172	8222	211.100.0.34	1375	295326	1462	152.1.2.0–24
41	6	152.1.2.172	35588	12.220.84.217	32863	223439	1110	152.1.2.0–24

Figure 5.23 – RUM Host Details – 152.1.2.172 as Source (Outbound)



Dst IP: 152.1.2.172
(Dataset automatically determined)
1,182,762 Bits/sec, 140,479 Packets/sample, and 391 Flows/Sample

[View sample of 391 flows]
Mon Mar 15 01:30:00 2004 at traffic4.cc.ncsu.edu

Num	IP Protocol	Source	Src Port	Destination	Dst Port	Bits/sec	Packets/sample	Dataset
1	6	68.173.40.46	3926	152.1.2.172	ftp-data(20)	77948	8747	152.1.2.0-24
2	6	68.61.8.26	32916	152.1.2.172	23518	77043	9630	152.1.2.0-24
3	6	12.218.238.243	33459	152.1.2.172	ftp-data(20)	76097	9511	152.1.2.0-24
4	6	67.22.186.31	35818	152.1.2.172	11584	71424	7754	152.1.2.0-24
5	6	67.9.86.18	1755	152.1.2.172	ftp-data(20)	54829	6230	152.1.2.0-24
6	6	66.122.183.185	63829	152.1.2.172	43284	32591	4040	152.1.2.0-24
7	6	68.42.22.151	2141	152.1.2.172	41745	31063	3838	152.1.2.0-24
8	6	67.140.205.227	4356	152.1.2.172	20216	29696	3712	152.1.2.0-24
9	6	67.97.199.84	2340	152.1.2.172	ftp-data(20)	28534	3404	152.1.2.0-24
10	6	68.42.22.151	2143	152.1.2.172	41931	26797	3308	152.1.2.0-24
11	6	4.40.7.66	1626	152.1.2.172	32487	24928	3096	152.1.2.0-24
12	6	68.42.22.151	2145	152.1.2.172	39826	24638	3044	152.1.2.0-24
13	6	12.221.192.91	3064	152.1.2.172	12304	24240	3030	152.1.2.0-24
14	6	69.110.29.189	1062	152.1.2.172	29872	23751	2968	152.1.2.0-24
15	6	12.221.192.91	3060	152.1.2.172	5060	23334	2916	152.1.2.0-24
16	6	12.221.192.91	3056	152.1.2.172	8584	23325	2915	152.1.2.0-24
17	6	12.220.84.217	32877	152.1.2.172	48770	22829	2594	152.1.2.0-24
18	6	67.97.199.84	2337	152.1.2.172	ftp-data(20)	21621	2577	152.1.2.0-24
19	6	68.169.34.27	1072	152.1.2.172	35303	21614	2679	152.1.2.0-24
20	6	24.247.146.166	1581	152.1.2.172	60462	20760	2558	152.1.2.0-24
21	6	68.169.34.27	1068	152.1.2.172	22188	20216	2497	152.1.2.0-24
22	6	218.163.134.168	65234	152.1.2.172	58992	19190	2125	152.1.2.0-24
23	6	218.163.134.168	65286	152.1.2.172	36181	19044	2107	152.1.2.0-24
24	6	66.188.217.148	2090	152.1.2.172	ftp-data(20)	17234	1910	152.1.2.0-24
25	6	66.188.217.148	2092	152.1.2.172	ftp-data(20)	16365	1811	152.1.2.0-24
26	6	68.94.31.69	32826	152.1.2.172	5497	15752	1759	152.1.2.0-24
27	6	66.188.217.148	2079	152.1.2.172	ftp-data(20)	15271	1679	152.1.2.0-24
28	6	209.217.123.204	1549	152.1.2.172	ftp-data(20)	13621	1543	152.1.2.0-24
29	6	66.188.217.148	2091	152.1.2.172	ftp-data(20)	13019	1429	152.1.2.0-24
30	6	68.94.31.69	32830	152.1.2.172	28083	12959	1443	152.1.2.0-24
31	6	209.32.154.13	1066	152.1.2.172	26923	11888	1475	152.1.2.0-24
32	6	201.128.95.165	2724	152.1.2.172	62317	11151	1393	152.1.2.0-24
33	6	203.101.80.14	1581	152.1.2.172	ftp-data(20)	8816	994	152.1.2.0-24
34	6	203.101.80.14	1580	152.1.2.172	ftp-data(20)	8510	959	152.1.2.0-24
35	6	64.112.207.118	1885	152.1.2.172	61919	8486	1060	152.1.2.0-24
36	6	203.101.80.14	1582	152.1.2.172	ftp-data(20)	8350	942	152.1.2.0-24
37	6	203.101.80.14	1583	152.1.2.172	ftp-data(20)	7994	896	152.1.2.0-24
38	6	68.158.74.188	4423	152.1.2.172	9987	6745	813	152.1.2.0-24
39	6	12.220.84.217	32861	152.1.2.172	63775	6485	736	152.1.2.0-24
40	6	211.100.0.34	1375	152.1.2.172	8222	5393	674	152.1.2.0-24
41	6	68.112.169.44	40649	152.1.2.172	7477	5168	614	152.1.2.0-24

Figure 5.24 – RUM Host Details – 152.1.2.172 as Destination (Inbound)

The host's index from network analysis was high due to similar reason – high traffic load and number of packets. In the network analysis table, both packets and bits generated over 600 index points individually, and the average, maximum, range, variance, and standard deviation for both packets per flow and bits per flow all had index of close to 300. More specifically, the high variance and standard deviation revealed that the host's traffic spread widely average, which indicates that the host had relatively inconsistent traffic pattern. Skewness and kurtosis for the hosts were also very high – around 4 for skewness and around 23 for kurtosis, whereas the network average was only about 0.2 and 0.8. The high skewness implied that the hosts had many flows with smaller packets (and bits) per flow than its own average, and the high kurtosis showed that the host traffic was highly peaked. When compared to some of the other metrics, these two parameters did not result in a high index. Since the current system uses percentage differences from the statistical parameters directly, the results sometimes can be biased. This particular host is a good example of it. This is one of the system's limitations. However, a weight can be multiplied to such parameters (such as skewness and kurtosis in this case) to bring up the significance of these parameters.

Across from network analysis table, to the right of the page, is the self analysis table. This table presents data and index generated from self-relative comparisons. This particular host's behavior was relatively consistent to itself from the previous sample; therefore it did not accumulate a very large self index total.

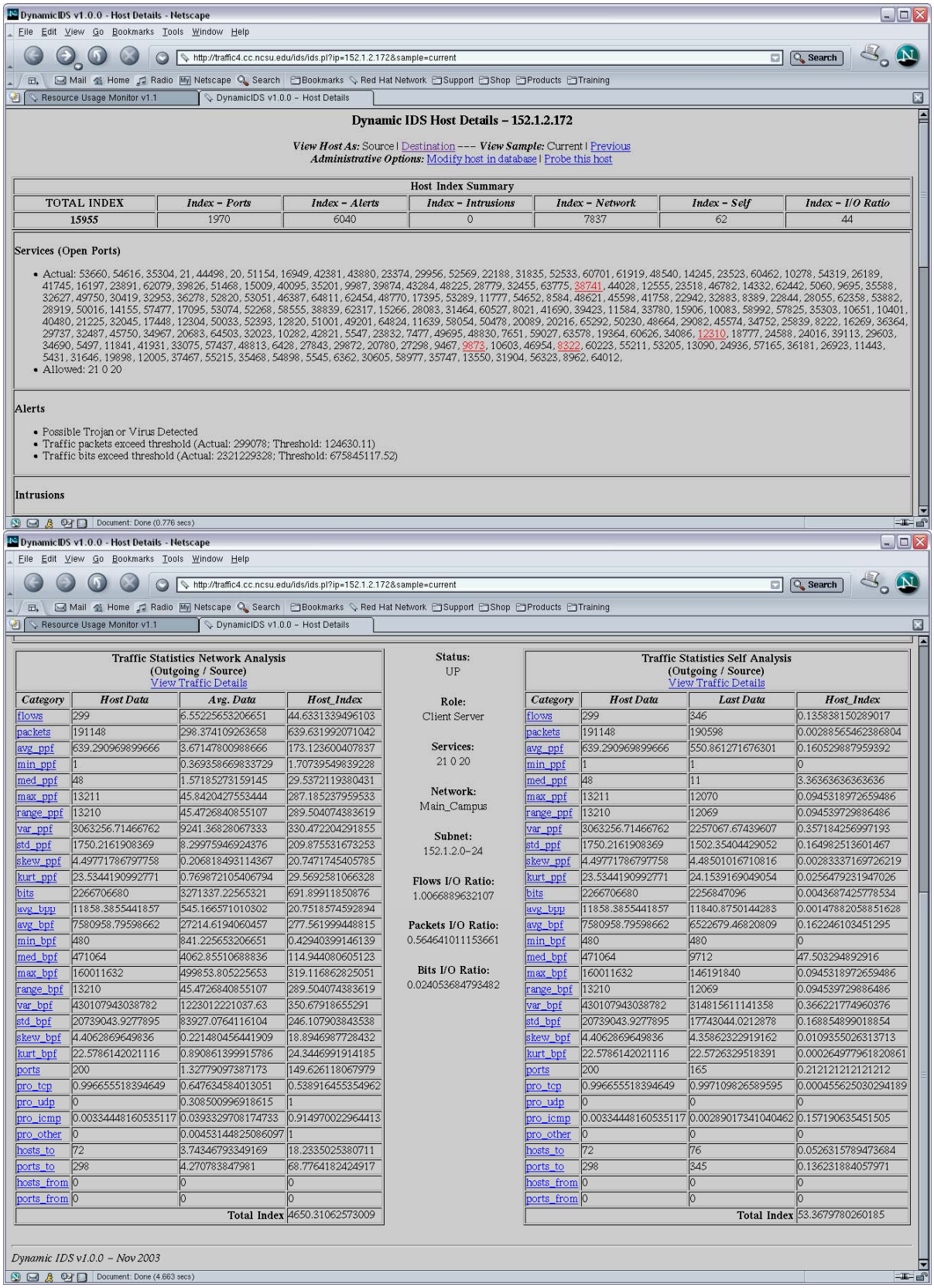


Figure 5.25 – DIDS Host Details (#5) – 152.1.2.172 as Source (Outbound)

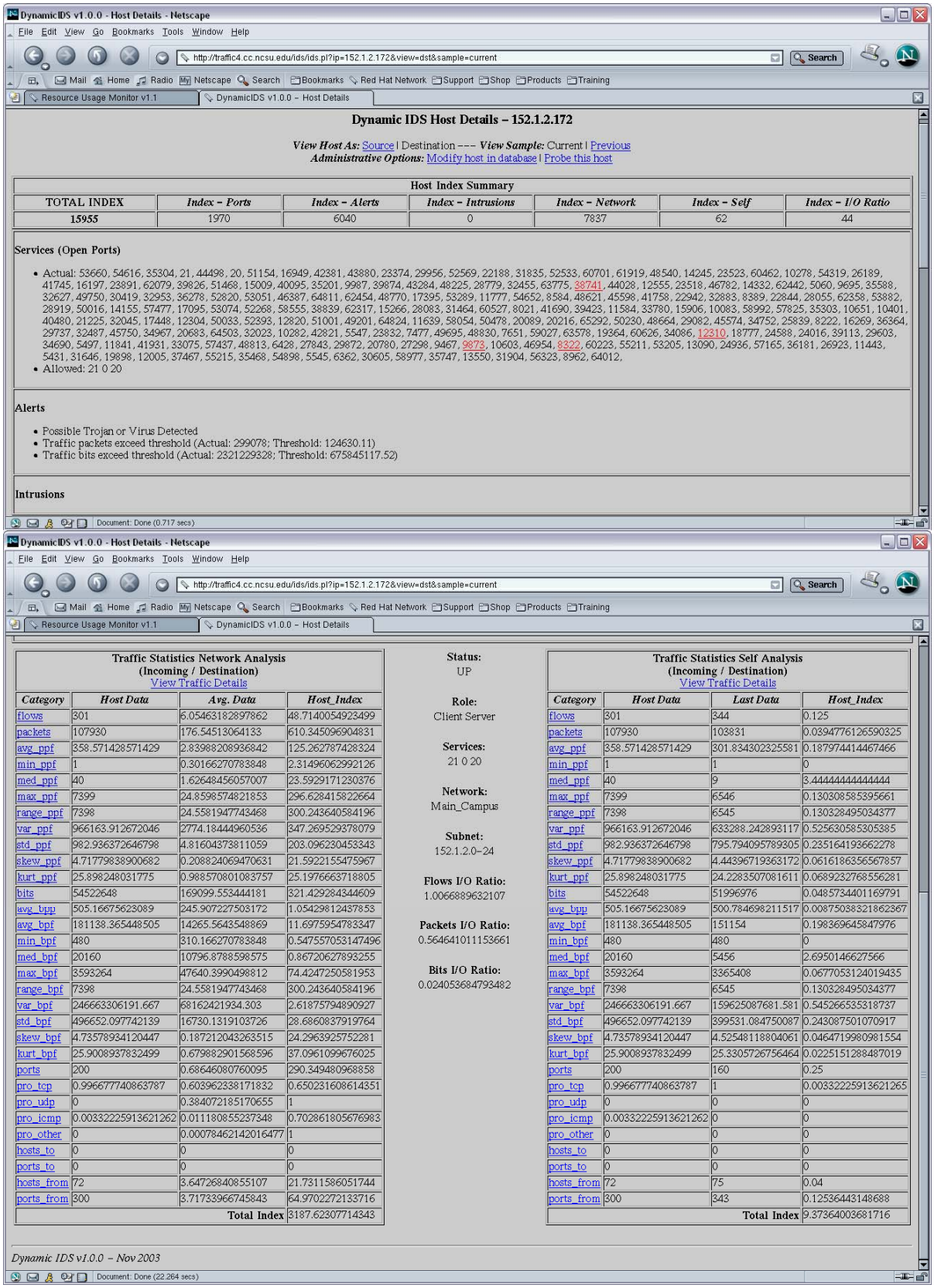
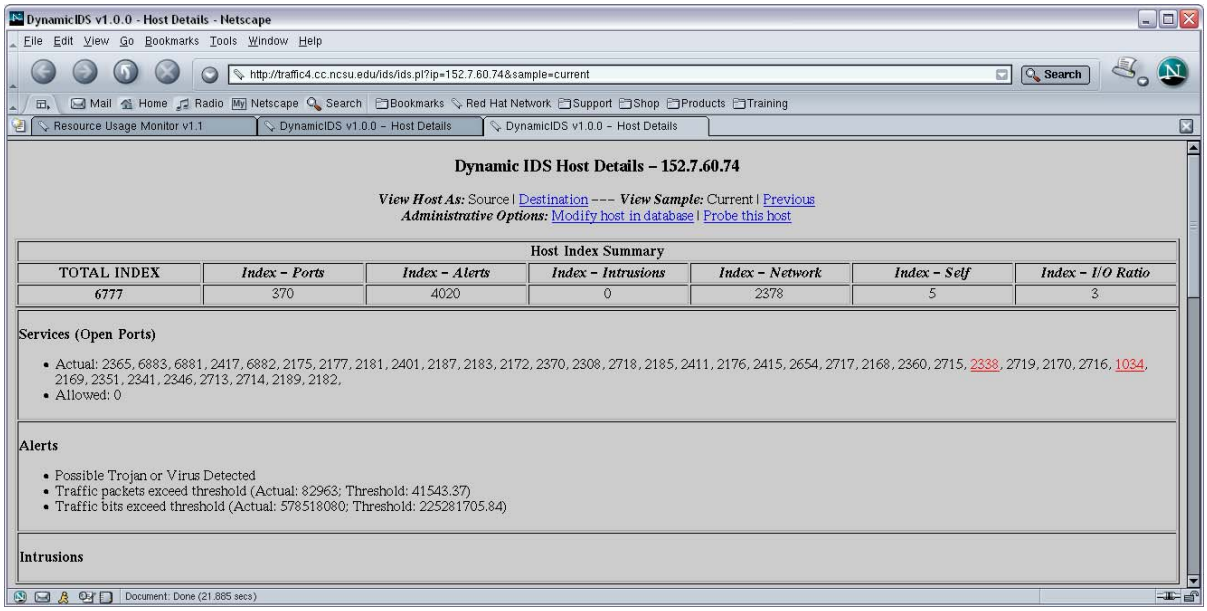


Figure 5.26 – DIDS Host Details (#5) – 152.1.2.172 as Destination (Inbound)

Nevertheless, when this host is compared to the first several hosts discussed above, its level of anomaly and severity become obviously lower. It had a much less I/O ratio than the first host (44 compared to over 100 thousand), a much lower port count than the second, third and fourth one (197 rather than 3900, 1972 and 1753).

Similarly, the first host on RUM's top load receivers list (and the second on the packets receivers list) had an index value of 6777, which ranked number 51. Aside from the high number of packets and bits it received, which generated 4020 alert index and 2378 network index, the host did not have any other severe issues such as high number of ports or high input/output ratio. Therefore, it is considered much less severe and suspicious than the DIDS system. Figure 5.27 gives the screenshot for closer visualization.



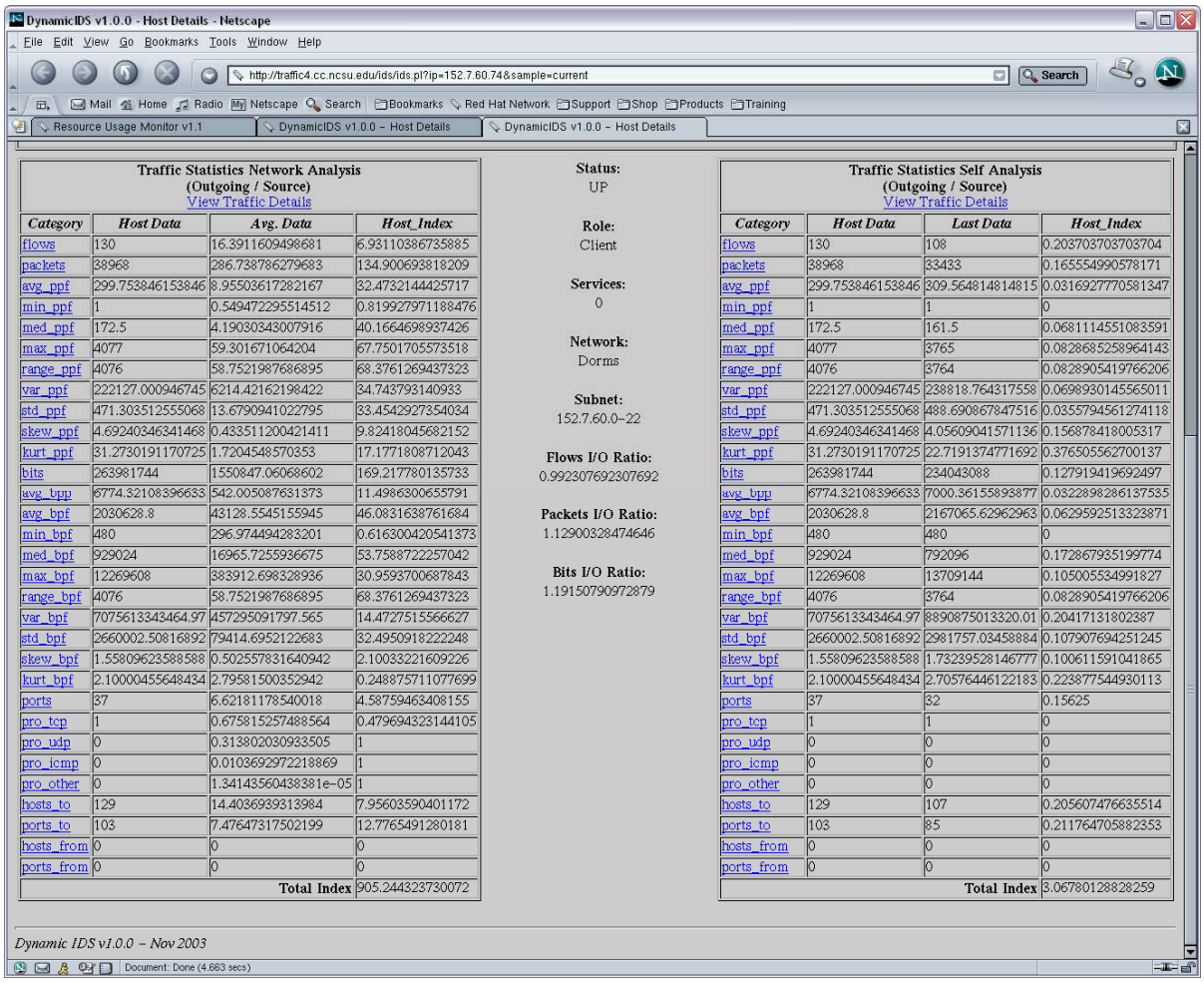


Figure 5.27 – DIDS Host Details (#51) – 152.7.60.74

For similar reason, the next two hosts (second and third on RUM's flow lists) also both scored over 10,000 of index points in IDS, although still not large enough to show up on the top 20 list. On the other hand, even most of the other hosts on the flow lists also had high number of flows, the numbers were not as drastic as the first three at all (mostly around 2000 flows instead of over 20,000 flows). In addition, they had fairly consistent incoming and outgoing number of flows, which indicates normal behavior under most circumstances. Therefore, they were not considered as the top suspicious hosts by the intrusion detection system.

In summary, as mentioned in section 5.1.1, DIDS considers much more parameters and does a lot more analysis than RUM, rather than just high flow or high load or high packets; hence the top hosts on RUM's ranking may be (e.g. number one host on RUM's flow rankings[21]) and may not be (e.g. the first host on RUM's top load receivers list[22]) on the top hosts on the DIDS's list, and vice versa.

5.1.4 SNORT Comparisons

Finally, to compare the effectiveness of Dynamic IDS against Snort, a test has been set up for both of them to analyze one-minute traffic (a sample taken on December 13, 2003). All the rules files that came with Snort distribution were turned on during the analysis. For the one-minute sample, Snort analyzed 987,493 packets and reported a surprisingly low number of 116 alerts. Out of the total 116 alerts, 110 of them – 95% – were ICMP Destination Unreachable, as shown in the example below.

[21] See discussions in section 5.1.2.

[22] See the previous discussion in the same section (5.1.3)

```
[**] [1:485:2] ICMP Destination Unreachable (Communication
Administratively Prohibited) [**]
[Classification: Misc activity] [Priority: 3]
12/13-15:26:15.276506 66.153.7.74 -> 152.7.14.169
ICMP TTL:236 TOS:0x0 ID:41432 IpLen:20 DgmLen:56
Type:3  Code:13  DESTINATION UNREACHABLE: ADMINISTRATIVELY PROHIBITED,
PACKET FILTERED
** ORIGINAL DATAGRAM DUMP:
152.7.14.169:0 -> 64.80.230.46:0
UDP TTL:113 TOS:0x0 ID:62911 IpLen:20 DgmLen:67
Len: 47
** END OF DUMP
```

Figure 5.28 – Sample Snort Alert

The remaining six alerts (for six different hosts) were one Stealth Activity (Vecna scan) detection, one Virus – Possible scr Worm, one Stealth Activity (FIN scan) detection, and two TTL Evasion (reassemble) detection. Some of the hosts flagged by Snort did show up with some general anomalies in the DIDS. For example, one of them had close to 3,000 points in index from network-relative comparisons because it had a high traffic load (bits) and another one had over 2,000 index values from both the communicating ports category and alerts category. However, none of the hosts was severe enough to score on the top 20 list. Further examination on the actual host traffic (provided by RUM) did not reveal additional anomalies either. Since Snort does pattern match against each single packet that passes through, it cannot correlate very well among all the packets in a certain sample. Yet the correlation plays a very important role in detecting many problems. For example, there is no way to accurately detect a DOS attack by looking at a single packet. As in the above example, 95% of the alerts simply say ICMP unreachable, which does not represent anything very meaningful. An ICMP unreachable packet can be absolutely normal and legitimate (e.g. when a host is performing troubleshooting on its connectivity), or it can be a sign of attack preparation. But there is no way to tell by look at a single packet. It is also very difficult and inefficient for the administrator to re-examine the

alerts log and try to correlate some of the events. In fact, since Snort cannot correlate the events itself, it is very likely that pieces of information that help identify an anomaly has already slipped away because it is not important or intrusive just by itself.

Dynamic IDS does things very differently. It considers all the information in a particular sample. In fact, during the same one-minute sample, there was a host communicating using a total of 3683 unique ports (see screenshot below). In addition, it was performing a port/host scan – it had high port usage, high number of outgoing flows (over 5000, 14 times the number of incoming flows) against a large number of unique target hosts, and small amount of packets and bits per flow. Snort does not log this host at all because none of the host's single behavior or each individual port raised a flag to Snort. Dynamic IDS put it on the very top of its ranking list. Several others on the top 20 list had similar problem, and again, none of them is logged by Snort. Dynamic IDS also presents great advantage when it comes to displaying the results. It displays hosts in a list sorting by the index number of the host. Hosts with higher index number are displayed earlier on the list because they are likely to have more serious problems and deserve higher attention. In addition, detailed information for any host can be looked up with the host's IP address, not only the hosts that the Dynamic IDS considers problematic. This is impossible in Snort, where the only thing available to the administrator is the alerts logged or the raw data.

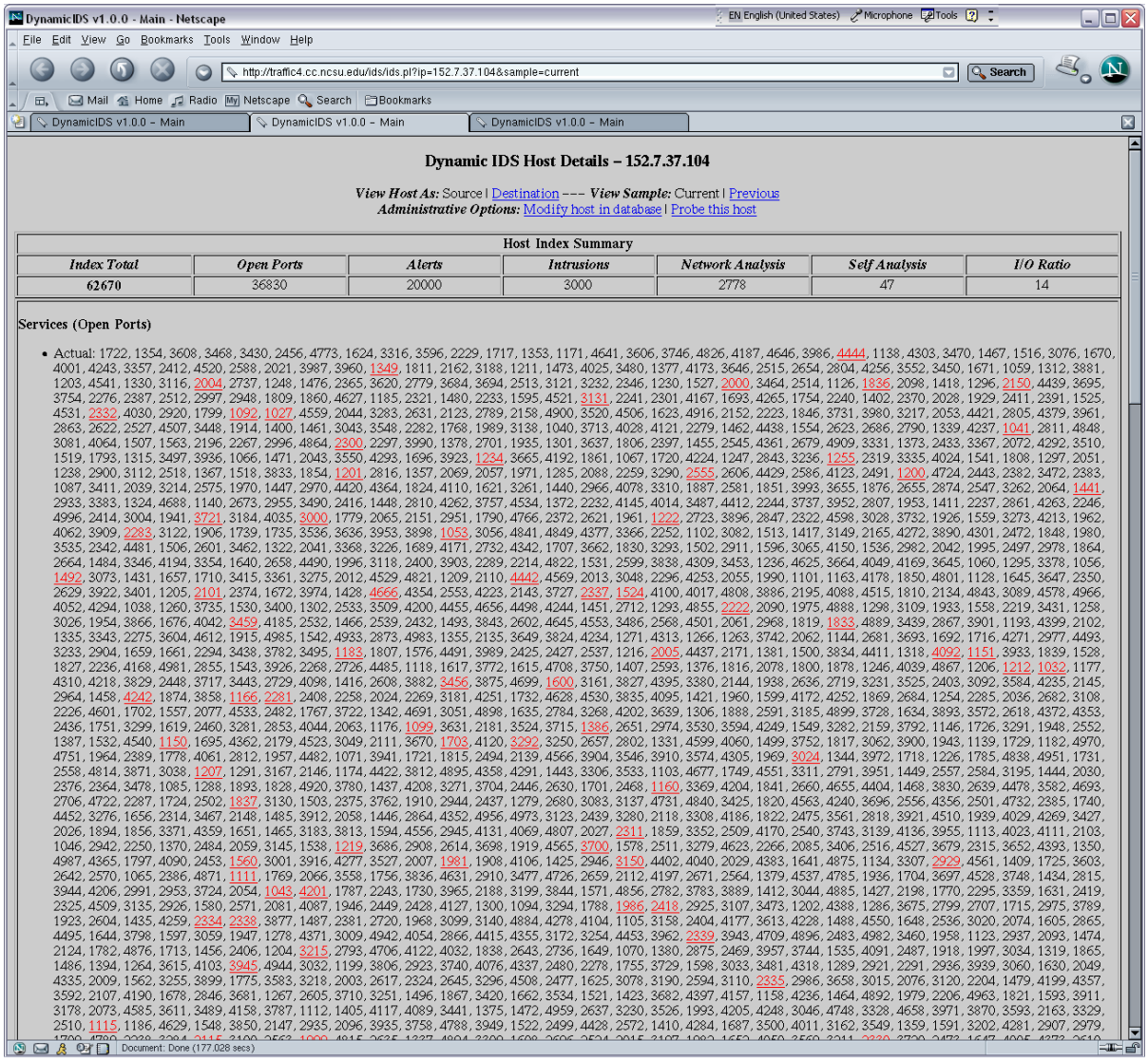


Figure 5.29 – Host Anomaly Sample Screenshot

5.2 Effectiveness Summary

From the test cases above, dynamic IDS is the clear winner in term of effectiveness. It takes into account not one but many parameters to analyze each host. Several other case studies were conducted and yield very similar results (details omitted) as previously described, when compared to RUM and other systems such as Snort. In addition, it presents the hosts according their level of severity, and the list can also be

sorted by individual index categories, so the user can easily pick the category that is of the most concern.

A report can be generated by the Dynamic IDS to show a summary of problems it has identified. The report generated for same the data sample that is used for case discussions in section 5.1 is shown here (Figure 5.30), and the report for the previous sample data is also provided (Figure 5.31).

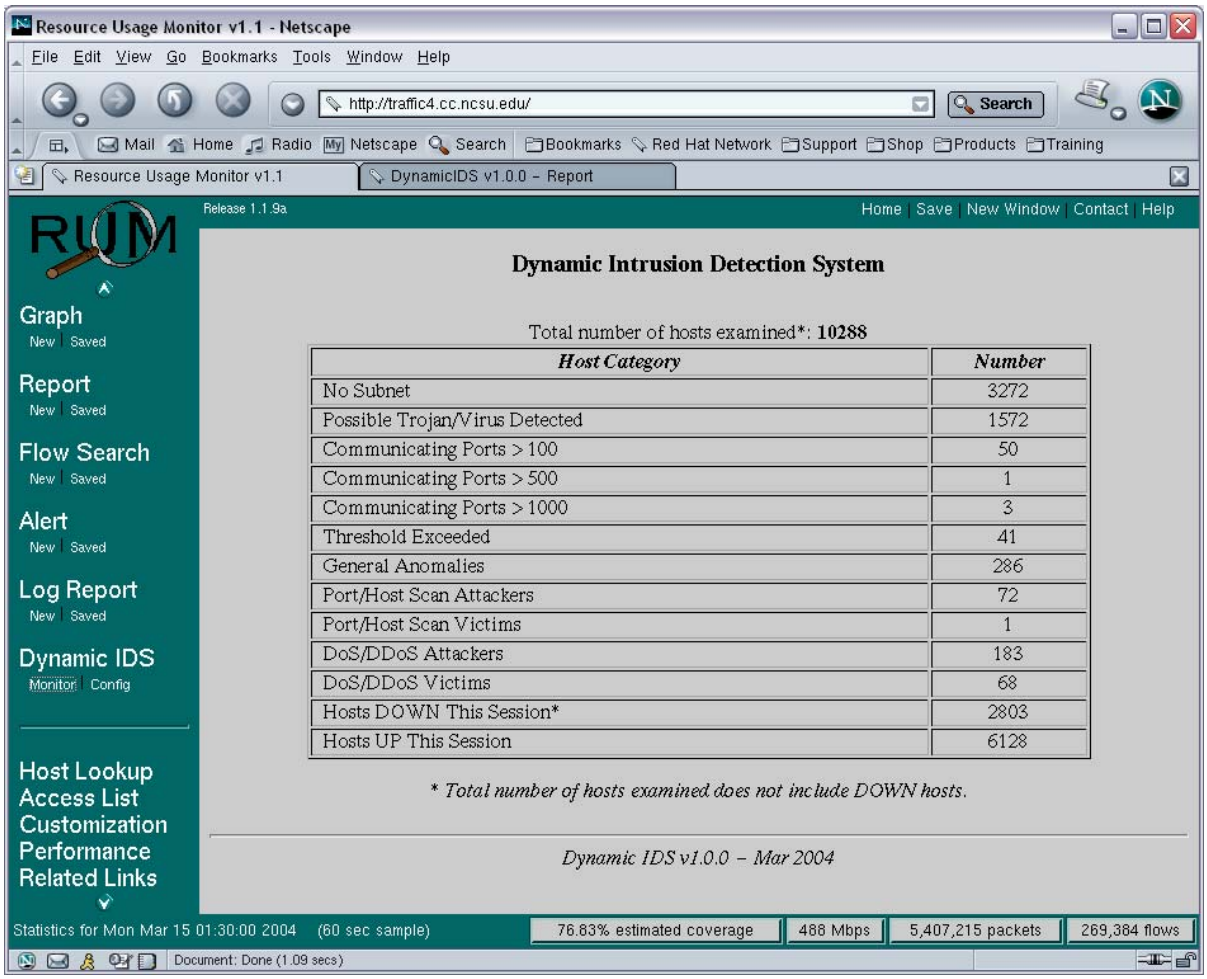


Figure 5.30 – Dynamic IDS System Report

This report shows, that out of a total of 10288 hosts, the system successfully caught 3272 hosts without a proper subnet, 1572 with possible Trojan or virus (with at least 1 matching port), 53 communicating on unusually large number of ports (50 hosts had greater than 100 ports, 1 had greater than 500, and 3 had greater than 1000[23]), 41 exceeded system thresholds (packets, bits/load, and/or flow), 286 that had general anomalies (with over 1000 in index from network- or self-relative comparisons), 72 possible port/host scanners and 1 possible scan victim, 183 possible DoS/DDoS attackers and 68 possible victims, and in addition, 2803 hosts that went down and 6128 that went up in this particular session[24]. Figure 5.31 shows another report in the same format for the preceding data sample (approximately 30 minutes before).

[23] The three hosts that had greater than 1000 communicating ports are the second, third, and fourth hosts on the DIDS ranking list. The second and third hosts are discussed in details in section 5.1.2.

[24] These numbers can be verified by taking the total number of hosts in previous sample, minus the number of down hosts then plus the number of up hosts. The result should equal exactly the total number of hosts in the current session. (e.g. 6963 – 2803 + 6128 = 10288)

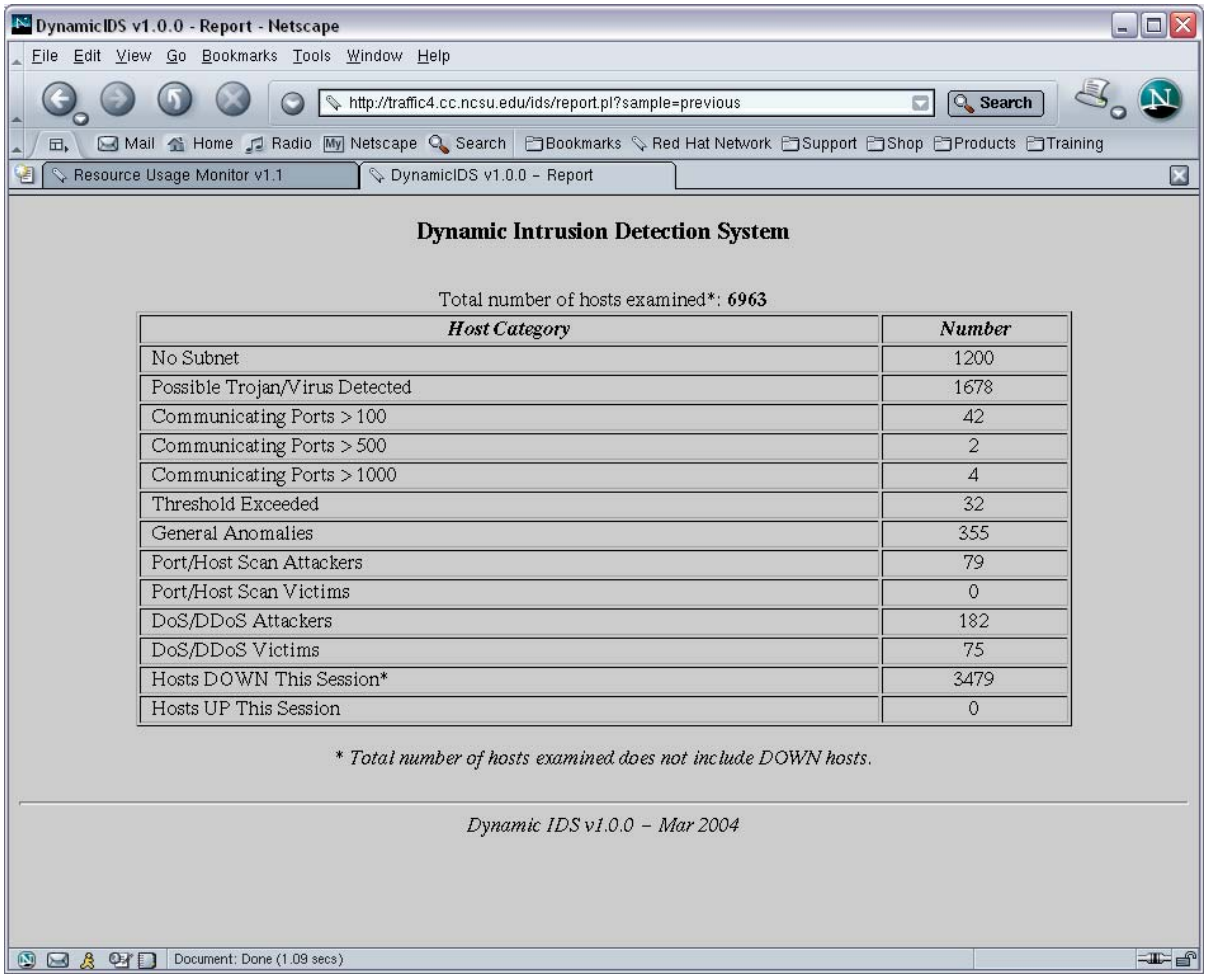


Figure 5.31 – Dynamic IDS System Report (Previous Sample)

5.3 Performance

The performance measurements are done on a 2-processor Pentium III 750MHz IBM server with 1GB of physical memory. For each step of the intrusion detection process, the system adds a timestamp immediately before the process starts and immediately after it ends, in order to capture the exact time spent on each step as well as on the entire process. As a result, for a relatively small amount of traffic, at a speed of around 300 Mbps, the system completes the analysis in around 7 minutes. When the traffic increases to about 500Mbps, it takes the system 12 minutes to complete its

analysis. However, the system is currently un-threaded, which means, although physically there are two processors present, the system process is only using one of them. This is also addressed in the next section as one of the system's limitations and also in the conclusions section as a possible future work. In addition, RUM takes on average 5 to 8 minutes to capture and sort the data; the total time required is around 20 minutes. Therefore, the optimal interval between each scan for the currently system is 30 minutes.

Similar to the effectiveness tests, two tests have been run on both Snort and Dynamic IDS to compare the performance. The first test was run on a weekend when traffic is relatively low, whereas the second test was run on a weekend with normal load on the network. During the tests, Snort (with default configuration in `snort.conf` file) was run for one minute continuously, and immediately following, Dynamic IDS was with one minute sampling period. As a result, in the first test, Snort reported that it has processed 1,340,361 out of 1,341,677 packets that it had captured. On the other hand, Dynamic IDS shows it covered approximately 87.26% of 5,111,407 packets (at 332Mbps). Apparently, Snort was only able to analyze about 30% of the traffic that was analyzed by Dynamic IDS, which was about 87Mbps. According to the Snort User Manual [31], even with the fast option turned on, Snort is only capable of keeping up with 100Mbps traffic. That almost matches exactly to what we saw during the test. The problem with the fast option is that Snort will log all traffic in binary Tcpdump format. To retract useful information from the raw data, Snort has to be run again on the binary file to generate alerts and logs. Dynamic IDS is a different story. Since it is running in statistical mode, it is potentially capable to keep up at any speed,

and the data is presented as soon as the analysis finishes. On a slower network, the system can be adjusted to run more frequently; and on a faster network, it can run less frequently to keep up with the mass amount of traffic. It has been tested on another server with two Pentium III 1GHz and 2GB of memory that the system was able to provide 100% coverage of traffic at a speed of over 600Mbps. In other words, a faster machine can be used on large networks to achieve better speed and coverage.

With the advance in computer processing chips technology nowadays, the IDS systems may yield very pleasant performance results on a multi-gigahertz machine with multi-gigabyte of memory and faster hard drives. Unfortunately, a machine with such configuration was not available for this research study. Nevertheless, performance of the system is one area that still needs improvement on.

5.4 Limitations

Like any other application, the dynamic intrusion detection system also has its own limitations. First of all, DIDS relies on system administrators, to some degree, to ensure a completely accurate set of information in end-host database[25]. Although complete automation and complete accuracy cannot possibly be guaranteed at the same time, an enhanced process in the system's initialization phase can reduce the amount of human work required.

Another limitation is the way that the metrics are used in the current system. Currently, the system uses the percentage difference of the metrics (see detailed

[25] See section 4.4 – Turning the System for more information on system initialization and database tuning.

discussion at the end of section 3.8.2.3). However, this can sometimes be biased (see example in the first discussion in section 5.1.3).

DIDS's system processing method is another current limitation. As mentioned in the performance evaluation (section 5.3) above, the processes of dynamic IDS are not threaded. In other words, the processes will only use one CPU in the system, even though there may be multiple ones. This introduces a scalability drawback, making it difficult to increase the processing speed. The workaround to gain performance for the current system is to use a faster CPU, such as replacing a Pentium III 1GHz with a Pentium 4 3GHz.

Finally, certain limitations exist in the user interface. For example, it would be helpful for the system to display the hostname along with the host IP address on its ranking list (see section 5.1.1); an advanced search form with multiple options can allow the user to pinpoint the exact subset of hosts desired; and it would be helpful if the report page can lead to display the collections of hosts that fall into each one of the anomaly categories shown on the page (see section 5.1.3).

Chapter 6 – Conclusions

6.1 Summary

The principal goal of the work presented in this paper was to develop and test the RUM IDS module and its initial set of metrics. This paper described the theory, development, and testing of a dynamic intrusion detection module built on the NC State open source Resource Usage Monitoring (RUM) tool [31]. This dynamic intrusion detection module utilizes RUM as its statistical packet capturing and basic analysis engine. It further uses RUM to cross check its problem detection abilities, and adds to RUM's resource risk assessment ability a facility for intrusion risk assessment using a suite of behavior description measures and intrusion threshold indicators.

More specifically, the dynamic IDS module uses the end-host database, which contains the hosts' registered services, roles and other vital information as the foundational knowledge (section 3.3), and a set of statistical metrics as the dynamic profile that describes the hosts' and networks' behavior and traffic characteristics (section 3.4 – 3.7), to analyze each network host for anomalies and misbehaviors. The system goes through a six-step hierarchical analysis process (section 4.3): examining the host database, measuring traffic thresholds, performing network- and self-relative comparisons, calculating inbound/outbound traffic ratio, and finally determining the hosts' up/down status. During the analyses, each host accumulates an anomaly index value based on a linear additive model, where a higher number represents a higher likelihood of misbehavior (section 3.8). The paper also provided a series of case

studies on the current implementation of the dynamic IDS using RUM and SNORT as the comparison products, which showed that dynamic IDS does indeed provide better high-speed problem detection when combined with a human analyst than the latter two.

To conclude, we have achieved the initial goals as follows:

1) developed and tested the dynamic IDS module for RUM,

2) determined an initial set of metrics (section 3.6) and two different "views" (section 3.8.2.3) (network- and self-relative) used in detecting anomalies related to network-based intrusion attempts and proven to be capable of detecting undefined anomalies (section 5.1), and

3) developed a (prototype) anomaly index (section 3.8) to represent the likelihood of misbehavior and by ranking them in different categories on an easy-to-use web interface (section 4.5), the system seamlessly integrated with human intrusion detection experiences and at the same time minimized false positives by allowing final decision making to human analysts.

6.2 Future Work

The first set of potential future work is to overcome the limitations of the current system, as described in section 5.4 of this thesis. For example, one can increase the system performance by adding multithreading capability and/or converting the source from Perl to a faster language such as C. In particular, each individual metric (see section 3.6) and their significance should be further studied and explored to reduce, if not completely remove, the possible biases caused by the percentage differences used

in the current system. In addition, it is also a potential possibility to continue the research onto the next phase with enhanced metrics, algorithms, and system processes.

References

[1] McGuire, David and Brian Krebs, ***Attack on Internet Called Largest Ever***, October 22, 2002, Washington Post, available at http://www.washingtonpost.com/ac2/wp-dyn/A50765-2002Jun26.

[2] CSI and San Francisco FBI, ***2001 "Computer Crime and Security Survey"***, available at http://www.crime-research.org/eng/library/Cyberstat.htm.

[3] Matt Curtin. ***Introduction to Network Security***. March 1997. Available at http://www.interhack.net/pubs/network-security/.

[4] C. Kaufman, R. Perlman and M. Speciner. ***Network Security – Private Communication in a Public World***. 2nd Edition. 2002. pp 15, 19-27, 593-594. Prentice Hall PTR.

[5] Innella, Paul (Tetrad Digital Integrity, LLC), November 2001, ***The Evolution of Intrusion Detection Systems***, at http://online.securityfocus.com/infocus/1514.

[6] Mark Gerken, Air Force Rome Laboratory. ***Intrusion Detection.*** Software Engineering Institute, Carnegie Mellon Univ. Jan 1997, available at http://www.sei.cmu.edu/str/descriptions/intrusions.html.

[7] Sample, Char and etc., ***Firewall and IDS Shortcomings***, October 2002. First presented at SANS Network Security, Monterey, California.

[8] Chuvakin, Anton, ***Network IDS Shortcomings: Has NIDS Reached the End of the Road?*** February 2002, available at http://www.infosecnews.com/opinion/2002/02/06_02.htm.

[9] B. Mukherjee, L. Herberlein, and K. Levitt (1994), Network Intrusion Detection, ***IEEE Network May/June 1994***, Page 26-41.

[10] P. Innella and O. McMillan (2001). ***An Introduction to Intrusion Detection Systems***. Available at http://www.securityfocus.com/infocus/1520.

[11] M. Asaka, A. Taguchi and S. Goto. "The Implementation of IDA: An Intrusion Detection Agent System". ***In Proceedings of the 11th Annual FIRST Conference on Computer Security Incident Handling and Response (FIRST'99)***. June 1999.

[12] David Elson. ***Intrusion Detection, Theory and Practice***. March 2000. Available at http://www.securityfocus.com/infocus/1203.

[13] Aurobindo Sundaram. "An Introduction to Intrusion Detection." ***ACM Crossroads Student Magazine***, Vol. 2, No. 4, April 1996. http://www.acm.org/crossroads/xrds2-4/intrus.html.

[14] Mark Gerken, Air Force Rome Laboratory. ***Rule-Based Intrusion Detection.*** Software Engineering Institute, Carnegie Mellon Univ. Jan 1997, available at http://www.sei.cmu.edu/str/descriptions/rbid.html.

[15] Symantec. ***Intrusion detection systems: Defining protocol anomaly detection***. April 2003. Available at http://www.igov.com/vendor/symantec/IDS_anomaly.pdf.

[16] M. Asaka, T. Onabuta, T. Inoue, S. Okazawa and S. Goto, "A New Intrusion Detection Method Based on Discriminant Analysis", ***IEEE TRANS. INF. & SYST.***, Vol. E84-D, No. 5, pp 570-577, 2001.

[17] Mark Gerken, Air Force Rome Laboratory. ***Statistical-Based Intrusion Detection.*** Software Engineering Institute, Carnegie Mellon Univ. Jan 1997, available at http://www.sei.cmu.edu/str/descriptions/sbid_body.html.

[18] H. Javitz and A. Valdes. ***The SRI IDES Statistical Anomaly Detector.*** IEEE CH2986-8/91/0000/0316, 1991.

[19] Smaha, Stephen E. "Haystack: An Intrusion Detection System," pp 37-44. ***Proceedings of the Fourth Aerospace Computer Security Applications Conference.*** Orlando, Florida, December 12-16, 1988. Washington, DC: IEEE Computer Society Press, 1989.

[20] Mukherjee, Biswanath, L.; Heberlein, Todd; & Levitt, Karl N. "Network Intrusion Detection." ***IEEE Network 8,*** 3 (May/June 1994): pp 26-41.

[21] D. Anderson, T. Frivold and A. Valdes. ***Next-generation Intrusion Detection Expert System (NIDES): A Summary.*** SRI International Technical Repost SRI-CSL-95-07. May, 1995.

[22] Dorothy E. Denning. “An Intrusion-Detection Model”. ***IEEE Transactions on Software Engineering***, Vol. SE-13, No. 2, February 1987, 222-232.

[23] C. Manikopoulos and S. Papavassiliou. “Network Intrusion and Fault Detection: A Statistical Anomaly Approach”, ***IEEE Communications Magazine***, Vol. 40, No. 10, pp. 76-82. October 2002.

[24] H. Debar. ***Intrusion Detection FAQ - What is behavior-based intrusion detection.*** Security Library :: Intrusion Detection :: Intrusion Detection FAQ, SANS Institute, October 2002 available at http://www.sans.org/resources/idfaq/behavior_based.php.

[25] McCarthy, Marci and etc., ***Behavior-based IDS – Navigating the Unknown***, Lancope Webcast at www.searchsecurity.com.

[26] Lancope, Inc. ***The Security Benefits of a Behavior-Based Intrusion Detection System.*** White Paper. September 2002.

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

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

[29] L. Degioanni, F. Risso, and P. Viano. ***WinDump.*** 1999. available at http://netgroupserv.polito.it/windump/.

[30] L. Degioanni, F. Risso, and P. Viano. ***WinPcap.*** 1999. available at http://netgroupserv.polito.it/windump/.

[31] Brian D. Goff, ***Distributed Resource Monitoring Tool and Its Use in Security and Quality of Service Evaluation***. NC State University. 2002.

[32] ***Snort – the open source intrusion detection system.*** (2002). Retrieved March 28, 2003, from http://www.snort.org.

[33] M. Roesch. "Snort - Lightweight Intrusion Detection for Networks", ***Proceedings of the 13th Systems Administration Conference (LISA '99).*** 1999; pp 229-236.

[34] Nmap. ***Nmap network security scanner man page***. September 2003. Available at http://www.insecure.org/nmap/data/nmap_manpage.html.

[35] Fyodor. ***The Art of Port Scanning***. 2003. Available at http://www.insecure.org/nmap/nmap_doc.html.

[36] R. Sharpe. ***Ethereal's User Guide***, 2001, available at http://www.ethereal.com/docs/userguide/.

[37] NFR Security, Inc. ***NFR Network Intrusion Detection Overview***, 2001, available at http://www.nfr.com/products/NID.

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

[39] Agnitum. ***What is a Trojan Horse and What Threat Does it Pose?*** Available at http://www.agnitum.com/products/tauscan/ttour1.html.

[40] Daniel Pétri. ***Trojan Port List***. Available at http://www.petri.co.il/trojan_ports_list.htm.

[41] Whatis.com Encyclopedia. ***Definition: Denial of Service***. Available at http://whatis.techtarget.com/definition/0,289893,sid9_gci213591,00.html.

[42] SearchSecurity.com Definitions. ***Intrusion Detection***. Available at http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci295031,00.html.

[43] Roamer. ***Network IDS Sensor Placement***. January 2001. Available at http://www.securityhorizon.com/whitepapers/technical/IDSplace.html.

[44] R. Walpole, R. Myers, S. Myers and K Ye. ***Probability & Statistics for Engineers and Scientists***. 7th Edition. Page 110-111. 2002. Prentice Hall, Inc.

[45] S. McCanne and V. Jacobson. ***The BSD Packet Filter: a New Architecture for User-level Packet Capture***. In USENIX, 1993.

[46] T. Oetiker, ***RRDtool Manual***, 2000, available at http://www.rrdtool.com/manual/.

[47] Information Sciences Institute, University of Southern California. ***RFC 793: Transmission Control Protocol***. September 1981. Available at http://www.ietf.org/rfc/rfc0793.txt?number=793.

[48] Julia Allen, etc. ***State of the Practice of Intrusion Detection Technologies***. January, 2000, at http://www.sei.cmu.edu/pub/documents/99.reports/pdf/99tr028.pdf.

[49] Raphael Manfredi. ***Perl Storable***. Available at http://search.cpan.org/~ams/Storable-2.08/Storable.pm.

[50] Apache Software Foundation. ***About the Apache HTTP Server Project***. Available at http://httpd.apache.org/ABOUT_APACHE.html.

[51] J. Reynolds. ***RFC 1135: The Helminthiasis of the Internet***. December 1989. Available at http://www.ietf.org/rfc/rfc1135.txt.

[52] Najmi. ***Types of Attacks on Web Servers***. June 2002. Available at http://www.techiwarehouse.com/Articles/2002-06-24.html.

[53] Marcus J. Ranum. ***Internet Attacks***. 1997. Available at http://www.sics.se/~lra/thesis/library/ranum96taxonomy.pdf.

[54] Allan Liska. ***Network Security: Understanding Types of Attacks***. June 2003. Available at http://www.informit.com/isapi/product_id~{EFFCDEEF-038A-4F14-A73D-47C49A1C7106}/session_id~{2FDAE0BF-7657-4633-B97B-07957D978E21}/content/index.asp.

[55] Tripwire (Open-Source). http://www.tripwire.org.

[56] Tripwire (Commercial). http://www.tripwire.com.

[57] J. D. Musa. "Operational Profiles in Software-Reliability Engineering." ***IEEE Software***, Vol. 10, Issue 2, Page 14-32. March 1993.

[58] Amogh Dhamdhere. ***Internet Traffic Characterization***. Available at http://www.cc.gatech.edu/classes/AY2004/cs8803ntm_fall/amogh.ppt.

Appendix A – Application Modules & Scripts

Default application root directory:

```
/usr/local/rum-ids
```

Sub-directories:

```
ids-src
      -> Main Application Modules
ids-data
      -> Application Data Files for Current Sample
ids-data.last
      -> Application Data Files for Previous Sample
scripts
      -> Additional Scripts
```

Main application modules (in directory `ids-src`):

```
IDSinit.pl
      -> Initialize RUM-IDS; calls parse_subnets.pl, mk_hdb.pl,
        and mk_rdb.pl
IDSengine.pl
      -> Calls mk_history.pl, mk_profile.pl, and mk_index.pl
mk_hdb.pl
      -> Creates host registered ports database
mk_rdb.pl
      -> Creates host roles database
mk_history.pl
      -> Create traffic history database for each host
mk_profile.pl
      -> Create profiles for individual hosts and networks
mk_index.pl
      -> Main anomaly detection analysis and index generation
parse_subnets.pl
      -> Reverse RUM subnets storage format
```

Web interface modules (in directory `ids-src`):

```
ids.pl
      -> Monitoring Web Interface Module
config.pl
      -> Configuration Web Interface Module
probe.pl
      -> Actively probe a host for open ports using Nmap
```

```
report.pl
      -> Reporting Web Interface Module
sethdb.pl
      -> Host Database Configuration Web Interface Module
trojan_port.pl
      -> Display Trojan port information
```

Main application data files (in directories `ids-data` and `ids-data.last`):

```
hosts.db
      -> registered service ports for all hosts
roles.db
      -> client/server role for all hosts
net.db
      -> Network-to-IP mapping information
hostnet.db
      -> Host-to-Network mapping information
vlan.db
      -> VLAN-to-IP mapping information
hosts.current
      -> Hosts in current sample
history.db
      -> Current traffic flow data
hostpro.db
      -> Individual host's profiles
profile.db
      -> Network profiles
alerts.db
      -> Alert messages for each host
intrusions.db
      -> Intrusion messages for each host
host_index.db
      -> Category anomaly index for each host
host_index_total.db
      -> Total anomaly index for each host
host_ratio.db
      -> Flow, Packet, and Bit Input/Output Ratio for each host
status.db
      -> UP/DOWN network status for each host
report.db
      -> IDS system reporting statistics
terms.db
      -> Contains definitions of statistical variables
trojans.db
      -> Common Trojan/virus ports information database
```

Additional scripts (in directory `scripts`):

```
datalook.pl
      -> Dumps a Perl storage file in readable format
examine.pl
      -> Same as datalook.pl
ip_convert.pl
      -> Converts an IP address b/w binary and dotted decimal
print_access_info.pl
      -> Displays a user's access rights to networks, vlans,
        and subnets
RUM_netaddr.pl
      -> Returns the network a host belongs to by IP address
saveData.sh
      -> Saves current webdata files
terms-db.pl
      -> Creates data file for statistical variables definitions
trojan-db.pl
      -> Creates Common Trojan/virus ports information database
```

Appendix B – Dynamic IDS User Manual

DYNAMIC INTRUSION DETECTION SYSTEM USERS MANUAL

DYNAMIC IDS RELEASE: 1.0.0

ERKANG ZHENG

16TH DECEMBER 2003

Table of Contents

I. OVERVIEW

A. Introduction to Dynamic IDS

Dynamic Intrusion Detection System (DIDS or Dynamic IDS) is a statistical network intrusion detection system. It uses RUM (Resource Usage Monitor) tool as the underlying packet capturing and sorting engine, based on the flow information that RUM captured to perform dynamic analysis of the network traffic, in order to find anomalies and intrusions. Its current features include:

- *High speed detection in statistical mode*
- *Detection of known intrusions such as DOS attacks and port/host scans*
- *Detection of various forms of unknown/undefined intrusions and anomalies*
- *Detection of viruses and Trojans*
- *Dynamic analysis*
- *Reduction of false positives*
- *Single-point deployment*
- *Minimal administrative overhead*
- *Efficient reporting*
- *Privacy guarantee*
- *Complete open-source*

Dynamic IDS was developed by Erkang Zheng of North Carolina State University. Erkang is currently responsible for all development and maintenance of the system.

B. System Requirements

Hardware Requirements:

For small scale network (with several hundred hosts or less and less than 100Mbps total network traffic):

	Minimum	Recommended
Processor	1 x Pentium III 700MHz+	1 x Pentium III 1GHz+
Memory	256 MB or more	512 MB or more
Hard Drives	2 x 9GB IDE or SCSI	2 x 18GB SCSI
Network Cards	2 x 10/100Mbps	2 x 10/100Mbps

For large scale network (with multiple subnets, thousands of hosts and up to 1Gbps total network traffic):

	Minimum	Recommended
Processor	2 x Pentium III 1GHz+	2 x Pentium 4 Xeon
Memory	1GB or more	2GB or more
Hard Drives	2 x 18GB Ultra SCSI	2 x 18GB Ultra160 SCSI

Network Cards	1 x 10/100Mbps and 1 x 10/100/1000Mbps	1 x 10/100Mbps and 1 x 10/100/1000Mbps

Software Requirements:

- Linux (Red Hat Linux 7.1 or later Recommended)
- Perl 5 or later
- Web Server (e.g. Apache)
- RUM (Resource Usage Monitor)
- Nmap (Network Mapper)
- Browser (e.g. Internet Explorer or Netscape)

Please see section II-A (Dependencies) for more information on the software requirements.

C. Obtaining the Product

The latest release of the product can be downloaded from the development site http://traffic4.cc.ncsu.edu. The site currently has restricted access. Please email Erkang Zheng (erkang@erkang.com) for access or to obtain the software.

D. Contact Information

For all development or maintenance issues, to report bugs, or give feedbacks, please email Erkang Zheng directly using the above email address.

II. INSTALLATION AND CONFIGURATION

A. Dependencies

Dynamic IDS was developed and tested on Red Hat Linux 7.1 with 2.4.x kernel. All its codes are currently written in Perl and the system does not contain operating system specific components. Therefore, it should be able to run on other Linux and UNIX operating systems. Since dynamic IDS requires RUM as the data capturing engine and RUM is not design to run in Windows environment, dynamic IDS currently cannot run in Windows. However, with a substituted capturing engine such as WinDump [29], and certain modifications to the codes, the IDS can easily be ported to run on any version of Windows with Perl installed.

Dependent applications include RUM, Nmap, and a web server. RUM is the system's packet capturing and sorting engine, while Nmap is used for optional host probing. Detailed installation procedures for the two applications can be found in the documentation of its own [31] [34]. A web server such as Apache

[50] is also required, since it uses web interface for monitoring and configuration.

B. System Structure

The system has the following directory structure:

```
$RUM_ROOT
      → bin              (RUM binaries)
      → conf             (RUM configurations)
      → data             (RUM data)
      → ids              (IDS source files, same as ids-src)
      → ids-data         (IDS data for current sample)
      → ids-data.last    (IDS data for previous sample)
      → IDSlog           (General log file for Dynamic IDS)
      → ids-src          (IDS source files)
      → RUMlog           (General log file for RUM)
      → scripts          (Advanced tools and utilities)
      → webdata          (Traffic data)
```

C. Manual Installation

To install Dynamic IDS manually, obtain the distribution file and un-tar it in the same directory as RUM (usually `/usr/local/rum`).

```
% mv dids-1.x.x.tgz /usr/local/rum
% gtar -zxvf dids-1.x.x.tgz
```
or
```
% gzip -cd dids-1.x.x.tgz | tar xvf -
```

D. Automatic Installation

To install Dynamic IDS automatically, first un-tar the distribution then run the install script with RUM's install directory as the only parameter.

```
% gtar -zxvf dids-1.x.x.tgz
```
or
```
% gzip -cd dids-1.x.x.tgz | tar xvf –
% cd dids-1.x.x
% ./install.pl –d $RUM_ROOT
```

The will install Dynamic IDS into the proper location. If no location is specified, the install script will assume RUM is installed at `/usr/local/rum` if such directory exists.

E. Initializing the System

The system need to be initialized before the first run. To initialize it, run the initialization script as below.

```
% /usr/local/rum/ids/initialize.pl
```

Alternatively, the system can be initialized at the same time of installation. Another argument is specified to achieve this.

```
% ./install.pl –d $RUM_ROOT -i
```

F. Tuning the System

Dynamic IDS does not require any manual configuration in order to work. However, some simple tuning of the system can be done to obtain more accurate results. The most effective and recommended tuning is the end-host database. Since the end-host database is created during the initial automatic system set up against some sample data, it is very likely that the end-host database may contain inaccurate data. For example, any host that is suppose to be a client can be mistakenly logged as a server if a "legitimate" server is running at the time of initial setup and vice versa. This can be easily correctly by configuration the host database from the system configuration web interface. Once all the hosts are correctly logged in the database, the system can perform analysis more accurately.

Please refer to section IV on configuring the system.

G. Running and Scheduling the System

To run the system once, execute the main execution script from command line. The script can be found under the `ids` directory.

```
% cd /usr/local/rum/ids
% ./IDSengine.pl
```

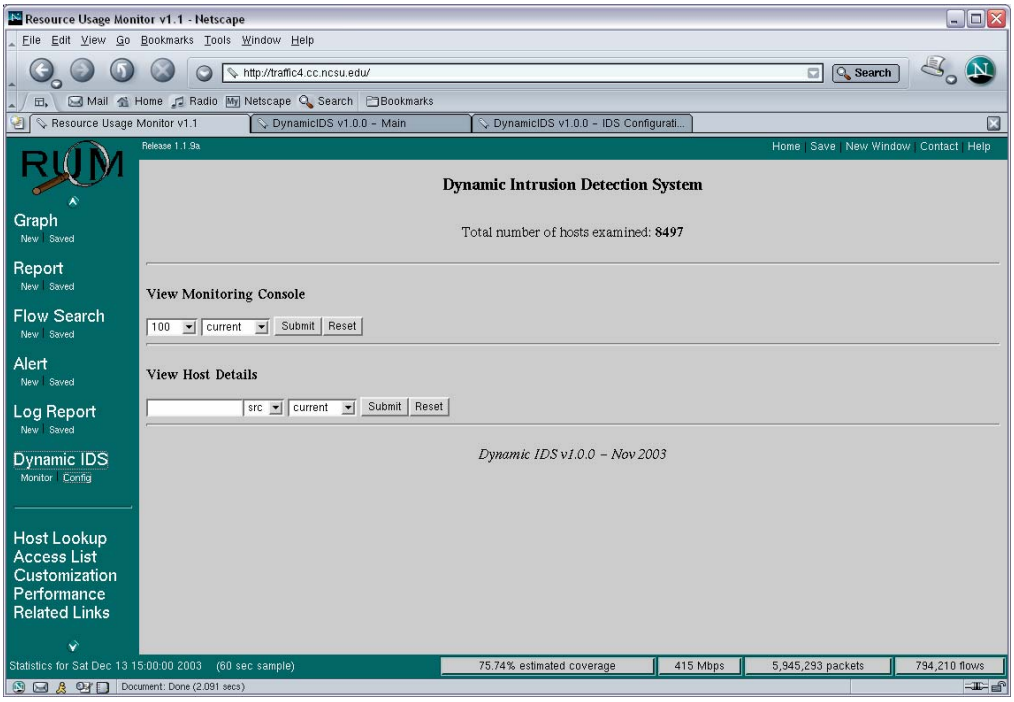
The main execution script can be added to root's cron jobs so that it will run continuously. The time between each run depends on the load of the network and it needs to be greater than the total time required for the system to run once. Thirty minutes is usually a good try. To find out the exact time required for the system to complete one cycle, execute the following command:

```
% date; ./IDSengine.pl; date
```

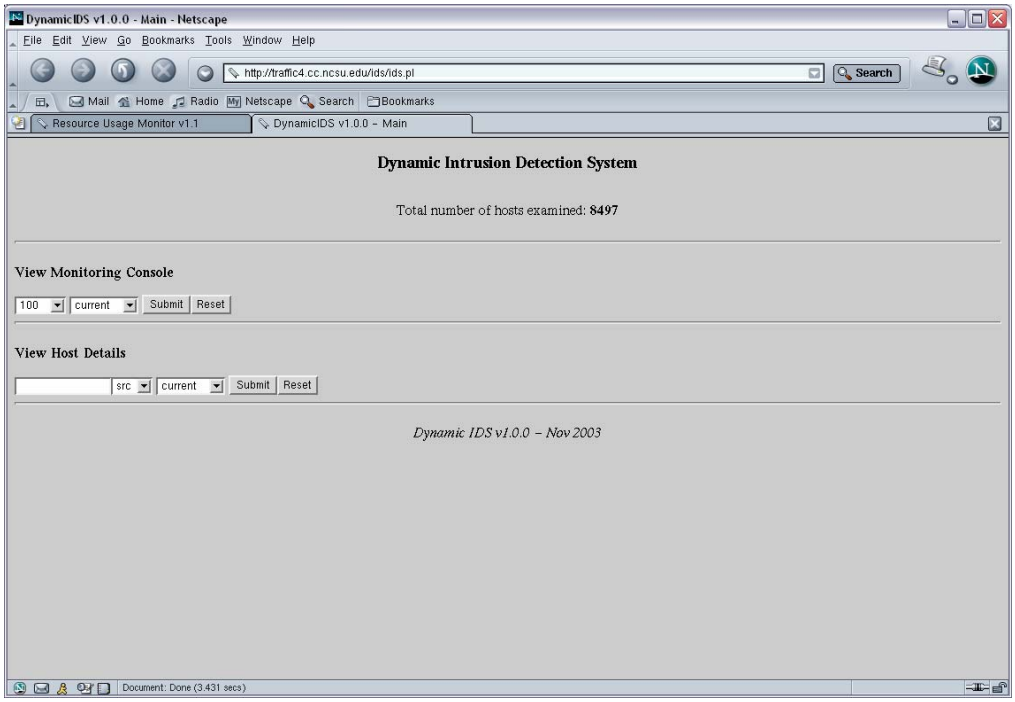
III. NETWORK MONITORING

A. Monitoring Web Interface

To launch the system's web-based user interface, open up a browser and type in the server's name or IP address. After entering the correct username and password, the page should load. Dynamic IDS is integrated into RUM's web interface, as shown in the screenshot below.

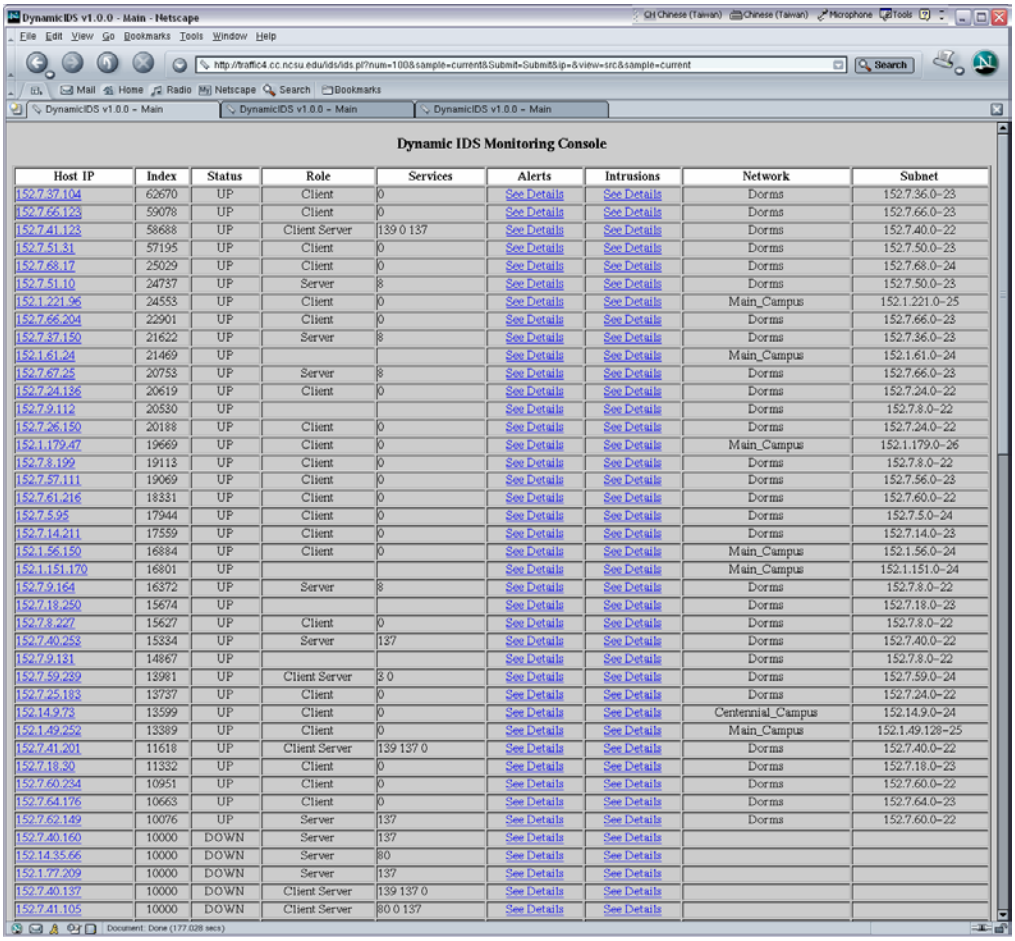


The monitoring interface is made up with three main sections. They are Monitoring Home, Main Console, and Host Details. The Console and Host Details page can be invoked from Monitoring Home.



B. Monitoring Console

The Main Monitoring Console page displays the selected number of hosts in a table format. For each host, brief information such as Index, Status, Role, Services, Network, and Subnet are displayed in the columns. The hosts are sorted based on their index value, with the highest ones on top. This way, the hosts that are considered more serious and problematic attracts more attention from the administrator. It conveniently provides the administrator an investigation list.

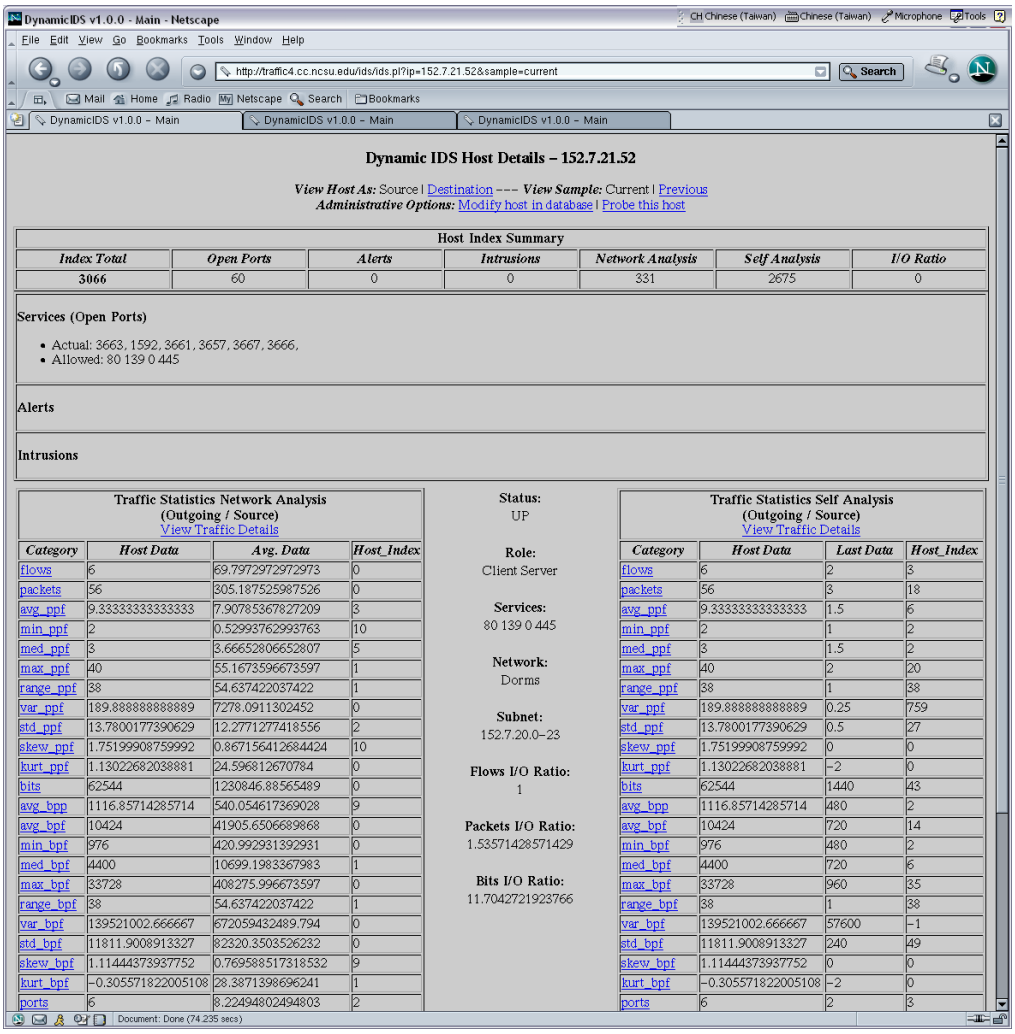


Dynamic IDS Monitoring Console

Host IP	Index	Status	Role	Services	Alerts	Intrusions	Network	Subnet
152.7.37.104	62670	UP	Client	0	See Details	See Details	Dorms	152.7.36.0–23
152.7.66.123	59078	UP	Client	0	See Details	See Details	Dorms	152.7.66.0–23
152.7.41.123	58688	UP	Client Server	139 0 137	See Details	See Details	Dorms	152.7.40.0–22
152.7.51.31	57195	UP	Client	0	See Details	See Details	Dorms	152.7.50.0–23
152.7.68.17	25029	UP	Client	0	See Details	See Details	Dorms	152.7.68.0–24
152.7.51.10	24737	UP	Server	8	See Details	See Details	Dorms	152.7.50.0–23
152.1.221.96	24553	UP	Client	0	See Details	See Details	Main_Campus	152.1.221.0–25
152.7.66.204	22901	UP	Client	0	See Details	See Details	Dorms	152.7.66.0–23
152.7.37.150	21622	UP	Server	8	See Details	See Details	Dorms	152.7.36.0–23
152.1.61.24	21469	UP			See Details	See Details	Main_Campus	152.1.61.0–24
152.7.67.25	20753	UP	Server	8	See Details	See Details	Dorms	152.7.66.0–23
152.7.24.136	20619	UP	Client	0	See Details	See Details	Dorms	152.7.24.0–22
152.7.9.112	20530	UP			See Details	See Details	Dorms	152.7.8.0–22
152.7.26.150	20188	UP	Client	0	See Details	See Details	Dorms	152.7.24.0–22
152.1.179.47	19669	UP	Client	0	See Details	See Details	Main_Campus	152.1.179.0–26
152.7.8.199	19113	UP	Client	0	See Details	See Details	Dorms	152.7.8.0–22
152.7.57.111	19069	UP	Client	0	See Details	See Details	Dorms	152.7.56.0–23
152.7.61.216	18331	UP	Client	0	See Details	See Details	Dorms	152.7.60.0–22
152.7.5.95	17944	UP	Client	0	See Details	See Details	Dorms	152.7.5.0–24
152.7.14.211	17559	UP	Client	0	See Details	See Details	Dorms	152.7.14.0–23
152.1.56.150	16884	UP	Client	0	See Details	See Details	Main_Campus	152.1.56.0–24
152.1.151.170	16801	UP			See Details	See Details	Main_Campus	152.1.151.0–24
152.7.9.164	16372	UP	Server	8	See Details	See Details	Dorms	152.7.8.0–22
152.7.18.250	15674	UP			See Details	See Details	Dorms	152.7.18.0–23
152.7.8.227	15627	UP	Client	0	See Details	See Details	Dorms	152.7.8.0–22
152.7.40.258	15334	UP	Server	137	See Details	See Details	Dorms	152.7.40.0–22
152.7.9.131	14867	UP			See Details	See Details	Dorms	152.7.8.0–22
152.7.59.239	13981	UP	Client Server	8 0	See Details	See Details	Dorms	152.7.59.0–24
152.7.25.183	13737	UP	Client	0	See Details	See Details	Dorms	152.7.24.0–22
152.14.9.73	13599	UP	Client	0	See Details	See Details	Centennial_Campus	152.14.9.0–24
152.1.49.252	13389	UP	Client	0	See Details	See Details	Main_Campus	152.1.49.128–25
152.7.41.201	11618	UP	Client Server	139 137 0	See Details	See Details	Dorms	152.7.40.0–22
152.7.18.30	11332	UP	Client	0	See Details	See Details	Dorms	152.7.18.0–23
152.7.60.234	10951	UP	Client	0	See Details	See Details	Dorms	152.7.60.0–22
152.7.64.176	10663	UP	Client	0	See Details	See Details	Dorms	152.7.64.0–23
152.7.62.149	10076	UP	Server	137	See Details	See Details	Dorms	152.7.60.0–22
152.7.40.160	10000	DOWN	Server	137	See Details	See Details		
152.14.35.66	10000	DOWN	Server	80	See Details	See Details		
152.1.77.209	10000	DOWN	Server	137	See Details	See Details		
152.7.40.137	10000	DOWN	Client Server	139 137 0	See Details	See Details		
152.7.41.105	10000	DOWN	Client Server	80 0 137	See Details	See Details		

C. Host Details

The Host Details page can be reached by entering its IP address from the Monitoring Home page, or by clicking on the link from the Console Table as shown above. The details page contains all the information that the administrator needs to know about this host. It first displays the total index value, as well as the index values generated corresponding to different categories – Ports, Alerts, Intrusions, Network Analysis, Self Analysis, and Input/Output Ratio. Then lists of communicating ports, alerts, and intrusions are shown. If any of the ports matches the ones defined in the Trojan Ports database, that particular port will have a red link, so the administrator can simply click on it to bring up a description for that Trojan port. Finally, the host's status, role, allowed services, network, subnet, and input/output ratios are displayed in the middle of the page. On its left the page shows details for network analysis, whereas details for self analysis are shown on the right.



D. Viewing Options

In addition to the host information, the host also presents several different viewing options and administrative options. The view options can be found near the top of the page, right under the Host Details title. The viewing options include View Host as Source or Destination, and View Current or Previous Sample.

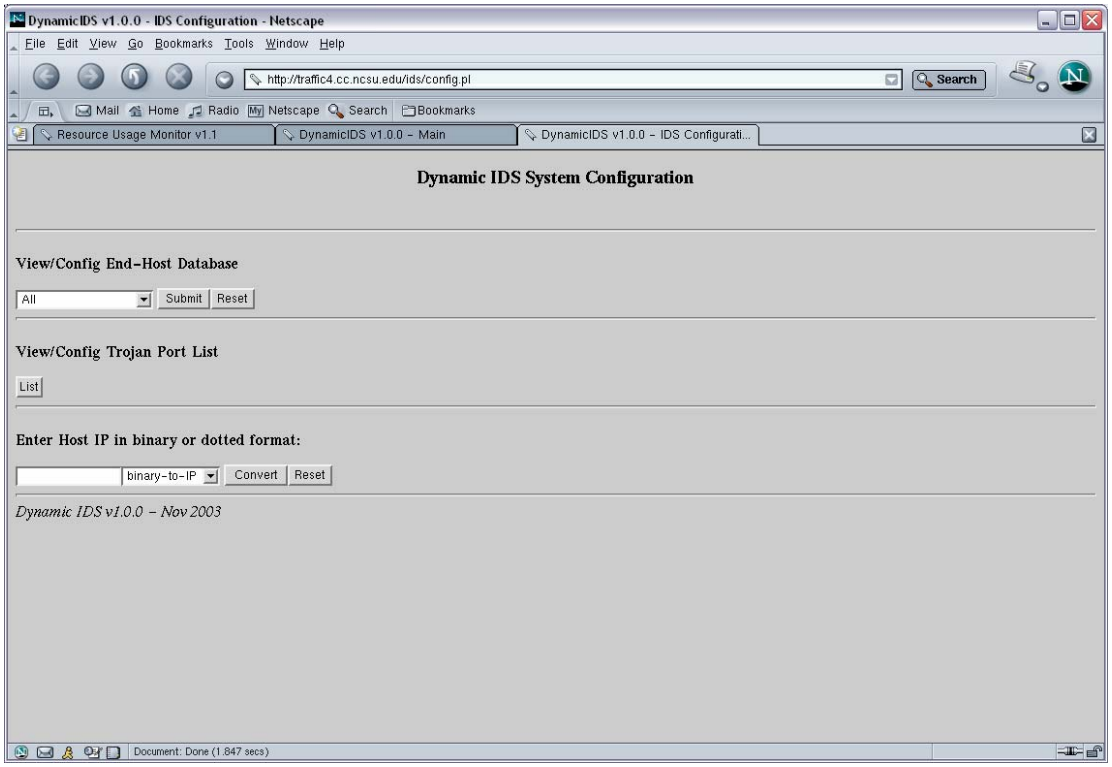
E. Administrative Options

In a similar format as to the viewing options, the administrative options are right underneath, which include modifying the host's information in end-host database, and actively probing the host to find open ports. The modifying option is explained in detail in the configuration section below. Probing of the host is done by implementing Nmap [34] in the execution, and presenting Nmap output onto the browser.

IV. SYSTEM CONFIGURATION

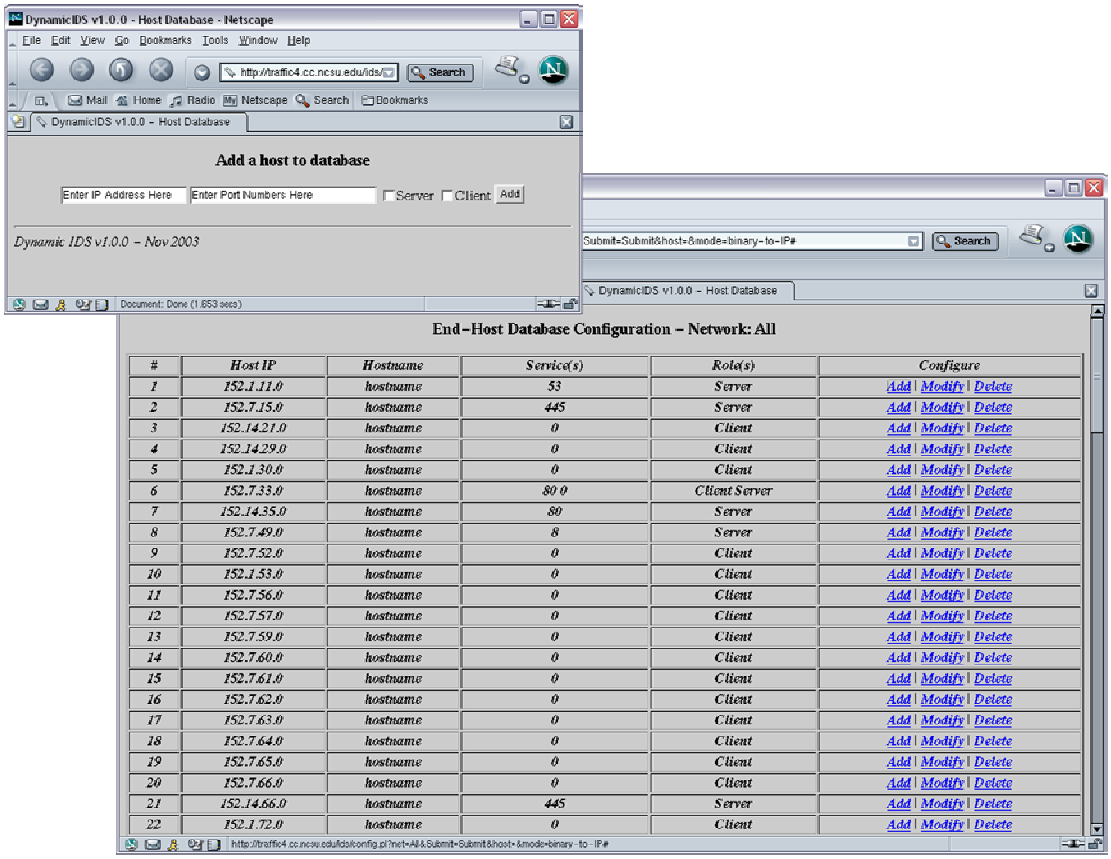
A. Configuration Interface

The system configuration page currently offers only two configuration options. One is end-host database viewing and configuring, the other is Trojan ports database. Similar to monitoring, both options can be invoked from the configuration home page.



B. End-Host Database

The host database configuration option lists the currently hosts in the end-hosts database by network. It lists the IP addresses and host names, as well as each host's allowed services and pre-defined client/server roles. It also gives options to modify any existing host, or add new host to the database.



C. Trojan Port List

The Trojan Port List is presented in a similar fashion as the end-host database configuration, where the page lists all the defined Trojan ports and corresponding description in a table.

D. IP Conversion Utility

In addition to the configuration options, an IP address utility is also provided. The utility can convert back and forth between the dotted decimal IP format and the binary format used in Perl Storable database files. It allows the administrator to view individual entries of the database files from command line and be able to figure out the corresponding IP address easily.

V. DEVELOPMENT INSTRUCTIONS

There is currently no specific development instructions or API available, but the product is undergoing further development and routine maintenance. For specific development instructions, please contact Erkang Zheng directly. To obtain a whitepaper on the technology behind dynamic intrusion detection system, go to http://www.erkang.com/papers/dids.pdf.