Reclaiming one's bandwidth: Dynamic filtering of traffic based on packet payload content

Barry Irwin <bvi@rucus.ru.ac.za>
Rhodes University, Grahamstown. South Africa
June 2000

Abstract

Dynamic filtering of IP traffic based on contents of packet payloads provides a means for monitoring and controlling a number of Internet services which prove difficult to manage by traditional means. Constituents of the portion of Internet traffic not attributable to traditional services can be identified and quantified, even if applications transfer data on a highly dynamic range of ports. An initial implementation of a dynamic filtering system under FreeBSD is also evaluated and strengths and weaknesses identified.

1. The need for dynamic filtering

In recent months applications have been developed to enable users to easily share a variety of file type easily over the Internet. Many of these build on the groundwork for sharing of mp3[1] media files pioneered by Napster[10].

Soon after its release, Napster became immensely popular and, due to the sheer volume of users making use of the application brought many institutions Internet connections to a standstill. The nature of mp3 files are that they are large, with a typical file consuming between three and four megabytes of disk space, depending on the encoding options chosen when the file was initially compressed.

Following on from this popularity, other authors produced similar software, such as Imesh[4], cuteMX[2] and Gnutella[3] that was extended to potentially be able to share any file type. Of particular note is the development of Gnutella which utilises a protocol designed to be difficult to filter [6]. Many of these applications are also able to operate if one party in the peer−to−peer connection is behind a firewall. [2,3,4,6]

Not all these applications work on the basis of connecting to some form of central server which is used for collating files being offered by connected clients. Napster uses a variety of ports and a number of servers which are tried when a connection is made. Thus, in order to block such traffic, firewall administrators would have to either block all outgoing connections, or spend time keeping up to date with server lists and block access to those. This task could become quite time consuming, yet indiscriminately blocking all possible ports that could be used by such applications would soon result in legitimate services being disrupted. Gnutella as feature if its design, allows any node on the network to act as an uplink server for new new nodes joining the network.

File sharing utilities are not the only cause for concern, monitoring of other network traffic of dubious intent may also be of use. Many 'warez' servers (servers containing illegal copies of software for download) make use of Internet Standard protocols such as FTP and HTTP, but run these services on non standard ports. These are usually undesirable either because of the content contained, or the lack of accounting information ('audit trail') for this traffic, which is the common case where a proxy server has been deployed.

These same methods can be used for monitoring connections to an external proxy server which would allow users to bypass any local proxy restrictions. Similarly with external SMTP, POP, IMAP and News servers. This could be regarded as a security measure −− especially if the organisation has intellectual property which it wishes to prevent being disseminated via e−mail or other electronic means.

A solution to the monitoring and filtering of these traffic types can be implemented with methods commonly used by Intrusion Detection Systems (IDS), where traffic entering a LAN is scanned for various signatures indicating a likely hostile probe or attack against machines on the network. An open source example of such an IDS is Snort[13,14] written by Martin Roesch and which comes with a rich library of traffic signatures.

Traffic is inspected at the packet level, and packets with a payload matching certain predefined rules is reported as suspicious. Optionally the system can generate an appropriate firewall rule for the specific connection matching a rule such as just adding an accounting entry, constricting traffic bandwith or denying the connection. Use of such a system allows legitimate traffic running on 'suspect ports' through with no interference.

1 MPEG Layer 3 Audio Compression

2. Implementation

A system can be implemented using ngrep[12] to gather packets off the network and apply preliminary filtering by means of packet capture logic (as implemented in the Lawrence Berkeley National Laboratory pcap library, and described in the tcpdump(1) man page on unix systems) and regular expressions. This output is then further processed by an encapsulating Perl script which performs logging, and generates appropriate firewall rule sets for each matching class of traffic. An example of a script implementing this can be found at http://rucus.ru.ac.za/~bvi/utils/sting.html

This preliminary proof of concept is implemented using ngrep rather than snort due to the fact that the protocols under investigation are mostly text based and also the system did not require many of the features offered by snort.

Appropriate regular expressions for protocols can be generated by examining packet dumps of a captured connection, and scanning for consistent data that can be encapsulated in a regular expression. These regular expressions are then used by ngrep for filtering general network traffic from the traffic under investigation. A useful tool for performing this examination is Ethereal[1]. Examples of regular expressions for matching common protocols are illustrated in Table 1.

Protocol	***Regular Expression***
FTP	(^PORT .+,.+,.+,.+,.+,) ^PASV
HTTP	(^GET .+ HTTP/1) (^POST .+ HTTP/1)
Napster	FILENAME CONTAINS \".+\" MAX_RESULTS
Imesh	GET http://www.imesh.com/
Gnutella	^GNUTELLA CONNECT/

Table 1: Regular expressions matching common protocols

3. Implementing dynamic filtering as a means of traffic management

The need for traffic management can arise due to a number of factors, one of the foremost being that an organisation's Internet link is becoming saturated. Other reasons can be for security reasons in which case the management comes close to traditional firewalling, or to prevent certain types of traffic for legal reasons.

The protocols discussed are known to major consumers of bandwidth usage. University of California, Berkeley cites a 61% use of the bandwidth by Napster traffic[17]. Vanderbilt University had similar figures with 45.26% of traffic being accredited to outbound napster traffic and 8% incoming [19]. In South Africa , the University of Cape Town also experienced problems with Napster and Imesh saturating their internet link. These figures illustrate that the problem with bandwidth saturation is widespread even on high speed links [16,17,19]. Many institutions have limited funding available for financing their Internet connections and consequently need to make optimal use of the infrastructure in place before upgrading can be considered.

The best place to implement a dynamic filtering system is between the organisational LAN and the router connecting the organisation to the Internet. This can be done with the detection PC also acting as a gateway for traffic, and hence able to firewall traffic directly, or as a 'passive' system sniffing network traffic and providing logging, but implementing the firewalling rules on another system or router. The actual physical and logical implementation would depend largely on the topology of the network concerned and network policies within the organisation.

Once the ports commonly used by a protocol have been identified, various techniques can be used to visualise the traffic attributable to these either via RMON type agents or direct accounting and display using existing utilities such as MRTG[5]. Data can also be archived for trend analysis over a period of time, or possibly for inclusion into some form of 'expert system' which can be used to change filtering policies based on available bandwidth. The rules for the dynamic filtering are not required to be static, different rule sets could be implemented depending on the amount of available bandwidth or the time of day, or even the day of the week.

The sting script developed, was found to perform well under the test load, utilising less than 2% of the CPU on the test machine (PIII 500), with the test link being run at 10Mbit. This should scale to operating efficiently on a lower powered machine, considering that the majority of Internet links in South Africa are in the sub−1Mbit range. Efficiency could be greatly improved by rewriting the system as a single application in C.

An example of a situation where dynamic filtering could have been used to provide a relatively easy solution can be

found in the Napster issue arising out of the mid April 2000 lawsuit opened by heavy metal band Metallica against Napster Inc. for the spread of pirated music[8,11]. Included in this suit were the Universities of Southern California and Indiana, who were included by virtue of the fact that they had not taken sufficient measure to prevent the use of Napster on their respective campuses[7,9]. The 'required' blocking was difficult to implement due to the multiple servers used by the Napster client. The Napster protocol does however have a distinctive signature, and this could have been used in a dynamic filtering system to automatically generate a list of Napster servers to be blocked. The alternative which was implemented, required administrators to manually generate this list based on empirical testing, or other sources such as server listings as provided by napigator [18].

4. Feedback and monitoring

The system is designed to be both flexible and extendable. Traffic does not necessarily have to be denied but can be logged for statistical or informational purposes. This data can be used for higher level decision making and capacity planning, by providing some kind of quantification of what contributes to the often significant but enigmatic 'other traffic' on a network. By analysing the matches reported by the system more permanent filters can be developed. An example of this is the collection of data about the various IP network blocks on which Napster servers are located. Filtering these address blocks at router or firewall level rather than having rules dynamically generated, yet the dynamic generation will still catch services running on other IP addresses and ports, easing the task of the administrator.

'Port hopping' applications such as Gnutella are also dealt with effectively. While the protocol is able to run on a myriad of ports in order to make it difficult to block, the actual exchange between clients remains the same and can be detected, thus effectively negating the multiple port functionality.

5. Caveats and further extensions

Sophisticated encryption tools which allow the tunnelling and encapsulation of IP traffic within an encrypted stream are freely available today on a variety of operating systems. Use of such encryption would negate the ability of a dynamic firewalling tool to detect nefarious traffic. Being able to encrypt a stream would still in the majority of cases require access to an external host on which the traffic is decrypted and forwarded to its destination. End to end encryption of the client−server or peer−to−peer link could pose more problematic.

A firewall system such as described above could potentially have a denial of service performed against it by a user forging malicious matching packets. Care thus needs to be taken both to limit the range of firewall rules that can be added and to ensure that rules should be expired after a given period of time since the last match.

The system as currently implemented only works effectively on protocols which have some form of repetitive identifier in plain text. Binary protocols (such as RealAudio) could theoretically be blocked using similar methods, but binary signatures would have to be found and an alternate capture program developed [15]. A possibility for this would be to used an identification system similar to that used by the file(1) command on unix systems, but identifying file types by means of a magic(5) number.

A further weakness is that the system is only able to act on traffic that is seem by the host machine. Thus on a fully switched network, unless the filtering system is installed on a monitor port which receives all traffic, the system becomes ineffectual. Implementing the system on a transparent bridge or gateway machine would be the preferable method for achieving maximum benefit. In the same vein, care needs to be taken in defining the traffic filter used by the pcap library, both to exclude extraneous traffic, but also to ensure that traffic if relevance is included.

6. Conclusion

Dynamic filtering of IP traffic provides an important tool in a network administrators toolkit for diagnosing network performance issues, as well as for implementing and enforcing appropriate policy on the network.

References

1. Combs G, Ethereal: A network protocol analyser. 2000. Online: http://ethereal.zing.org

2. CuteMX Online: http://www.cutemx.com

3. Gnutella Online: http://gnutella.wego.com

4. Imesh Online: http://www.imesh.com

5. Irwin, BVW, A per protocol approach to bandwidth monitoring and management of IP traffic, March 2000. Online: http://rucus.ru.ac.za/~bvi/research/papers/per−protocolbandwidth.ps

6. Irwin, BVW, An Analysis of common high bandwidth protocols(work in progress), April 2000. Online: http://rucus.ru.ac.za/~bvi/research/papers/proto_analysis.txt
7. Metallica Club,The. Napster Lawsuit FAQ, May 2000. Online**:** http://www.metallica.com/news/2000/napfaq.html
8. Metallica Club,The. Metallica sue Napster, May 2000. Online: http://www.metallica.com/news/2000/napsterfacts.html
9. Metallica Club, The. Metallica Files Suit Against Napster, University of Southern California, Indiana University, April 2000. Online: http://www.metallica.com/news/2000/000413a.html
10. Napster.com Online:http://www.napster.com
11. Napster Inc. Information About Metallica's Request To Disable Napster Users, May 2000. Online: http://www.napster.com/metallica−notice.html
12. Ritter J, Network Grep. Online: http://www.packetfactory.net/Projects/Ngrep/
13. Roesch M, Snort homepage. Online: http://www.clark.net/~roesch/security.html
14. Roesch M, Snort − Lightweight Intrusion Detection for Networks, USENIX LISA '99 November 1999. Online: http://snort.safenetworks.com/lisapaper.txt
15. Roesch M, Writing Snort Rules, January 2000. Online http://snort.safenetworks.com/snort_rules.html
16. Stenger, R, (CNN) Campuses seek compromise over popular bandwidth hog , April 2000. Online: http://www.cnn.com/2000/TECH/computing/03/01/napster.ban/
17. Talcott, S, Restriction of Internet access at UC−Berkeley to continue.(Daily Californian),March 23, 2000. Online: http://lists.fsu.edu/pipermail/nolenet/2000−March/002241.html
18. thirty4 interactive LLC, Napigator Online: http://www.napigator.com
19. Vanderbilt University, Napster and bandwidth usage, 2000. Online: http://www.vanderbilt.edu/resnet/napster.html