Detecting policy violations through traffic analysis

Jeffrey Horton and Rei Safavi-Naini
Centre for Information Security
University of Wollongong
Northfields Avenue, Wollongong, Australia
{jeffh,rei}@uow.edu.au

Abstract

Restrictions are commonly placed on the permitted uses of network protocols in the interests of security. These restrictions can sometimes be difficult to enforce. As an example, a permitted protocol can be used as a carrier for another protocol not otherwise permitted. However, if the observable behaviour of the protocol exhibits differences between permitted and non-permitted uses, it is possible to detect inappropriate use.

We consider SSH, the Secure Shell protocol. This is an encrypted protocol with several uses. We attempt firstly to classify SSH sessions according to some different types of traffic for which the sessions have been used, and secondly, given a policy that permits SSH use for interactive traffic, to identify when a session appears to have been used for some other purpose.

1 Introduction

Many modern organisations provide employees with access to the Internet and other networks to enable them to perform their jobs. However, it is important to ensure that this access is being used appropriately and in compliance with organisational policy. For example, policies can be stated in terms of restrictions on Web access or the types of protocols that employees can use. However, permitted protocols can have uses which are undesirable, and it can be difficult to prevent or even detect instances of inappropriate use with current network security tools such as firewalls and intrusion detection systems. Controlling the use of encrypted network protocols is also problematic.

Modern firewalls can be very good at blocking network traffic based on low-level criteria such as source address and destination port, and can be very useful in protecting networks of machines from inappropriate or unnecessary outside access. However, firewalls that perform only this sort of simple stateful inspection cannot check that the traffic being passed conforms to the specification of the desired protocol. Firewalls providing some form of "deep inspection" [25] are capable of detecting a limited set of known attacks and protocol anomalies in certain common protocols. Proxies (such as are found in proxy firewalls) can check conformance to protocol specifications to a greater extent than deep inspection stateful packet filtering firewalls. No type of firewall provides the higher-level protocol understanding that can enable the detection of malicious usage of the protocol — a conversation which complies with the specifications, does not attempt to exploit a vulnerability such as a buffer overflow vulnerability but which does not otherwise closely resemble "normal" traffic. Tools such as `httptunnel` [4] exist to facilitate tunneling arbitrary network traffic through an HTTP proxy, for example.

Neither firewalls nor proxies handle encrypted traffic well, having no access to the data protected by encryption. Encrypted data cannot easily be scanned for malicious contents such as viruses or spyware until it reaches the end system and is decrypted. Outgoing encrypted data cannot easily be scanned for content such as proprietary corporate information being inappropriately released. It is difficult to ensure even that the underlying protocol being used is appropriate. For example, SSL may be permitted in order to allow secure access to websites, with an expectation that the protocol being protected is HTTP. However, SSL may be used in conjunction with other protocols, and it may be difficult to distinguish desired uses from unwanted and inappropriate ones.

In some cases, it is not reasonable to block all encrypted protocols — SSL is required for secure website access, and SSH (Secure Shell protocol) may be required by some users for remote access to computing resources, for example.

In this paper we will be looking at the identification of different types of uses of SSH using only the information that is available for inspection after the data has been encrypted: the packet size and inter-arrival time. We assume that the primary desirable use of SSH is the provision of secure remote terminal access. Implementations of the pro-

tocol commonly have many additional features with substantial potentially legitimate uses. However, these features can also enable users to bypass organisational security policies (for example, port forwarding features can be used to browse inappropriate Web sites during the working day, if the site terminating the SSH tunnel has a more permissive general Internet access policy than the client site) and in some environments it may be considered appropriate to attempt to detect such activity. Our intention is to be able to provide assistance in auditing compliance with relevant organisational policies. We are not just looking at tunneled traffic: we are also interested in identifying usage of features such as X11 forwarding, use of which may be discouraged due to security concerns.

We have some success classifying SSH sessions according to the type of traffic being carried in the session; for the main classes of traffic being examined we can assign the traffic to classes with low false positive and false negative rates but with significant "don't know" rates, where a connection is not assigned to an appropriate class. However, the classification process does not perform as well for some of the minor classes of traffic studied, for which we did not collect many samples. We are more successful at differentiating interactive traffic from other types of traffic, with false positive and false negative rates of around 3%. In the absence of an active attacker, we believe it is possible to identify types of activity other than simple interactive use of SSH with a high degree of confidence.

In Section 2 we look at related work, Section 3 provides a brief overview of some of the relevant capabilities of SSH implementations, in particular the OpenSSH [22] implementation of the protocol. Section 4 discusses our analysis of SSH traffic. In Section 5 we discuss experiments, data processing and results. We discuss ways in which detection could be evaded in Section 6. Finally, we conclude in Section 7.

2 Related Work

Traffic classification has emerged as an important area, now that the port over which traffic flows is no longer sufficient to determine what the actual protocol of the traffic is. This has been identified as a problem in the identification of peer-to-peer traffic by Karagiannis et al. [17], who develop a method of identifying P2P activity based on the patterns of interconnection between network peers. Non-payload based methods of traffic classification can be of particular interest, owing to privacy concerns that can arise over access to the full traffic payload.

Work in the area of traffic classification has used a variety of machine learning techniques, including decision trees [8] and hidden Markov models [31, 32]. Wright et al. [31, 32] consider a general traffic classification problem, for which SSH is one type of traffic being classified. Some interesting results are reported, but they do not look at attempting to identify different types of tunneled SSH traffic, or detection of the use of SSH features such as X11 forwarding.

Song et al. [28] have reported on an attack against SSH that uses keystroke timings as reflected in SSH packet inter-arrival times to substantially reduce the amount of work that needs to be done when cracking passwords.

We are not aware of other work that addresses the use of SSH specifically.

Tunneling has been proposed and implemented over many different protocols, including ICMP [6], Domain Name System (DNS) request and replies [16] and HTTP. An example of an HTTP tunneling package is `httptunnel` [4].

Measures to detect and filter tunneling through ICMP have been proposed [27, 26]. We are not aware of any work having been done on the detection of tunneling using DNS messages. Some work has been done on detecting tunneling using HTTP requests and responses. Pack et al. [24, 23] describe their work on detecting HTTP tunneling using behaviour profiles consisting of features such as the number of packets, the ratio of large and small packets and the total amount of data received. Unfortunately, it is difficult to say how effective this system is as there is no way presented to determine what attacks might be missed.

Mudge [19] has proposed detecting tunneling by determining what "normal" HTTP traffic looks like, and generating some simple measures, such as length of session or amount of data transferred, to use in detecting abnormal sessions. This paper does not discuss an implementation of a system to detect misuse of HTTP. Borders and Prakash [3] describe the results obtained from implementing a system to detect intruder reverse tunnels and communications back to home base of adware and spyware using a range of measures taken from "normal" HTTP traffic. Rather than focusing on packet level statistics, measures included header formatting, inter-request arrival time, request regularity and request size, among others. These are higher-level measurements than those used by Pack et al. The system was successful at detecting tunneling activity, including the activity of a tunneling program custom-designed by the authors. Bissias et al. [2] have had some success identifying the source of SSL-protected HTTP traffic using profiles of statistical characteristics of Web requests to sites of interest.

Covert channels have been extensively reported as an undesired feature of multi-level security operating system implementations that can be used to violate the security policy of these systems by enabling information to pass from a high security level to a lower one. [21, 1]. Guidelines required the minimisation of the bandwidth available through these channels [21].

Covert channels exist in modern network protocols as well [12]. For example, a method of embedding information using the TCP timestamp option has been proposed [11]. Murdoch and Lewis [20] review many of the proposals for embedding covert channels into TCP and IP, and propose a method of embedding data into TCP sequence numbers and IP identification fields that respects the statistical properties of these fields as generated by Linux and OpenBSD. Lucena et al. [18] examine IPv6 for opportunities for embedding covert channels. "Active wardens" have been proposed and to some extent implemented to clean network protocols of covert channels [9]. Firewall proxies can be helpful at removing covert channels from low-level network protocols, as these systems typically do not pass IP packets or TCP segments directly from one side of the firewall to the other.

3 SSH Capabilities

SSH is used by many as a secure replacement for remote access methods such as `telnet` and `rlogin`. Unlike these protocols, SSH provides facilities to validate the identity of the remote host in order to reduce the possibility of a man-in-the-middle attack, and encrypts and authenticates all SSH traffic during the session, protecting against network snooping attacks. In contrast, `telnet` and `rlogin` are plaintext protocols that have no security features beyond verifying a user password. However, implementations of SSH commonly have many additional features whose use is not appropriate in all environments. In this paper, we work with OpenSSH [22], principally late releases of version 3.

- **X11 forwarding**. When accessing a remote host, OpenSSH has the ability to tunnel the X11 protocol messages through the SSH connection, protecting the X11 traffic from inspection by an attacker. Unfortunately, if the administrators of the remote machine are not fully trusted, or it has been compromised, X11 forwarding may[1] allow anyone with access to the user's `.Xauthority` file to access other X windows open on the user's desktop. As an example, `xkey` [10] allows an attacker on the remote machine to snoop on key presses made in other windows on the user's desktop which are not related to the SSH session [13].

- **File transfer**, using either `scp` or `sftp`. This can be used to export corporate information, or to import arbitrary data, including malicious software which then can only be detected by antivirus software running on the user's machine.

[1] In recent versions of OpenSSH, this may require "trusted X11" forwarding to be enabled. Some X11 clients do not work properly without trusted X11 forwarding enabled, so it seems likely that trusted X11 forwarding could be expected to be enabled if X11 forwarding is enabled at all.

- **Dynamic port forwarding** causes SSH to behave as a SOCKS4 or SOCKS5 proxy. Connections to a specified local port are tunneled through the SSH connection to an arbitrary (dynamic) destination host and port. Many applications have built-in SOCKS support, which makes this technique very powerful. It works very well for performing Web browsing, which is the common use for this type of forwarding in the experiments performed in this paper.

- **Static local port forwarding** allows connections to local ports to be tunneled through the SSH connection to a designated remote host and port. It is very similar to dynamic port forwarding, except that the final destination for the forwarded traffic is fixed. Some protocols cannot be used effectively with static port forwarding.

Other relevant capabilities which we have not investigated in detail include **remote static port forwarding** and (new in version 4) support for establishing full VPNs using an SSH tunnel.

OpenSSH also provides the ability to compress data before forwarding it through the tunnel, which can be very useful for reducing the amount of network traffic required, particularly for slow networks. It does mean that it is necessary to consider both compressed and uncompressed forms of different types of traffic when attempting to build a useful classification system.

ACSI 33 [7], an information security policy guide published by Australia's Defence Signals Directorate, includes guidance on the recommended configuration of SSH for Australian government agencies. In particular, it is recommended to disable connection forwarding and X11 forwarding. However, not all servers located at other organisations with different security policies to which a user might connect will necessarily have these features disabled, and enforcing the restrictions from the client side of the connection may be impractical.

We identify SSH's interactive capabilities as its primary desirable feature. We aim to detect non-interactive use and possibly even identify use of X11 forwarding, file transfer and different types of port forwarding, so that some form of limited audit capability could be implemented for compliance with high-level policies on the use of SSH.

4 Analysis of SSH traffic

Our analysis of SSH traffic is based on collecting statistical distributions for SSH traffic in which particular types of activity are conducted, and using these distributions to classify unknown connections using an instance-based learning approach. Our hypothesis is that data such as packet sizes will reveal information about the activity that is being conducted during the SSH session. We represent an SSH

connection with a probability distribution (normalised frequency of packet sizes) and use multiple samples of a particular type of traffic (for example, file transfer) to define a class of connections.

A "connection" can be described by a 4-tuple consisting of source and destination IP and source and destination port. We do not use timeouts to separate connections with the same 4-tuple from each other, because the protocol which we are studying uses TCP for data transport. A TCP connection can be left idle without timing out, and use can then be resumed without difficulty. The packets that are used to initiate a TCP connection have the TCP SYN flag set; we use packets with the SYN flag set and corresponding to the same 4-tuple to separate connections from each other.

The size of packets in the SSH connection provides only a general indication of the amount of actual application data being carried. This is because of the cryptographic protection mechanisms: block ciphers process data in chunks of a particular size (AES, for example, has a block size of 16 bytes), and integrity protection can result in message authentication codes of 10 bytes or more being attached. Small amounts of extra padding can also be added. A single keypress in an interactive SSH session can result in an SSH packet of around 50 bytes. In some cases, compression prior to encryption will remove redundancy from the plaintext and result in smaller packets observed than would otherwise be the case.

We divide a connection into two halves, the part of the connection in which data flows from client to server, and the part in which data flows from server to client. Statistics are computed separately for each.

We evaluate how well the classification process works using *cross-validation* [30] and *confusion matrices*. Cross-validation is a standard technique for evaluating machine learning algorithms. In this case, we use 10-fold cross-validation. We implement cross-validation by randomly partitioning the data set into 10 partitions. One partition is used as test data, the other nine as training data for classifying the test samples. Each partition is used as test data in turn, and the results of the classification are used to form a confusion matrix. The rows of the confusion matrix represents the actual class of an item, and the columns the classes to which the classification algorithm assigns the item. Confusion matrices show how readily different classes of items are distinguished from each other by the classification algorithm. Small variations are possible in the results of cross-validation due to different random partitions of the data set.

4.1 Algorithm description

The classification algorithm can be described as follows:

- Load connection packet size probability distributions and other statistics for connections of known classification.
- Generate statistics for connection of unknown classification.
- Measure the "distance" between the statistics of the unknown connection and each known connection.
- Sort the scores in order of closeness.
- The unknown connection is assigned to the same class as the majority verdict (3 or more) of the closest five connections.

4.2 Frequency distributions

The maximum size of packets will vary depending on the physical network medium used to transmit the packets. Our experiments were conducted on Ethernet networks, and for this medium the largest packet size is 1500 bytes. This quantity is otherwise known as the Maximum Transfer Unit (MTU). However, we use only the amount of user data present in each TCP segment for generating the frequency distributions. We do not count the IP or TCP headers. The amount of user data that is present in packets is further constrained by the TCP Maximum Segment Size (MSS) for each part of the connection. The network MTU affects the MSS setting, but it is also affected by the presence of firewall equipment in the network, which can adjust the MSS of connections. We chose a maximum for our packet size range of 1300 bytes; larger packets would be assigned to the same bin as a packet of 1300 bytes.

We did not count packets with no user data, such as TCP acknowledgment-only packets.

We assigned each data point to one of 30 bins. Too few bins do not bring out the features of each distribution sufficiently. Too many bins will be oversensitive to minor variations in the traffic. 30 bins was considered to approximate a distribution with a bin size of 1 (the extreme) reasonably well. The bins were of integral size; leftovers were distributed evenly across the bins, so that all bins are very similar in size.

After computing the packet size frequencies, the distribution was normalised to form a probability distribution which could be used with some selected distance measures.

4.3 Distance measures

There are many distance measures which can be used to compare the closeness of two probability distributions, including the variational distance, harmonic mean and Kullback-Liebler divergence. We used the Jensen-Shannon divergence [5] (a form of the Kullback-Liebler divergence) and the Bhattacharyya distance [15].

The Bhattacharyya distance was calculated using[2]:

$$B(P,Q) = \sqrt{1 - \sum_i \sqrt{p_i q_i}} \quad (1)$$

The Jensen-Shannon divergence was calculated[3] using:

$$D_{\mathrm{JS}}(P\|Q) = \frac{1}{2}\sum_i (p_i \lg \frac{p_i}{\frac{p_i+q_i}{2}} + q_i \lg \frac{q_i}{\frac{p_i+q_i}{2}}) \quad (2)$$

In both equations P and Q represent the two probability distributions being compared. P might represent a distribution of known classification, Q a distribution of unknown classification. However, as these measures are symmetric, which is known and which is unknown is not important.

Both measures are easily calculated, the Bhattacharyya distance being somewhat simpler to compute than the Jensen-Shannon divergence. Results closer to zero indicate better matches for each measure. The Bhattacharyya distance has been used with machine learning approaches [5] and has been used with success for signal selection [15], for example. The Jensen-Shannon measure is symmetric, unlike the Kullback-Liebler divergence itself, and was chosen as a representative of the divergence-type measures that were described as a form of the Kullback-Liebler divergence.

Since we separate a single connection into client- and server-sourced parts, we will have two distance measure calculations, one for each distribution, forming a score vector. The final score for any two distributions being compared is the magnitude of the score vector. This allows additional measures to be easily used if desired.

4.4 Additional statistics

In addition to the probability distribution distance measures we can compute, there are many other possible statistics that may be used to shed some light on the type of activity occurring during an SSH session. We compute the following additional statistics:

- Bytes per second for packets with an inter-arrival time of less than 2 seconds. This statistic captures information on the *amount* of data transferred during the times when the connection is active, so periods of time when the connection is idle do not influence the generation of the statistic. The statistic is computed by totaling the packet sizes and dividing by the sum of the inter-arrival times, for packets with inter-arrival times less than 2 seconds.
- Packets per second for packets with an inter-arrival time of less than 2 seconds. This statistic is computed in the same manner as that of bytes per second.
- Bytes per packet.
- We define a "chain" as a sequence of packets of the same size at least 5 packets in length, and use this to calculate two statistics:
  - The packet size whose chains account for the largest number of packets in the connection.
  - The packet size that accounts for the largest number of chains.

  The idea behind these statistics is that an interactive connection will consist of a large number of packets of a very similar size.

Typically when using these statistics for classification the logarithm of the value rather than the value itself will be used. Bytes per second, for example, exhibits a wide range of values. Using a logarithm of the value reduces the range of variation and minimizes the effect that a statistic has on the other statistics being used in the classification so that the effect is not overwhelming. In the case of bytes per second we also impose a threshold on the value of four kilobytes per second — anything larger is treated as 4kB/second. This is sufficient for the main types of behaviour we seek to identify.

5 Experiments

5.1 Data Collection

The data collected constitutes full header and partial payload information on a subset of the principal author's SSH sessions from late 2005 to May, 2006. This constitutes over 400 sessions and 11.5 million packets. All except some file transfer sessions (which appears not to be truly interactive, not even when using `sftp`) were operated manually. Network traffic was captured using `tcpdump`, running on either the source or sink of the traffic. Care was taken that packets were not double-counted. Capturing traffic at a point intermediate to the SSH client and server would have been preferred, but this option was not available for these experiments.

Most of the traffic captured comprised interactive, X11, file transfer and Web browsing sessions tunneled through an SSH connection. Small amounts of USENET news traffic (NNTP) and Samba SMB/CIFS file sharing traffic were captured[4]. Finally, a number of graphical sessions using

[2] An alternative form could be $B(P,Q) = -\ln \sum_i \sqrt{p_i q_i}$.

[3] Note that we used logarithms to the base 2 in calculating the Jensen-Shannon divergence.

[4] Tunneling Samba traffic through SSH required the aid of a transparent SOCKS proxy redirector and local firewall support to redirect the traffic to the proxy.

NoMachine's NX X11 compression technology were captured between a NoMachine NX client and a FreeNX server. We provide further information on the main categories of monitored sessions:

- Interactive sessions consisted of a variety of activities, including mail reading and editing, programming, system administration tasks and other command shell work. In many sessions extensive use was made of `screen`, a terminal multiplexer that allows many command line shells to be executed simultaneously within a single login session. The size of the window will affect the size of response packets from the server where screen refreshes are required, so windows of a variety of sizes were used.

- X11 applications used in these sessions included `xterm`, `synaptic`, Acrobat Reader, OpenOffice, `xdvi` and ImageMagick.

- Web browsing included access to daily news sites, corporate sites, sites with programming and development information and a variety of other sites of personal and professional interest to the principal author. Only small files such as PDFs or certain source code packages were downloaded during these monitored Web browsing sessions. Web browsing used Mozilla, Safari and Firefox.

- File transfer was principally composed of small files (on the order of several megabytes in size) or groups of such files. File transfers were performed in both directions; that is, files were transferred both to and from the file transfer client.

It should be noted that as far as possible, sessions were "pure": a session used for HTTP tunneling would be used only for HTTP tunneling, not interactive traffic as well (it would be possible in practice to do both, however).

NX was of interest because it is claimed that this technology can make X11 sessions useful even over network links with very restricted bandwidth capabilities, and it was thought that for some types of activities such a session might more closely resemble an interactive session than other types of graphical session. In Section 3, we identified security concerns with the use of X11 forwarding through SSH. In some brief experiments with NX X11 sessions, we were unsuccessful in using the X11 xkey exploit previously discussed to snoop on other X windows unrelated to the NX session; it only appeared to allow snooping on other X windows open in the NX session, a much less significant problem than for ordinary X11 forwarding but still possibly of concern.

A variety of machines were involved, all older than 3 years running some version of Linux or Mac OS X. OpenSSH clients and servers were all using protocol 2. No special effort was made to tune the clients and servers, with the exception that one client was adjusted to prefer a MAC (Message Authentication Code, used for data integrity protection) based on SHA1 instead of MD5, and in order to prevent firewall timeouts disconnecting idle sessions keepalive messages were enabled to some destinations.

Sessions were established between computers over several different types of network connection:

- university computers and researcher's home (ADSL 512/128);

- university computers and computer of ISP offering SSH access;

- two university computers, mostly on the same network subnet;

- two home computers, on the same subnet.

Where possible, traffic was generated with and without SSH session compression enabled.

The SSH traffic involved with X11 forwarding, NoMachine NX and Samba was restricted to local network activity — we did not have cause to employ these protocols over a wide-area network such as the Internet.

One challenge in work of this nature is identifying the nature of the training sample. Is this an interactive session? Or does the data represent a tunneled Web browsing session? In some areas, such as the identification of P2P connections [17], it is possible to validate a non-payload based protocol identification technique by using an alternative payload-based method. However, this is not an option in this case.

So a wrapper script, written in Python, was installed on those machines used to initiate SSH connections. This wrapper script would log the command and start time of an SSH session in an SQLite [14] database. On command exit, an opportunity would be presented to the user to add a short description of the completed session, which would also be logged to the database. This information could be used later in classifying a session into one of several different classes.

5.2 Data Processing

Useful information such as the source and destination addresses and ports, TCP flags and sequence numbers was extracted from the headers of captured packets and logged to an SQLite database. Connections were identified by searching for the SYN-flagged packets that identified the start of each TCP session, and information on the session, including its start and end time, and a unique 48-bit identifier based on a hash of the connection start time, source and destination addresses and ports, would be entered into

F	File transfer
H	Tunneled HTTP
I	Interactive
IX	X11 forwarding
N	Tunneled NNTP
Q	NX graphical session
S	Tunneled Samba

Table 1. Traffic class labels

the database. Connections were manually classified into an appropriate class using stored information on the command executed and user-supplied comments. Statistical information was calculated for every classified connection. All this information was stored in the SQLite database for later use.

When generating the statistics for each connection, we use only packets that have arrived in sequence. The sequence can be disturbed by packets reordered in the network, packets retransmitted (and possibly combined with other packets, rather than the original packet alone being retransmitted) or packets dropped by the traffic capture program. We implement some simple code to follow the TCP sequence numbers for each connection.

5.3 Results

We use 10-fold cross-validation to generate confusion matrices to evaluate the performance of the classification. In the figures and tables that follow, the different types of traffic are labeled as described in Table 1.

Figures 1, 2, 3 show sample cumulative probability distributions for traffic through an SSH tunnel representing interactive, Web browsing and compressed Web browsing traffic respectively. In each figure, the graph on the left represents traffic sent by the client to the server, and the graph on the right represents traffic sent by the server to the client. Each line represents a cumulative probability (frequency) distribution of sizes of packets from a single connection, split into a client half and a server half. At any point, the graphs reflect the proportion of packets from the connection that were of a particular size or smaller.

As can be seen, these distributions are quite distinct. However, the same could not be said for all of the distributions.

Tables 2 and 3 show confusion matrices using just the Bhattacharyya and Jensen-Shannon measures to classify connections into a particular class. 'XX' denotes the unknown class, indicating that no majority verdict could be reached. Recall that rows are known classes, columns indicate predicted class and the final column indicates the number of instances available for classification. Matrix entries are % of total number of connections of each type.

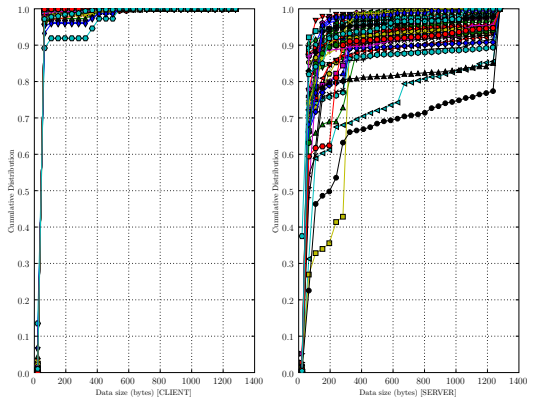

Figure 1. Cumulative probability distribution for class 'I', interactive SSH traffic.

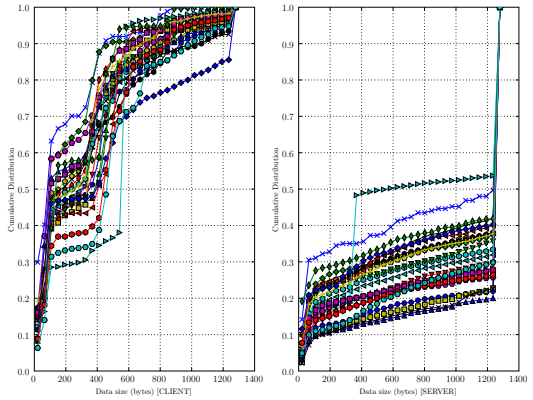

Figure 2. Cumulative probability distribution for class 'H', Web browsing forwarded through an SSH tunnel.

	F	H	I	IX	N	Q	S	XX	Count
F	93.86	0.00	0.88	0.00	0.00	0.00	0.00	5.26	114
H	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00	49
I	0.00	0.00	96.48	0.88	0.00	0.88	0.00	1.76	227
IX	0.00	1.52	6.06	90.91	0.00	0.00	0.00	1.52	66
N	12.50	0.00	0.00	0.00	37.50	0.00	0.00	50.00	8
Q	0.00	0.00	44.44	11.11	0.00	44.44	0.00	0.00	9
S	14.29	0.00	28.57	28.57	7.14	0.00	14.29	7.14	14

Table 2. Confusion Matrix: Bhattacharyya

	F	H	I	IX	N	Q	S	XX	Count
F	93.86	0.88	0.00	0.00	1.75	0.00	0.00	3.51	114
H	2.04	97.96	0.00	0.00	0.00	0.00	0.00	0.00	49
I	0.00	0.00	96.48	1.32	0.00	1.32	0.00	0.88	227
IX	1.52	1.52	6.06	87.88	0.00	0.00	0.00	3.03	66
N	37.50	12.50	0.00	0.00	12.50	0.00	0.00	37.50	8
Q	0.00	0.00	55.56	11.11	0.00	33.33	0.00	0.00	9
S	14.29	0.00	28.57	28.57	7.14	0.00	14.29	7.14	14

Table 3. Confusion Matrix: Jensen-Shannon

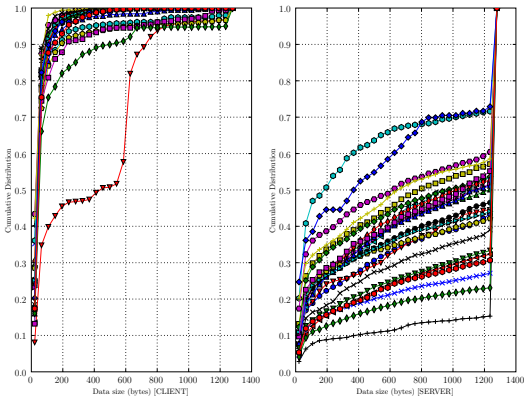

Figure 3. Cumulative probability distribution for class 'H' compressed, Web browsing forwarded through an SSH tunnel where SSH is performing compression prior to encrypting the tunneled data.

	F	H	I	IX	N	Q	S	XX	Count
F	74.56	0.00	0.00	0.00	0.00	0.00	0.00	25.44	114
H	0.00	85.71	0.00	0.00	0.00	0.00	0.00	14.29	49
I	0.00	0.00	89.43	0.00	0.00	0.44	0.00	10.13	227
IX	0.00	0.00	0.00	74.24	0.00	0.00	0.00	25.76	66
N	0.00	0.00	0.00	0.00	12.50	0.00	0.00	87.50	8
Q	0.00	0.00	22.22	0.00	0.00	44.44	0.00	33.33	9
S	0.00	0.00	0.00	0.00	0.00	0.00	14.29	85.71	14

Table 4. Confusion Matrix: Bhattacharyya + avg. bytes/sec + threshold=0.09

As can be seen, both measures do a reasonable job of separating the four classes with the largest number of connections, but neither does particularly well for the three classes with only a few connections. We can improve the class separation by introducing the bytes per second measure previously described, to add an indication of how much data was transferred during each session, and a simple threshold on the final composite score, to capture the idea of another distribution needing to be "close enough" before being considered to be a good match. This is likely to be especially important when attempting to classify connections involving a protocol for which no training samples had previously been collected. Table 4 shows the results of the classification for this combination, using a threshold value of 0.09 (threshold value determined by experiment).

Quite a few connections are now classified as "unknown", but class separation is otherwise much improved. This could be useful in detecting certain types of non-permitted uses of SSH. For example, 85% of tunneled HTTP connections were successfully classified, with no false positives. So where this type of SSH usage is not permitted by policy, a reasonable level of confidence could be had in this identification. Class separation when using the Jensen-Shannon method can be improved as well, but not as substantially as is the case when using the Bhattacharyya method, as Table 5 indicates.

	F	H	I	IX	N	Q	S	XX	Count
F	79.82	0.88	0.00	0.00	0.00	0.00	0.00	19.30	114
H	0.00	87.76	0.00	0.00	0.00	0.00	0.00	12.24	49
I	0.00	0.00	90.75	0.00	0.00	0.44	0.00	8.81	227
IX	1.52	0.00	1.52	72.73	0.00	0.00	0.00	24.24	66
N	0.00	12.50	0.00	0.00	50.00	0.00	0.00	37.50	8
Q	0.00	0.00	22.22	0.00	0.00	44.44	0.00	33.33	9
S	7.14	0.00	0.00	7.14	0.00	0.00	35.71	50.00	14

Table 5. Confusion Matrix: Jensen-Shannon + avg. bytes/sec + threshold=0.015

	I	XX	Count
I	96.48	3.52	227
XX	5.00	95.00	260

Table 6. Confusion Matrix: Bhattacharyya

We now consider how compliance with a policy that specifies that SSH is to be used in an interactive manner only might be audited. In this case, interest is not so much in what particular sort of activity is being conducted, but in whether the session represents interactive SSH or something else. Table 6 displays the results of using only the Bhattacharyya measure to classify connections into classes "interactive" and "everything else". Classifications have a reasonably low level of false positives and false negatives, which can be improved by including more of the statistical measures previously described in the classification process. Table 7 displays the results of this classification process. It can be seen that both the false positive and false negative rates have been improved.

6 Evasion

The distribution of packet sizes from an SSH connection is an important part of the classification performed here. It would be possible to affect the results by altering this distri-

	I	XX	Count
I	97.80	2.20	227
XX	1.92	98.08	260

Table 7. Confusion Matrix: Bhattacharyya + maxpacketsize + modepacketsize + avg. bytes/sec + threshold=0.9

bution of packet sizes. Simple methods by which this could be attempted include reducing the MTU of the network interface through which outgoing SSH traffic is sent and compiling a modified SSH binary which adjusts the TCP maximum segment size using the TCP_MAXSEG socket option [29, p. 219]. Changing the MTU reduces the maximum packet size that can be sent over the interface, not just of SSH traffic but for all traffic and also affects the size of packets sent in both directions on the connection. The TCP_MAXSEG option affects only the particular connection to which it is applied.

Reducing the size of the largest chunk of data that can be sent in a single TCP segment will change the packet size distribution for uses of SSH in which there are many larger packets (such as file transfer and HTTP(S) forwarding).

Significantly changing the size of the largest packet that can be sent or received will certainly affect the ability to classify traffic as representing a particular type of usage. However, the classification of traffic as representing an interactive SSH session or some other type of SSH session would be less affected: Figure 1 illustrates that both halves of an interactive SSH connections are principally composed of small packets. Sessions that have a higher proportion of larger packets, such as HTTP(S) and file transfer sessions, would not easily be confused with interactive SSH, even when limiting the maximum packet size (by changing the MTU, for example). The use of additional statistical measures may need to be applied. For example, greater intervention would be required to spread the data transferred out over a longer period of time in order to preserve the bytes per second, packets per second and bytes per packet. For the client to server part of the connection, 92% of interactive sessions in the test data collected satisfy conditions of average bytes per second less than 300, average packets per second less than 6 and average bytes per packet less than 70. Only 8% of forwarded Web browsing sessions satisfy the same requirements. However, a sufficiently motivated and patient adversary with the necessary control over both the SSH client and server would be able to evade detection.

An adversary that combines several different types of usage in the one SSH session would also be able to frustrate classification of the session as one particular type of traffic, but without any guarantees of what sort of traffic the session would be classified as instead.

Practical deployment of systems that detect anomalies in the operation of some system or the use of some protocol can be challenging. It is difficult to collect training data which is "clean", that is, which does not contain any attacks. This problem would be familiar to researchers working with some types of intrusion detection systems. In our case, some generic behaviour profiles could be provided. If local customisation is required to improve detection performance or provide detection for obscure uses, trusted staff (perhaps including IT staff and others from the wider user population) who had occasion to work with SSH could be enlisted to help create clean data that could be used to detect anomalies.

7 Conclusion

Tunneling one network protocol through another is one way that attackers or malicious insiders can communicate through firewalls or other traffic control and monitoring devices. Furthermore, some network protocols such as SSH have multiple uses, some of which may be desirable, others which are prohibited by security policies. By monitoring network traffic, it may be possible to determine what types of activity is taking place, and so provide a simple audit capability for compliance with security policies. We describe work to separate SSH connections into different classes, using some simple statistical calculations and comparisons. We found that using these methods, it is possible to identify different types of SSH activity with a reasonable degree of confidence (greater than 74% for the main four types of activity examined, with very low false positives). Furthermore, without identifying what *kind* of traffic it is, it is possible to identify types of activity other than simple interactive use of SSH with a high degree of confidence.

We found that it was possible to obtain a reasonable classification without attempting to make much use of interarrival times. While we did collect this information, the manner in which we captured the traffic did not ideally reflect the environment in which such captures would be conducted in practice, which would ordinarily be on a gateway machine at some intermediate point between the traffic source and sink. It was considered that this did not adversely affect the packet size data, however.

All sessions were based on the activities of a single user. We have endeavoured to ensure that a reasonable range of activities were performed during the monitored sessions so that there could be some expectation that the results would be reflective of the activities of a wider user population. In addition to enrolling additional users in a subsequent study, further work could include use of more sophisticated machine learning methods to improve the classification, use of more statistical measures in classification, extension to other types of tunneled protocols and attempting classification based on a sliding window of packets within a session, to identify different uses, some perhaps not conforming with policy, within the one SSH session.

References

[1] R. J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, 2001.

[2] G. D. Bissias, M. Liberatore, D. Jensen, and B. N. Levine. Privacy vulnerabilities in encrypted HTTP streams. In *Privacy Enhancing Technologies Workshop*, May–June 2005.

[3] K. Borders and A. Prakash. Web tap: detecting covert web traffic. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 110–120, Washington, DC, 2004.

[4] L. Brinkhoff. GNU httptunnel. `http://www.nocrew.org/software/httptunnel.html`. Last checked: 20060529.

[5] S. Chapman. String similarity metrics for information integration. `http://www.dcs.shef.ac.uk/~sam/stringmetrics.html`. Last checked: 20060529.

[6] daemon9. Project Loki: ICMP tunneling. `http://www.phrack.org/show.php?p=49\&a=6`, November 1996.

[7] Department of Defence. *Australian Government Information and Communications Technology Security Manual (ACSI 33)*, March 2006. Available from `http://www.dsd.gov.au/library/infosec/acsi33.html`.

[8] J. P. Early, C. E. Brodley, and C. Rosenberg. Behavioral authentication of server flows. In *ACSAC '03: Proceedings of the 19th Annual Computer Security Applications Conference*, December 2003.

[9] G. Fisk, M. Fisk, C. Papadopoulos, and J. Neil. Eliminating steganography in internet traffic with active wardens. In *Information Hiding 2002*, volume 2578 of *Lecture Notes in Computer Science*, pages 18–35. Springer-Verlag, October 2002.

[10] D. Giampaolo. xkey.c. `http://www.phreak.org/archives/exploits/unix/xwin-exploits/x11serv.c`. Last checked: 20060531.

[11] J. Giffin, R. Greenstadt, P. Litwack, and R. Tibbetts. Covert messaging through TCP timestamps. In *Privacy Enhancing Technologies*, volume 2482 of *Lecture Notes in Computer Science*, pages 194–208. Springer-Verlag, April 2002.

[12] T. G. Handel and I. Maxwell T. Sandford. Hiding data in the OSI network model. In *Proceedings of the First International Workshop on Information Hiding*, volume 1174 of *Lecture Notes in Computer Science*, pages 23–38. Springer-Verlag, 1996.

[13] B. Hatch. SSH users beware: The hazards of X11 forwarding. `http://www.hackinglinuxexposed.com/articles/20040705.html`, 2004. Last checked: 20060531.

[14] D. R. Hipp. SQLite: An embeddable SQL database engine. `http://www.sqlite.org/`. Last checked: 20060529.

[15] T. Kailath. The divergence and Bhattacharyya distance measures in signal selection. *IEEE Transactions on Communications*, 15(1):52–60, February 1967.

[16] D. Kaminsky. OzymanDNS. `http://www.doxpara.com/ozymandns_src_0.1.tgz`. Last checked: 20060531.

[17] T. Karagiannis, A. Broido, M. Faloutsos, and kc claffy. Transport layer identification of P2P traffic. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement (IMC'04)*, pages 121–134, Taormina, Sicily, Italy, October 2004.

[18] N. B. Lucena, G. Lewandowski, and S. J. Chapin. Covert channels in IPv6. In *Privacy Enhancing Technologies Workshop*, May 2005. (Not yet published.).

[19] Mudge. Insider threat: Models and solutions. *;login:*, 28(6):29–33, December 2003.

[20] S. J. Murdoch and S. Lewis. Embedding covert channels into TCP/IP. In *Information Hiding: 7th International Workshop*, volume 3727 of *Lecture Notes in Computer Science*, pages 247–261. Springer Verlag, June 2005.

[21] National Computer Security Center. Department of Defense Trusted Computer System Evaluation Criteria. DoD 5200.28-STD, December 1985. ("Orange Book").

[22] OpenBSD Project. OpenSSH. `http://www.openssh.org`. Last checked: 20060531.

[23] D. J. Pack and B. E. Mullins. A portable microcontroller-based HTTP tunnelling activity detection system. In *Proceedings of the 2003 IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 1544–1549, October 2003.

[24] D. J. Pack, W. Streilein, S. Webster, and R. Cunningham. Detecting HTTP tunneling activities. In *Proceedings Of The 2002 IEEE Workshop On Information Assurance*, 2002.

[25] M. J. Ranum. What is "deep inspection"? `http://www.ranum.com/security/computer_security/editorials/deepinspect/%index.html`, 2005. Last visited: 2006-08-25.

[26] A. Singh, O. Nordström, C. Lu, and A. L. M. dos Santos. Malicious ICMP tunneling: Defense against the vulnerability. In *Proceedings of ACISP '03: the 8th Australasian Conference on Information Security and Privacy*, volume 2727 of *Lecture Notes in Computer Science*, pages 226–236. Springer-Verlag, 2003.

[27] T. Sohn, J. Moon, S. Lee, D. H. Lee, and J. Lim. Covert channel detection in the ICMP payload using Support Vector Machine. In *Computer and Information Sciences - ISCIS 2003*, volume 2869 of *Lecture Notes in Computer Science*, pages 828–835. Springer-Verlag, 2003.

[28] D. X. Song, D. Wagner, and X. Tian. Timing analysis of keystrokes and timing attacks on SSH. In *Proceedings of the 10th USENIX Security Symposium*, Washington, D.C., USA, August 2001. USENIX Association.

[29] W. R. Stevens, B. Fenner, and A. M. Rudoff. *UNIX Network Programming: The Sockets Networking API*, volume 1. Addison-Wesley, 3rd edition, 2004.

[30] I. H. Witten and E. Frank. *Data mining: practical machine learning tools and techniques*. Morgan Kaufman, 2nd edition, 2005.

[31] C. Wright, F. Monrose, and G. M. Masson. HMM profiles for network traffic classification. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 9–15, October 2004.

[32] C. V. Wright, F. Monrose, and G. M. Masson. Towards better protocol identification using profile HMMs. Technical Report JHU-SPAR051201, Johns Hopkins University, 2005.