Detecting HTTP Tunneling Activities*

Daniel J. Pack$,*Member*, IEEE , William Streilein+, Seth Webster+, and Robert Cunningham+

Abstract**—*In this paper we present a novel intrusion detection system which makes use of behavior profiles to identify HyperText Transfer Protocol (HTTP) tunneling activities. Behavior profiles correspond to inherent attributes of application network sessions. Our system evaluates network behaviors at two different levels: a local multi-packet level and a session level. When suspicious behavior is detected, a verification module performs a detailed analysis of the corresponding session data. Currently, our system detects both malicious and unauthorized HTTP tunneling activities. Our experimental results show the effectiveness of our system and demonstrate the validity of using packet features for anomaly detection.*

Index Terms**—HTTP Tunneling, Tunneling Detection, Behavior Profiles, Network Intrusion Detection**

I. Introduction

RECENTLY, HTTP tunneling activities[1] have received an increased amount of attention from the Intrusion Detection community. The primary reason for this is the extensive use of HTTP in the Internet traffic and thus the widespread potential for misuse of HTTP for tunneling data and control. current Intrusion Detection Systems (IDSs), however, do not adequately detect HTTP tunneling activities. This lack of protection against misuses of the HTTP is troublesome, given its pervasiveness and the fact that it is allowed to flow freely through most firewalls. For instance, we monitored the Internet traffic of a large enterprise for a one week period and found that over 40% of all incoming and over 90% of all outgoing data consisted of HTTP traffic.

In this paper, we demonstrate the utility of packet features to detect, identify, and verify abnormal HTTP web traffic. When we use the term *packet features*, we mean attributes extracted from a single packet or a set of packets such as packet size, change of packet size, or the time between the first and last packets in a connection. One advantage of using packet features for the detection of HTTP tunneling is that more effort is required on the part of the attacker to manipulate these aspects of the traffic, in order to hide unauthorized HTTP tunneling activities and attacks. Another important advantage of using packet features is that these features can easily be extracted from packet headers without having to parse large volumes of data associated with web traffic. Lightweight traffic parsing such as this makes the proposed techniques appealing for real-time applications. The extensive use of packet features as described in our paper has not been found in existing IDSs except in the work presented by Paxon and Zhang [1] where they used the frequency of small packets to detect the presence of interactive backdoors.

II. Background

To some extent most firewall and IDS developers are aware of the potential for illegal HTTP tunneling. In fact, simple signature matching techniques for detecting them have already been incorporated into some existing firewalls and IDSs. However, these techniques have the following shortcomings.

For intrusion detections software running on dedicated machines, parsing all the packet data and then searching for attack signatures is computationally expensive and thus, not feasible for real-time applications. Without parsing the data, however, simple string matching tends to produce high false alarm rates [2]. Furthermore, signature based systems can not generalize attack patterns and fail to recognize new types of tunneling activities [3]. A similar problem exists with the firewalls. Stateless firewalls allow all HTTP traffic to pass through them as long as allowed port numbers and IP addresses are appropriately specified in access control lists. Stateful application proxies provide a bit more protection by performing limited protocol verification and sometimes removing Java and Javascript from the data stream. These proxies cannot, however, perform detailed analysis of each packet and still keep up with the high data rates associated with web traffic. Unfortunately, studying limited HTTP header information does not detect tunneling activities; simple signature matching techniques cause high false alarm rates, making the techniques impractical; and parsing all packet contents is computationally too expensive. To further complicate this problem, HTTP tunneling techniques are currently used in many legitimate network activities such as streaming video and audio [4], management of networks using remote procedure calls encapsulated in CMIP and SNMP [5], and for passing intrusion detection alerts [6]. For these common activities, the HTTP is chosen because its traffic flows freely through most firewalls.

These same HTTP tunneling techniques, however, also provide opportunities to misuse an organization's computer network resources. By encapsulating their activities within HTTP traffic, attackers are able to interact with and control machines that would otherwise be iso-

*This work was funded by AFIWC. Opinions, interpretations, conclusions, and recommendations are those of the authors and do not necessarily represent the views of the agency or the US Air Force.

$Department of Electrical Engineering, U.S. Air Force Academy, CO 80840, USA

+MIT Lincoln Laboratory, 244 Wood Street, Lexington, MA 02420-9185

[1] The HTTP tunneling is a method to establish a bi-directional connection between two computers by encapsulating messages or attacks with the HTTP protocol for the purpose of tunneling through firewalls.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **2002**	2. REPORT TYPE	3. DATES COVERED **00-00-2002 to 00-00-2002**
4. TITLE AND SUBTITLE **Detecting HTTP Tunneling Activities**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Massachusetts Institute of Technology,Lincoln Laboratory,244 Wood Street,Lexington,MA,02420**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR’S ACRONYM(S)
		11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
U.S. Government or Federal Rights License

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**8**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

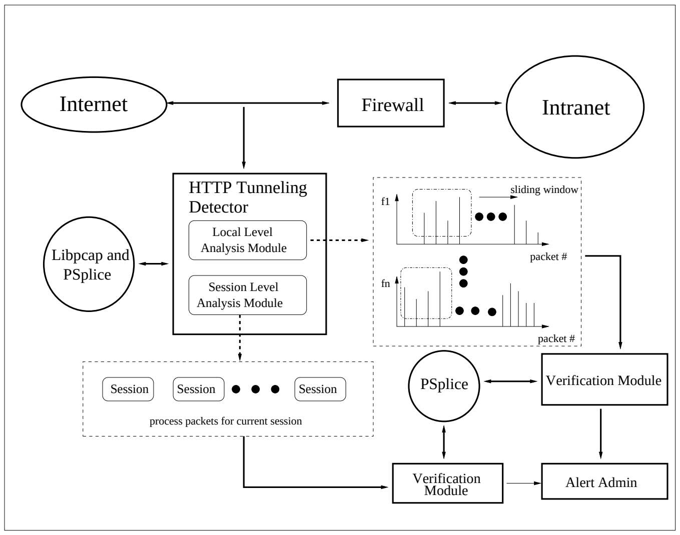


Fig. 1. System Architecture

lated from them behind a firewall. Legitimate users may use HTTP tunneling without being blocked by a firewall or detected by an IDS to access unauthorized software or to access services from the internet in violation of their organization's policies or services from the Internet without being blocked by a firewall or detected by an IDS. Occasionally, these individuals may inadvertently bring in or download malicious programs to an organization's computer network, but in most cases, such activities simply waste available network bandwidth and lower productivity. A prominent example of such activity is to stream non-work related video segments during periods of high network usage.

Given the aforementioned status of current IDSs and firewalls regarding the HTTP tunneling, the objectives of our work are (1) to introduce and demonstrate a novel approach of network intrusion detection by characterizing the inherent features associated with packet level network traffic for HTTP tunneling activities; (2) to illustrate the use of behavior profiles to capture network events; and (3) to present a hierarchical system that evaluates network behaviors at different resolutions, detecting and classifying illegitimate and unauthorized HTTP tunneling activities.

III. System Description

The system, shown in Figure 1, contains two different packet level processing modules and one additional transcript analysis module to detect, identify, and verify interactive, scripted, and streaming sessions: the local level analysis module searches for local features of individual session packets, the session level analysis module examines the average activities for an entire session up to the current time, and the verification module extracts and considers the contents of packets to verify the existence of HTTP tunneling activities.

Libpcap: The Libpcap library routines provide an interface through which applications can obtain raw packets from the network. The HTTP tunneling detection program uses its algorithms along with the PSplice routines to process each packet, which we describe later in this section. The Libpcap routines, therefore, remove the details of communications with the network interface card, allowing the tunneling detection program to focus on a high level packet feature analysis.

Psplice: Psplice [7] is a library of C++ Classes and methods that handles all the complexities of associating packets into connections and keeping track of each connection's status. It is composed of two separate components: The packet parsing component of Psplice parses individual packets and provides ready access to packet internals through the creation of packet objects. The connection tracking component of Psplice matches incoming packets to their respective connections and tracks the state of all connections seen. The connection tracking component is also able to piece the data from a connection's packets together into a seamless transcript. The HTTP detection system uses the Psplice library routines to keep statistics on and extracted information from network connection objects.

Local Level Analysis Module: The local level analysis module applies a sliding window to a set of packets to search for abnormal local activities that match any of the anomalous behavior profiles pre-generated using training data. The sliding window size is defined by the user, and it defines the number of packets considered at a time in the local level analysis module. The features for abnormal local behaviors include actual and differential packet sizes between consecutive packets and the direction of data flow.

Session Level Analysis Module: While the local level analysis module searches for distinctive local behaviors, the session level analysis module evaluates global connection behaviors by examining the running average of features and changing feature patterns. The features used to detect global anomalous activities include average size of packets, direction of data flow, ratio of large to small packets, total amount of data transfer, and total number of packets exchanged.

Verification Module: The verification module is only invoked when a suspicious activity is detected by the local analysis module or the session analysis module. Currently, the verification process consists of parsing the transcript of a connection and searching for keywords. If particular keywords are found, the verification module informs a network administrator. To verify an attack activity, we use a set of 29 keywords while 11 keywords are used to identify a stream session. We plan to interface the attack verification process with the bottleneck verification algorithm [8], a transcript analysis tool.

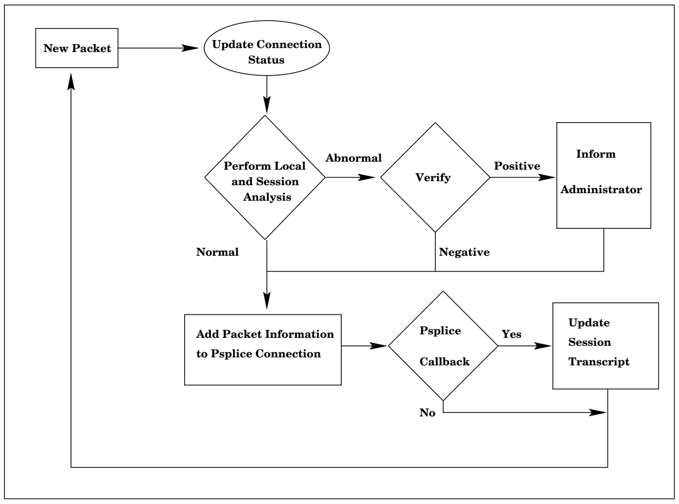


Fig. 2. Flow-Chart depicting the process involved in detecting HTTP tunneling activities.

Processing a Packet: Figure 2 shows the flow-chart of the processing path a packet goes through within the detection system. When a new packet is collected by the tunnel detection system, the detection system updates a connection object. If a packet does not belong to any existing sessions, a new connection object is created. A connection object contains information pertaining to the source and the destination of the connection, such as IP addresses and port numbers as well as the number of packets received by the client and the server, packet size, arrival time, and the contents of each packet. Each connection object also contains information such as the running packet size average, the total number of bytes received so far, and updated transcripts for both source and destination sites.

Once an appropriate connection object has been established, the new packet features along with features derived from the past n packets are analyzed by the local level analysis module. At the same time, the session level analysis module examines the set of packets seen to this point in the connection.

The verification module parses through the transcript and searches for a set of preselected keywords. If the verification module finds a tunneling activity, the appropriate information is sent to the system administrator. Otherwise, the packet is sent to the next processing module. The next processing module, shown as “Add Packet Information to Psplice Connection” block in the figure, appends the current packet information to an existing connection object if it exists or starts a new connection object The corresponding session transcript is updated only if necessary conditions are met by the Psplice callback routine[2].

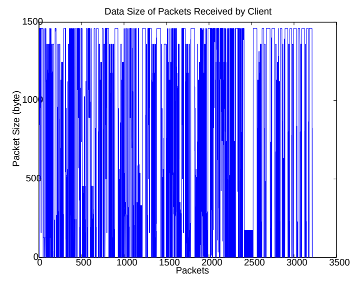
[2] We can’t simply add the packet information to a connection every time a packet arrives, since packets arrive out of order.

IV. Behavior Profiles

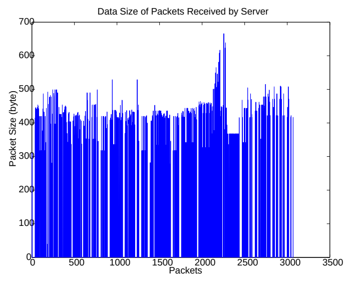
Two important advantages exist for using behavior profiles in an IDS as the basis for detecting network intrusions: generality and extendibility. One of most desirable attributes of an IDS is the ability to generalize attack patterns instead of matching specific signatures or fingerprints. In place of specific keyword string patterns, our system searches for general packet level behavior patterns. For example, our system contains one behavior profile to detect interactive sessions, another one to detect scripted attacks, and third one to detect all streaming sessions. That is, we do not have a list of different profiles to detect streaming video, streaming voice, and streaming music: one general behavior profile covers all three activities. Furthermore, this single behavior profile is used to detect streaming activities generated by four different streaming software tools and protocols: MediaPlayer, RealPlayer, QuickTime, and WinAmp. Experiments with operational data show that the same stream behavior profile allows the system to find streaming data produced by other tools, such as the NSPlayer.

The second advantage of using behavior profiles is the ability to extend the scope of detection to other types of clandestine activities. Our system can be extended to detect new abnormal behaviors by creating new behavior profiles. We show how we generate such profiles using packet level features shortly. Each behavior profile consists of a set of programmable conditions. A profile developer describes the local behavior and global behavior of an abnormal activity using individual features such as the average packet size, total number of packets, rate of packet size changes, and relational attributes such as the packet ratio between large and small packets and the disparity between the number of packets originated from a destination and the number of packets initiated from a source. The ability to define new behavior profiles provides the flexibility for administrators to easily add the detection capability for new attacks once its general packet behavior is identified.

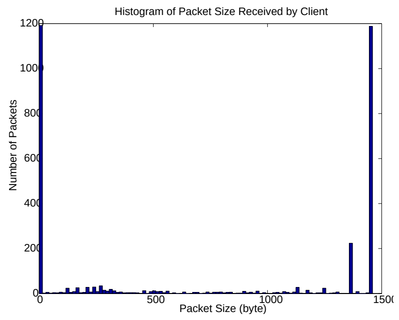
We now describe the training data used to generate behavior profiles. For normal web traffic, we collected over 100 Mbytes actual web traffic data. The data is supplemented with over 2 Gbytes of simulated web traffic data using the LARIAT [9] testbed. The data generated by LARIAT accurately portrays actual web traffic and provide much needed volume and variety of normal web traffic for training. For interactive attack sessions, seven different attack scenarios using the GNU http_tunneling program [10] were carried out while capturing the associated network traffic. The seven scenarios include probing directory structure, copying, modifying, inserting, and moving files, and carrying out FTP sessions. To generate traffic for scripted attacks, we captured the network traffic from six attack scenarios generated using an HTTP tunneling program developed at MIT-Lincoln Laboratory. These six scenarios include the following activities: probing directory



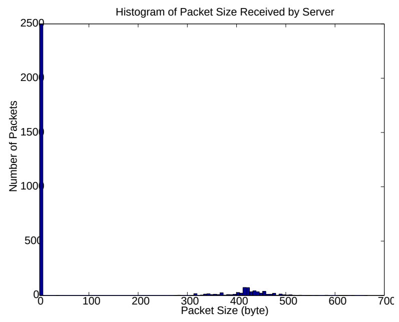
(a)



(b)



(c)



(d)

Fig. 3. A sample of authors' normal web traffic: frames (a) and (b) show the packet size received by the client and the server, respectively; the histograms for the data in frames (a) and (b) are shown in frames (c) and (d), respectively.

structure, copying and moving files, removing processes, and changing file access modes. Finally, for stream sessions, we collected network traffic while using four different streaming software clients: Realplayer, Mediaplayer, Quicktime (Window version), and WinAmp. Using each client we collected data while streaming music, audio, and video.

Creating Behavior Profiles: Currently, a behavior profile is generated by specifying the following seven packet features: (1) packet size, (2) number of packets (duration, sliding window size), (3) ratio between large and small packets, (4) data directions, (5) average packet size, (6) change of packet size pattern, and (7) size of total packets received. From our experiments, we found that these features are sufficient to distinguish abnormal HTTP session from a normal one.

As an example, we first consider the case of normal web traffic, shown in Figure 3. This data was produced by capturing network traffic while the authors surfed the Internet. A total of 3217 packets were received by a client with average of 9.218 packets per session, while a total of 3071 packets were received by a server with average of 8.799 packets per session. A total of 349 sessions are included in the data shown in the figure. Average packet sizes, shown in Figure 3 frames (a) and (b), are 745.964 bytes and 78.993 bytes, respectively. The histogram in Figure 3 frame (c) shows the distribution of packet sizes for the normal web traffic shown in Figure 3 at the client side, and frame (d) shows the corresponding distribution at the server side. Note that distribution peaks present at packet data size 0 and 1460 exist, and there is a wide spread of packet data sizes shown in frame (c). From frame (d) we observe that the data size of packets from the client to the server is relatively small with the maximum packet data size under 700 bytes. These packet data sizes reflect the average amount of data exchanged by clients and servers [11]. The example shows that, on average, the client receives an order of magnitude more data than the server. These characteristic makes intuitive sense when we consider that most web sessions consist of small data request packets from clients and large data reply packets from servers.

To demonstrate how anomalous behavior profiles are created, we compare normal web traffic data with data collected from interactive attack sessions and stream sessions. Figures 4 and 5 correspond to an interactive attack session and a stream music session using the Microsoft Mediaplyer, respectively. For each figure, frame (a) represents the size of the packets received by the client during the session, frame (b) represents the size of the packets received by a server over the same session, frame (c) shows the histogram of packet sizes corresponding to frame (a), and frame (d) shows the histogram of packet sizes associated with frame (b). Frames (e) and (f) of Figure 5 are the running averages of packet sizes at the client and at the server using the data shown in frames (a) and (b) of the same figure.

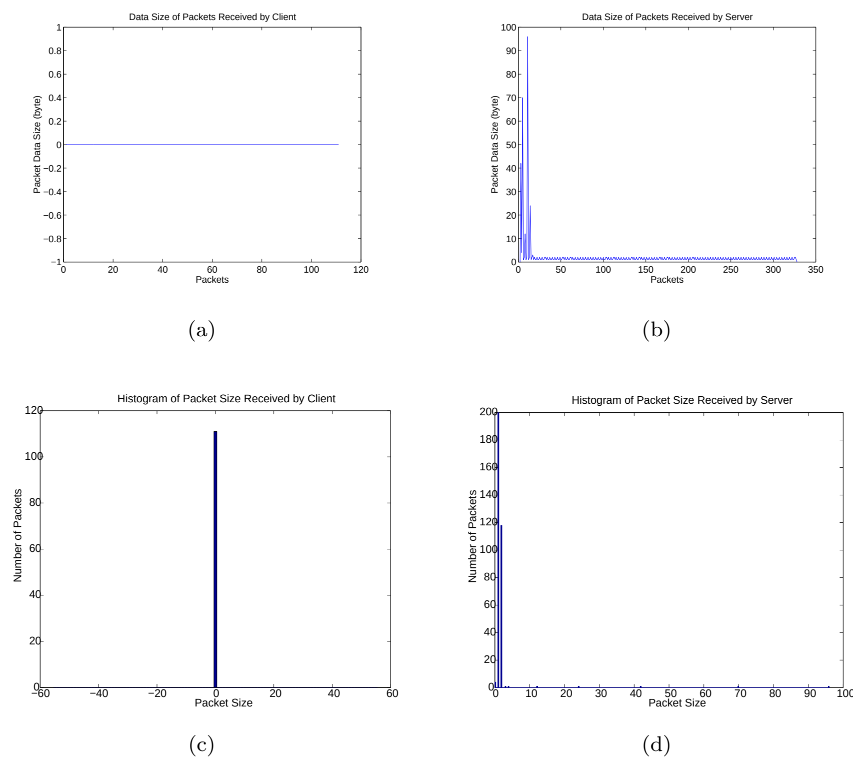


Fig. 4. Interactive tunnel session: frames (a) and (b) show the packet size received by a client and a server; frames (c) and (d) represent histograms using the data shown in frames (a) and (b), respectively

For the interactive attack session, a total of 111 packets were received by the client and total 329 packets were received by the server. Average packet sizes shown in Figure 4 frames (a) and (b) are 0 bytes and 2.088 bytes, respectively. For the streaming session, 1357 packets were received by the client and 834 packets were received by the server. Average packet sizes shown in Figure 5 frames (a) and (b) are 609.059 bytes and 0.5072 byte, respectively.

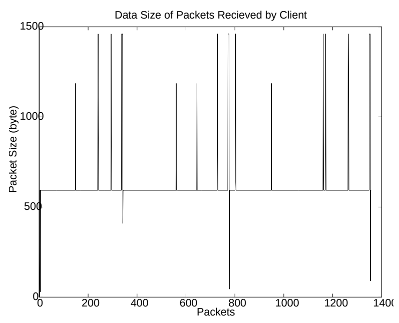
Using the data shown in the three figures, we now give examples of how each component of a behavior profile is specified.

(1) Packet Size: frame (a) in Figure 4 and Frame (b) in figure 5 show a sequence of packets with zero size being received at one of the traffic ends. While many zero packets appear in the normal web traffic (Fig. 3 frames (a) and (b)), such appearance only lasts for a "short" period of time and does not constitute an entire session as shown in the aforementioned figure frames. This feature alone, of course, does not separate normal web traffic from the tunneling traffic, but we can use it as one of the indicators. Another example follows. The size of packets sent from the client to the server for the interactive session indicates that the packet size for most packets is small: these are packets containing short commands. This feature is used to detect interactive sessions.

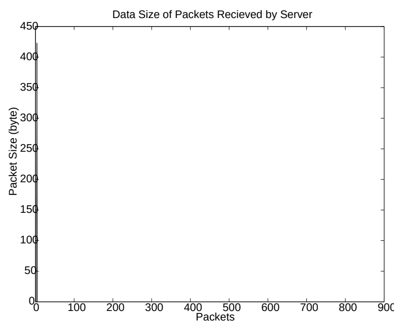
(2) Number of Packets: The system measures a session duration by the number of packets exchanged between a client and a server. Empirical results show the script session and normal web traffic sessions tend to have a small number of packets exchanged per session while interactive and stream sessions tend to have a large number of packets exchanged.

(3) Ratio of Large and Small Packets: We can use this feature to detect both an interactive session and a stream session. In an interactive session, one communication end usually sends small size packets with commands to be carried out. By collecting the number of "small" packets and comparing it to the total number of packets being sent, our system can detect an interactive session. In a behavior profile, a user can specify the "smallness" of a packet in terms of the number of bytes a packet contains. Experimental study also shows that when a stream is in progress, a large number of packets with a similar size tends to be sent to a client while a sequence of acknowledgement packets are sent in response to a server. The exact packet size varies depending on the streaming software used, but we can still exploit the pattern of same-sized packets to distinguish streaming session from normal web traffic.

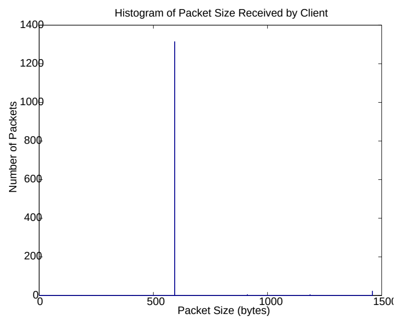
(4) Direction: The general direction of network traffic is another distinguishing characteristic of web activity. To detect an interactive session initiated from inside a local network, one can observe the number of packets being sent to the Internet and compare it to the number of packets



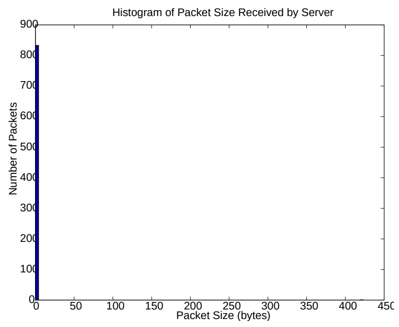
(a)



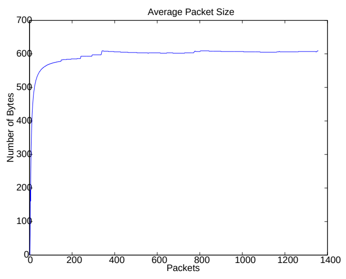
(b)



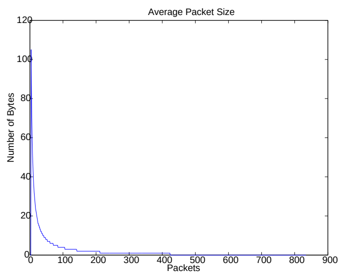
(c)



(d)



(e)



(f)

Fig. 5. Music stream session: (a) size of packets received by the client, (b) size of packets received by the server, (c) histogram of data in frame (a), and (d) histogram of data shown in frame (b).

received by the local machine. In a normal web session, a client asks for a webpage or a file from a server and the server then sends a relatively large amount of data to the client, while the client acknowledges the receipt of the data. Thus, the total number of packets received by the client is much larger than the total number of packets being sent. On the other hand, for a tunneled interactive session, a client tends to send many small packets over a "long" period of time to a server and receives relatively small sized packets when compared to normal web traffic sessions[3]. Figure 4 shows the initial session where this activity occurs. Frame (b) of this figure shows the packets being sent to the destination where we can see the small packet size. Also observe that the client is receiving a long sequence of acknowledgement packets (frame (a) of the same figure) from the server, which is abnormal.

[3] A similar pattern is observed when a client posts a message to a server in a normal web session, but the difference is the session duration for an interactive session is significantly longer.

(5) Average Packet Size: This feature in conjunction with the session length can give added evidence as to whether a stream session or an interactive session is in progress. As shown in Figure 5 frames (e) and (f), observing the average packet size over time shows that the average packet size received by a client in a stream session will reach a non-zero constant value while the average packet size being received by a server reaches almost zero. For an interactive session, the average size of the packets sent from a server to a client decreases as a session continues. This characteristic can be

Type	Outside IP	Outside Org	Num
S	207.239.241.41	winmed.westwindmedia.com	6
S	198.88.9.80	mail.alumrpi.edu	1
S	208.184.44.20	netshowc.mpo.intervu.net	1
S	64.89.104.44	abowms02.obmnet.com	7
S	63.250.208.18	wmcontent01.broadcast.com	1
S	207.46.184.73	msn.expedia.com	1
S	216.34.209.10	www.cj.com	1
S	140.177.203.60	www.wolfram.com	1
S	159.142.1.210	arpet.gov	4
Chat	63.160.183.240	chat.yahoo.com	1
OM	64.58.76.99	mail.yahoo.com	5
OM	205.188.160.121	aol.com	2

TABLE I
Summary of HTTP tunneling activities

seen in Figure 4 frame (b).

(6) Change of Packet Size Pattern: In a stream session, a significant change in packet size over five consecutive packets is a common event. Thus, the presence of this event indicates a streaming session is in progress versus a normal web sessions in which a large number of static pictures or data being transferred.

(7) Size of Total Packets Received: The amount of data being received also plays an important role for detecting abnormal activities. In normal web traffic, a client asks for information and a server provides the requested information. When we compare the amount of data transferred from a client to a server and vice versa, we find that a client receives a larger amount of data than was sent. A client that sends more data than it receives is an indication of tunneling activity.

The seven features are chosen to describe behavior profiles after analyzing the training data. Further study is necessay to verify whether the selected features are sufficient to detect arbitrary HTTP tunnecting activities.

Interactive Tunneling Session Behavior Profile: Our system detects interactive sessions by detecting the following behaviors: (a) the size of packets being sent by a client (originator) is relatively small; (b) while receiving small packets, a server sends a sequence of long acknowledgement packets in response; and (c) the number of small packets compared to the total number of packets being sent to a server is large; (d) the duration of a session is long.

Scripted Tunneling Session Behavior Profile: To detect scripted sessions the system looks for the following behaviors: (a) a short session with a relatively large data transfer; (b) a sequence of acknowledgement packets from a server; and (c) the amount of data received by a client compared to the amount of data received by a server.

Stream Session Behavior Profile: Compared to the normal web traffic, streaming audio, music, or video has the following distinctive behaviors: (a) the session duration is significantly longer; (b) the number of packets with a same size dominates the total number of packets in a session; (c) a client sends a sustained sequence of acknowledgement packets to a server; (d) a significant change in packet data size exists among neighboring packets in between two sequences of identical sized packets; (e) a constant packet size not corresponding to the maximum allowable packet size is observed; (f) the size of packets to the server is smaller; (g) the running average of the packet size to the server decreases over the duration of a session; (h) the number of packets received by a client is always greater than the number of packets received by a server.

V. Results and Discussion

In this section we present the experimental results using real operational web traffic of a large organization. We used 1Gbyte of network traffic collected by sniffing a network with a 2Mbits/sec data rate. Out of the 1Gbyte of data, just over 60% was web traffic. When we fed the web traffic to our system, it detected the activities shown in Table 1.

In the first column of the table, “S” indicates activities where audio, video, music, and other data are streamed to a computer of the organization and the OM stands for interactive mail activities using mailservers outside of the organization. A total of 38 different alerts were generated while processing the data (1 hour of net traffic representing communication between the Internet and the organization during 11 to 12 AM on a weekday). Out of the 38 alerts, 14 were from streaming music, 1 was from streaming radio broadcast, 8 were from interactive mail related activities, 1 was from chatting session, 7 were from streaming large data files, and 7 were false alarms. The sample experimental result data cannot be used to measure the effectiveness of the system but shows that a number of tunneling activities are taking place in the organization. No attack session was detected.

VI. Conclusion

In this paper, we presented a novel system to detect HTTP tunneling activities. In particular, the system uses user-defined behavior profiles based on packet flow directions, packet sizes, large and small packet ratios, average packet size, change of packet size pattern, connection duration, and size of total transfer of packets. Behavior profiles are used to detect interactive and scripted sessions as well as streaming data over the HTTP protocol. The system consists of three levels of analysis: the local level analysis module processes a limited number of packets within a sliding window to extract local behaviors; the session level analysis module processes all packets received for a particular session to capture global behaviors; and the verification module analyzes session transcripts if either the local level module or the session level analysis module detects suspicious activities. Application of actual operational data of a large organization shows the effectiveness and validity of the proposed system approach. We are currently working to expand the system to detect different attacks and to fuse the verification module with a transcript analysis system to perform a detailed analysis of suspicious transcripts.

References

[1] Y. Zhang and V. Paxon, "Detecting backdoors," *Proceedings of 2000 USENIX Security Symposium*, August 2000.

[2] A. Valdes and K. Skinner, "Adaptive, model-based monitoring for cyber attack detection," *Proceedings of the 2000 Recent Advances in Intrusion Detection Conference, Toulouse, France*, October 2000.

[3] P. Porras and A. Valdes, "Live traffic analysis of TCP/IP gateways," *Proceedings of the 1998 Internet Society's Network and Distributed Systems Security Symposium, Athens, Greece*, March 1998.

[4] "Vividon focuses on video streaming," *http://boston.bcentral.com/boston/stories/2000/10/23/newscolumn2.html.*

[5] "Using HTTP as an RPC transport," *http://msdn.microsoft.com/library/psdk/rpc/ pv-http_7h4k.htm.*

[6] "ICEcap manager," *http://www.networkice.com/products/icecap_manager.html.*

[7] S. Webster, "PSplice," *Personal Communication.*

[8] R. Cunningham and R. Lippmann, "Host-based bottleneck verification efficiently detects novel computer attacks," *Proceedings of IEEE Military Communications Conference*, 1999.

[9] L. Rossey, R. Cunningham, D. Fred, J. Rabek, R. Lippmann, J. Haines, and M. Zissman, "LARIAT: Lincoln adaptable real-time information assurance testbed," *Submitted for publication.*

[10] "HTTP tunnel," *http://www.nocrew.org/software/httptunnel.html.*

[11] D. Gregg, W. Blackert, D. Heinbuch, and D. Furnanage, "Analyzing denial of service attacks using theory and modeling and simulation," *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, pp. 205–211, United States Military Academy, West Point, NY, June 2001.