Protocol Identification via Statistical Analysis (PISA)

Black Hat 2007

Rohit Dhamankar and Rob King, TippingPoint Technologies

Abstract

A growing number of proprietary protocols are using end-to-end encryption to avoid being detected via network-based systems performing Intrusion Detection/Prevention and Application Rate Shaping. Attackers frequently use well known ports that are open through most firewalls to tunnel commands for controlling zombie systems.

This paper shows that a framework is indeed possible to identify encrypted protocols or anomalous usage of well known ports. The framework relies on performing statistical analysis on protocol packets and flows, and uniquely maps each protocol in a 10-dimensional space. Clustering algorithms can be applied to reasonably accurately identify a wide variety of protocols.

This novel approach provides network and security administrators a powerful tool to use in enforcing traffic policy, even when users are actively attempting to evade these policies.

1 Introduction

As the processor speeds have increased and fast performing processors have become a commodity, there has been a rise in the use of encryption for a variety of protocols. The race to encrypting more traffic has also been fueled by various compliance laws that are being enforced on organizations. As a result, desirable as well as undesirable traffic from the perspective of an organization, is undergoing encryption. This poses a great challenge to network-based prevention/detection and rate-shaping appliances as unless the appliance can proxy encrypted traffic for all the protocols running in a network, it is virtually impossible to detect the nature of traffic flowing in the network.

Peer-to-Peer protocols (P2P), that have become popular since 2002, have evolved from using clear-text HTTP like protocols and proprietary protocols to encrypted protocols to avoid being detected. P2P traffic chews up network bandwidth; Further, studies have shown that increasing bandwidth does not solve the problem as P2P traffic ramps up to consume more of the available bandwidth. The following graphs shows the P2P traffic (KaZaa, eDonkey and WinMX) consuming a large percentage of bandwidth instead of the mission-critical traffic like Oracle.

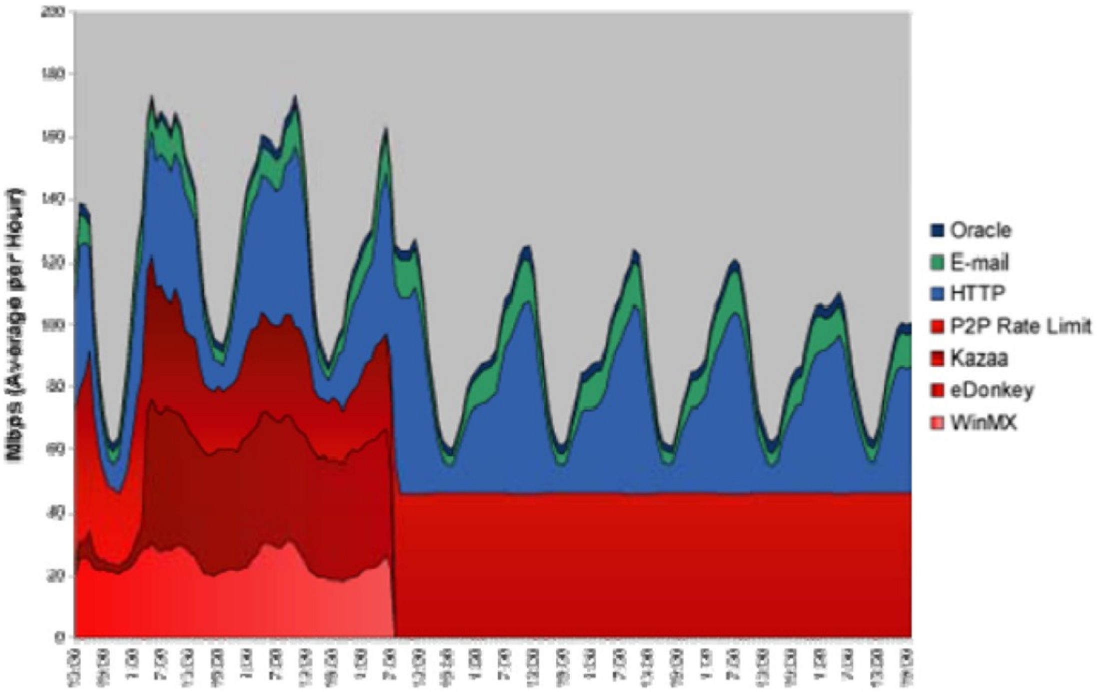


Figure 7: This graph details an eight-day period. Each peak represents the peak traffic during that day. All of the red data represents P2P traffic, which is rate-limited to 45Mbps on day three. The data in blue and green represents mission critical traffic: Oracle, E-mail, and HTTP. It is not rate limited receiving the full bandwidth advantage of the pipe.

The next-generation P2P protocols like Skype have adopted encryption and obfuscation to avoid being detected by network appliances. This is in sharp contrast to the first generation P2P protocols like KaZaa or eDonkey.

Here is a request from a client using KaZaa program for downloading a file from its peer.

```
172.16.5.20:1277 -> 24.141.247.100:2785
GET /.hash=1b48a19af2dab74f73990f6336ad16dac40ecffe HTTP/1.1
Host: 24.141.247.100:2785.
UserAgent: KazaNov  3 2002 20:29:03
X-Kazaa-Username: bigtex7501
X-Kazaa-Network: KaZaA
X-Kazaa-IP: 172.16.5.20:2049
X-Kazaa-SupernodeIP: 24.79.124.178:3783
Range: bytes=2772358-4739071
Connection: close
X-Kazaa-XferId: 4170
X-Kazaa-XferUid: w25IZcM3edfD3p1uuk2p2ogQzPGyLPPETAFspJH6P8A=
```

As can be seen from the request, the KaZaa protocol is based on HTTP protocol with KaZaa-specific fields in the HTTP header.

Let’s consider another example of eDonkey client requesting a file download from its peer. The packet content requesting the download looks like:

```
rrcs-24-153-164-134.sw.biz.rr.com.4662 > pD9E620B3.dip.t-dialin.net.3939
P [tcp sum ok] 71:130(59) ack 100 win 17325 (DF) (ttl 128, id 18580, len 99)
0x0000   4500 0063 4894 4000 8006 fa47 1899 a486        E..cH.@....G....
0x0010   d9e6 20b3 1236 0f63 cc8f bdf4 3a6d 75b6        .....6.c....:mu.
0x0020   5018 43ad 4fa0 0000 e336 0000 0059 56ea        P.C.O....6...YV.
0x0030   7f9b 8db9 d70a ef91 b390 c3f5 13a8 2300        ..............#.
0x0040   5465 7272 7920 436c 6172 6b20 2d20 4120        Terry.Clark.-.A.
0x0050   4c69 7474 6c65 2047 6173 6f6c 696e 652e        Little.Gasoline.
0x0060   6d70 33                                        mp3
```

e3 indicates that it is eDonkey packet, the next 4 bytes (0x36000000) are the length of the packet in little-endian form. So, 0x36 = 54 bytes is the length of the rest of the eDonkey data starting with the byte 0x59. The name of the file being downloaded "A Little Gasoline.mp3" is in clear-text. So, even the proprietary eDonkey protocol can be decoded by looking closely at the packets on the wire.

The next-generation Skype protocol uses random ports for communication and the data is encrypted and obfuscated. Hence, no detection has been currently possible for this protocol using the "signature-based" approach.

In addition to the encrypted P2P protocols, botnets are increasingly using encryption for avoiding detection.

"Botnet watchers are also seeing a trend toward stronger encryption. Encryption is used by attackers to ensure that bots added to the network are in fact legitimate, as opposed to being nodes belonging to researchers working to infiltrate a botnet and block it or take it down." eWeek Report, April 2007

Identifying encrypted traffic as accurately as possible without a large processing overhead has been the primary motivation for this study

2 Proposed Solution

A wise statistics professor in the college had once quoted – "Statistics is like a bikini that reveals what is interesting and hides what is vital." We embarked on a mission to see if studying the statistical properties of various protocols would reveal anything interesting!

Having worked with a large number of protocols, we were clear about some of the initial properties to pick - the packet sizes used in communication being one of the obvious choices. There are interactive protocols that are very chatty by nature such as telnet or ssh. These protocols typically use small packets in either direction. This is in contrast to protocol such as HTTP(S), which is an example of imbalanced request-response protocol. For HTTP(S) the requests are typically small and the responses are much larger. So, we decided to add another property – traffic difference in request and response for this study. Yet another property that was obvious was the average time-lag between the packets.

We also wanted to characterize the nature of bytes seen in protocols to distinguish protocols that use, for instance, only ASCII characters from the ones that are truly encrypted. The metric used for this purpose is the "entropy" of the protocol.

"Shannon Entropy" is used in data communications to figure out how the best transmission rates can be obtained. The Shannon entropy takes into account an outcome x_i and the probability of the outcome $p(x_i)$. The final value is computed by summing up the product of probability and the log of the probability over all possible outcomes.

$$S = \sum p(x_i)\log p(x_i)$$

Where $p(x_i)$ is the probability of occurrence of element x_i

Let's take a few examples and see why the Shannon Entropy gives us a notion of randomness for the packet data.

Example 1:
If a protocol contains only the letter "a" in all its packets, the probability of occurrence of "a" is 1. Hence, Shannon Entropy is 0 (log 1 = 0). This agrees with the notion that the data for this protocol is not random at all.

Example 2:
Now, say the protocol has two characters "a" and "b" occurring equally in the packets. So, the probability of occurrence of "a" as well as "b" is 0.5. Shannon Entropy in this case is: $-2*1/2*\log(1/2) = 1$.
So, the data for this protocol is more random than the previous example.

One may ask the question as to what is the maximum Shannon Entropy possible for a protocol. The entropy will reach its maximum limit when all the characters from 0x00 through 0xff are present equally likely i.e. the probability of p(0x00)=p(0x01)=p(0x02)…=p(0xff) is 1/256. This case yields us the maximum Shannon Entropy of 8 for any protocol.

2.1 PISA Axes

We chose to represent the protocol traffic we were examining in the 10-dimensional space with the following co-ordinates:

- Average Packet Size to client
- Average Packet Size to server
- Average Time for client responses
- Average Time for server responses
- Standard Deviation of Packet Size to client
- Standard Deviation of Packet Size to server
- Standard Deviation of Time for client responses
- Standard Deviation of Time for server responses
- Traffic difference between server and client
- Shannon Entropy for the protocol

3 Experimental Data and Results

Since Skype was intriguing as a protocol, we initially focused on UDP traffic obtained from Skype audio and video by making various Skype calls in the US and abroad (mostly India).We also obtained UDP traffic from the corporate LAN environment consisting of other protocols.

Traffic composition for the training set:
Approximately 50 Gigabytes of packet captures for:

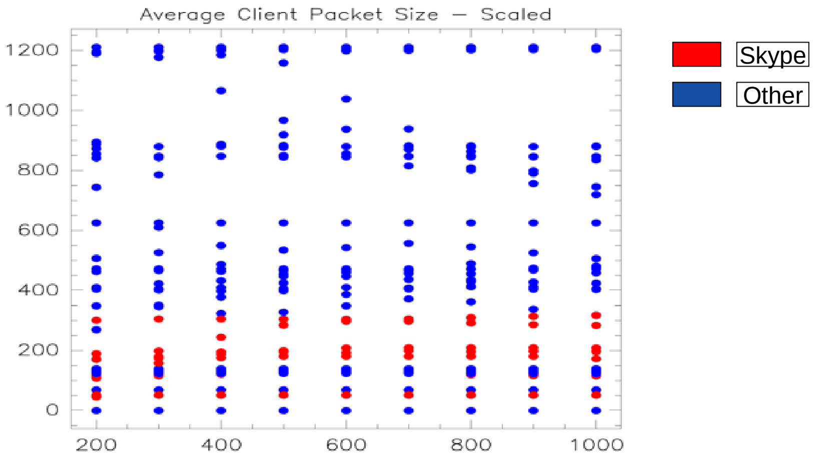
- Skype Voice data
- Skype Video data
- Gizmo Voice data
- UDP DNS Traffic
- NFS Traffic
- NTP Traffic
- NetBIOS Traffic
- Other UDP Traffic

The only caveat is that the traffic was mostly collected from LAN and broadband environments and not dial-up links. Hence, the inter-packet delay and packet sizes are more skewed for Ethernet environments.

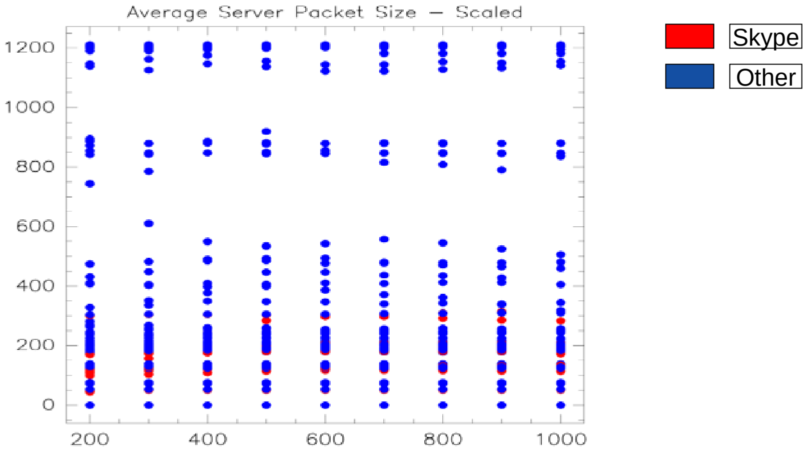
The first question we asked ourselves is: do the *statistical* characteristics of a particular protocol stabilize after a certain number of packets?

To answer this question, we have plotted the graphs of Skype and other UDP traffic for the 10 PISA co-ordinates shown below.

Graph 1: Average Client Packet Size

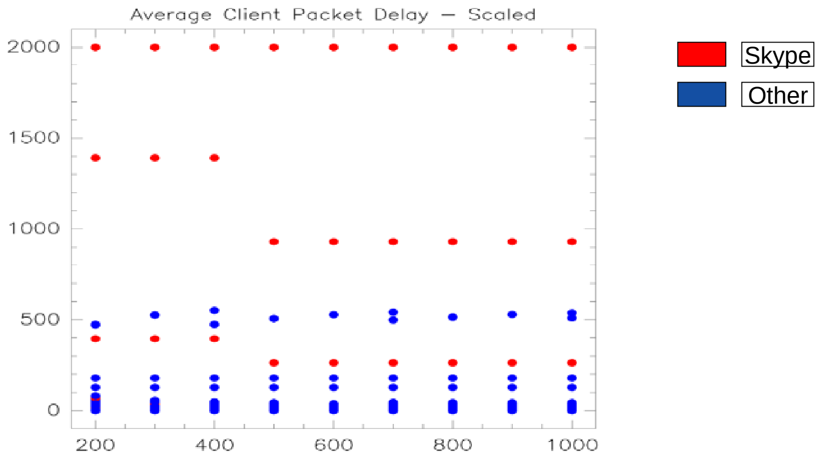


Graph 2: Average Server Packet Size



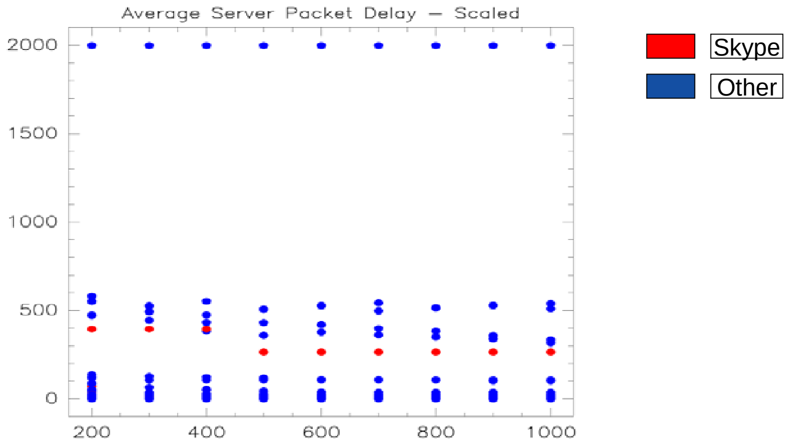
Graph 3: Average Client Response Delay

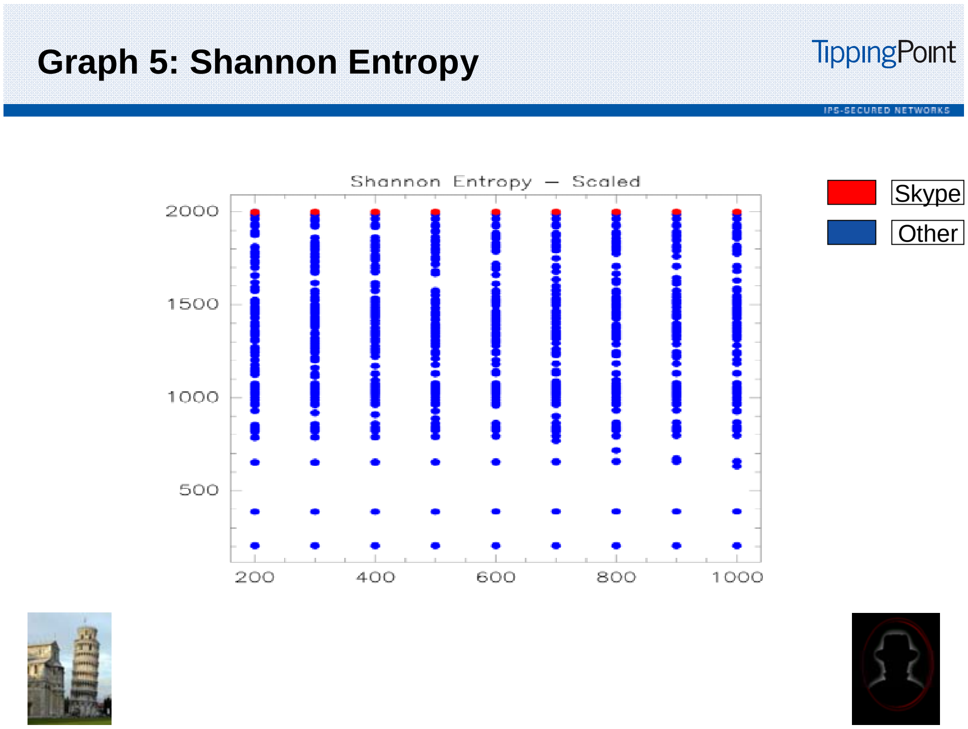
TippingPoint
IPS-SECURED NETWORKS



Graph 4: Average Server Packet Delay

TippingPoint
IPS-SECURED NETWORKS





Graph 5: Shannon Entropy

You will notice from the graphs that by about 600th packet, the Skype traffic characteristics stabilize. So, the Skype traffic can be potentially identified in about 1.5 seconds of the Skype call starting. Also, note the fact that the Skype traffic is truly random from other protocols in the mix namely DNS, NFS etc. This is indicated by the Skype traffic reaching the maximum Shannon Entropy even by 200th packet.

3.1 Clustering in 10-dimensional PISA space

The scales for the co-ordinates of the PISA space are all in different units – milliseconds for the inter-packet delay, number of bytes for the packet sizes. In addition, certain co-ordinates have a maximum like packet size and Shannon Entropy, others do not. For the initial analysis, we decided to scale each co-ordinate by assuming a certain maximum for all the axes, so that equal weight could then be provided for each co-ordinate of the PISA axes.

Each sample for Skype and other protocols was represented in the 10-dimensional space by computing the Euclidean distance of the scaled co-ordinates. After plotting the samples, we applied the K-means clustering algorithm described at http://www.elet.polimi.it/upload/matteucc/Clustering/tutorial_html/kmeans.html .

In a nutshell, the K-means clustering algorithm divides a set of point into a pre-defined number of clusters and yields the centers of the k clusters.

The following results were obtained for traffic samples that were not used in the training set.

Case 1: NetBIOS Traffic
pcap: 192.168.61.25:137-192.168.61.255:137
Expected protocol: netbios-ns

Distance from various protocol cluster centers:

```
1780.30860264 = ntp
1936.35599254 = route
2764.66914234 = snmp
1832.0630088  = netbios-dgm
1818.12445314 = skype
2199.13745758 = nfs
676.334483051 = netbios-ns
```

```
3244.52297705 = bootpc
```

best guess: 676.334483051 = netbios-ns
second best guess: 1780.30860264 = ntp
distance between guesses: 1103.974119589

Case 2: Skype Traffic
pcap: pcap.skype..2126.41329
Expected protocol: skype

Distance from various protocol cluster centers:

```
1960.45561284 = ntp
2522.05029833 = route
2689.22193848 = snmp
2549.95681014 = netbios-dgm
737.228693256 = skype
1837.09071885 = nfs
1710.04898741 = netbios-ns
3296.3372724  = bootpc
```

best guess: 737.228693256 = skype
second best guess: 1710.04898741 = netbios-ns
distance between guesses: 972.820294154

Case 3: RTP Traffic with Steganography

Real-time Transfer Protocol (RTP) is used almost ubiquitously by Voice over IP technologies to provide an audio channel for calls. As such, it provides ample opportunity for creation of a covert communications channel due to its very nature and use in implementation.
We compared the entropy of a lot of SIP RTP traffic in the corporate with traffic generated by a steganography too. As expected, the Shannon Entropy for typical RTP traffic in the corporate was around 4.3 Vs 5.8 or higher (~ 35% difference) for RTP with embedded steganographic data.

As can be seen from the results for the 3 examples, the statistical nature of the protocols can be used to identify protocols in an accurate fashion or identify anomalous use of protocols for covert operations.

3.2 PISA Framework

The PISA analysis and clustering code will be posted at:
http://dvlabs.tippingpoint.com/projects/PISA

4 References

[1] Peer-to-Peer Botnets
http://www.usenix.org/events/hotbots07/tech/full_papers/grizzard/grizzard_html/

[2] Botnets Getting Beefier
http://www.eweek.com/article2/0,1895,2114741,00.asp

[3] RTP Steganography
http://www.theblackandwhiteball.co.uk/talk-realtime.php