

A Modified VPN Solution Based on Hybrid Tunnels

Prasanna. P*, Chellappan. C**, Rhymend Uthariaraj. V ***

Department of Computer Science and Engineering, Anna University, Chennai
*pprasannachn@yahoo.com , **drcc@annauniv.edu , ***rhymend@annauniv.edu

Abstract-Conventional SSL tunnel of SSL-based Virtual Private Network is symmetric, in which the data must be encrypted at one end and decrypted at the other end, or contrariwise for the reverse direction. Because all data flows of VPN are relayed by VPN server via SSL tunnels, those symmetric SSL tunnels cause a lot of computational load concentrated in VPN server, and make it the bottleneck of VPN. This paper proposes a modified solution to eliminate the bottleneck prevailing in the existing VPN solutions: The VPN based on Hybrid Tunnel. It is done using Message Pushing (MP) algorithm. In this solution, portion of the computational load is distributed to disengaged internal application servers by establishing data link layer type tunnel connections between VPN server and Application servers, thereby increasing the throughput of VPN server and response time for the VPN clients

I. INTRODUCTION

Secure socket layer (SSL) protocol [2] was initially proposed by Netscape to enhance Web security, but additional computation overhead introduced by SSL always reduces the throughput of Web server by one or two orders of magnitude [3]. SSL VPN is a secure remote access solution based on SSL/TLS [4] protocol, and develops rapidly. According to the prediction of Gartner Company [5], SSL VPN market will grow more than 170% per year. Compared with other virtual private network technologies, SSL VPN has the following outstanding advantages: low cost, easy-to deploy, fine-grained access control, etc. But its performance and scalability are also hampered by the computation overhead of SSL protocol.

VPN server is the key equipment to construct a SSL VPN. VPN server was always deployed in the front of a LAN, and acts as a portal. After a VPN client was authenticated by the VPN server, a SSL tunnel will be created, connecting the client and the VPN server. Thereafter, the VPN server relays data between the VPN client and internal application servers. Inside the LAN, data communication between VPN server and application servers can be either in plain text, or protected by additional internal SSL tunnels. It's up to internal security requirement. Before being forwarded, all data flows of the VPN should be encapsulated or unwrapped at VPN server according to SSL protocol. VPN server is computing intensive, and communication quality of the whole VPN is determined by the computing power of VPN server. Research shows that the bandwidth utilizations of all open-source Linux-based SSL VPN solutions are less than 50 percent bandwidth of the 100Mb/s fast Ethernet, even using Pentium IV 2.0G platform [5]. Vendors usually have to install an expensive hardware SSL accelerator in VPN server to meet the demand of high-end applications or large scale LAN cases. For the purpose of

improving the performance and scalability of SSL VPN, this paper proposes a hybrid tunnel solution for VPN, accompanied with Message Pushing algorithm. In this solution, portion of computational load was distributed to disengaged internal application servers using data link layer tunnel connections, and thus the overall VPN throughput can be improved greatly.

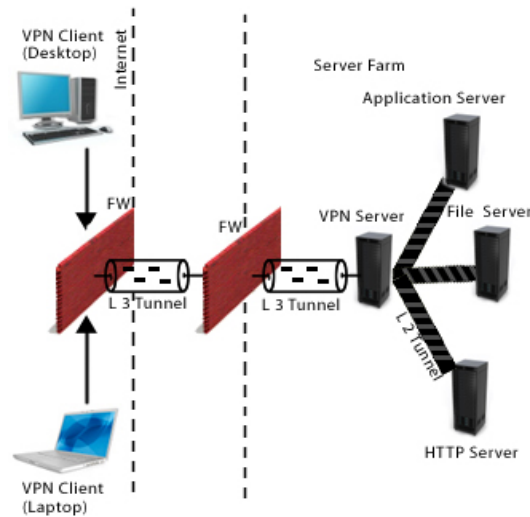


Fig 1. A simple Hybrid Tunnel example

II. SYMMETRICAL TUNNEL

SSL tunnel is the overlay networking facility for creating a SSL VPN on top of existing Internet or IP based network. Before passing through SSL tunnel, an IP packet is compressed and encrypted in advance according to SSL/TLS protocol. After that, it is carried by the payload of an IP data packet in SSL tunnel. When creating a SSL tunnel, the SSL Handshake Protocol was used by communication peers to authenticate each other, and to negotiate the encryption and message authentication coding (MAC) algorithms, coupled with cryptographic keys. Those algorithms and keys are used to protect data in subsequent communication, and called as *cipher spec*. In transfer phase, both peers use the cipher spec. to do SSL encapsulation or decapsulation, according to the process defined by SSL Record Protocol.

SSL tunnel is full-duplex communication channel. For conventional SSL tunnel, data transferred in both directions is encapsulated using the same cipher spec and both peers have two roles: sender peer and receiver peer. The computational loads of both peers are approximately symmetric, when data passing through the tunnel. For each outgoing packet,

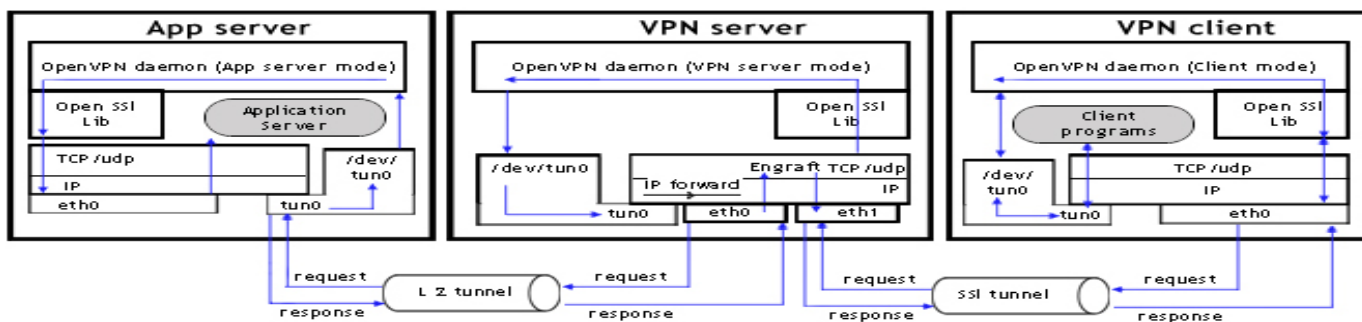


Fig 2. Packet Flow between VPN server, Application server and VPN client

the sender peer needs to do compressing, computing MAC, and encrypting. The reversed processes are needed in the receiver peer: decrypting, verify MAC and decompressing. Considering above characters of conventional SSL tunnel, we call it symmetric SSL tunnel. VPN server is essentially a SSL proxy accompanied with authentication and access control capabilities. VPN server authenticates client and creates SSL tunnel with it, and then relays the request from client to internal application servers. After receiving response from application servers, VPN server relays it to the client via SSL tunnel.

III. ASYMMETRICAL TUNNEL

Conventional SSL VPN solutions use symmetric SSL tunnels as their overlay networking facility, and VPN server is the common end of all those tunnels. Therefore all data flows of VPN must pass through VPN server, and a lot of computational load is concentrated in VPN server. VPN server becomes the bottleneck, but CPU utilizations of internal application servers are always low. The more application servers exist, the worse. If some computation load can be transferred to those I/O intensive but CPU disengaged application servers, the overall performance of VPN should be improved.

VPN server must decrypt the request packets from clients, because it needs the plain request to do authentication, access control and finally forwarding it to the correct application server. In contrast, the plaintext responses from application servers are not necessary for VPN server. According to that, a novel asymmetric SSL tunnel (AST) [1] is : If the cipher spec has been transfer to application server via secure tunnel, the application server can do SSL encapsulation immediately after the response IP packet generated, and VPN server just forwards the encapsulated packet to client. Using asymmetric SSL tunnel, output data is SSL encapsulated by internal application servers and computation load of VPN server can be reduced.

IV. HYBRID TUNNEL

In Asymmetric tunnel, where the load of VPN Server was distributed to internal application server during their disengaged time and their internal communication between them is by using the network layer protocol. In case of hybrid tunnel, we are communicating with the internal application servers using data link layer protocol i.e., the communication is made using the frames instead of packets. Research shows that communication at data link layer is much faster than at the network layer for the intranet communications purposes. This has been taken as a core for our project work and we tried to implement this fact into the existing asymmetrical tunnel environment so that the throughput of the VPN server can be increased a lot by using our hybrid tunnel as shown in Fig. 1.

When the packet from VPN client arrived at the network interface card (NIC) of the VPN server, it will be delivered up, passing through TCP/IP protocol stack and socket layer one by one, and finally the VPN service process will receive it, which runs at user level. VPN service process chooses an outgoing tunnel according to the destination address of this packet, and then passes it down to NIC, layer by layer. Here exist at least two times of data copying between kernel space and user space, and also other processing of each layer, including generating new checksum in TCP/UDP protocol layer.

Message Pushing (MP) means that it directly inserts the Layer 3 Payload on to the NIC buffer setting Media Access Control (MAC) address of the internal application server. This reduces the copying of the data from kernel space to user space and vice versa is avoided. VPN server must decrypt the request packets from clients, because it needs the plain request to do authentication, access control and finally forwarding it to the correct application server. VPN Server also transmits the cipher spec and other details needed by the application server for encapsulation and decapsulation of the payload. When the application receives the payload data, it unpacks the payload, determines the cipher spec that is to be used for this payload and processes the payload accordingly.

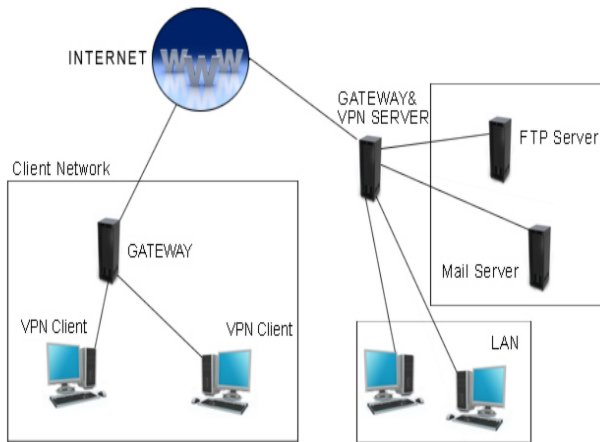


Fig 3. Experimental Test Bed

The response from application for the VPN clients are again encapsulated at application server itself using the cipher spec residing with them and are transmitted to the VPN server for the forwarding them. On receiving the payload from application server at NIC of the VPN server, the payload header is analyzed and using the client base information, the VPN server does only the forwarding of the payload from it to the VPN clients. The work consist of three main modules namely, VPN server Capturing Module, Monitoring Module and Application server Capturing Module.

A. Capturing Module at VPN Server

Message Pushing means it captures the packets arriving at the NIC of the VPN server authenticates and encapsulation and decapsulation till the connection between the VPN server and VPN client is broken; the packet between them is shown in Fig 2.

B. Monitoring Module

This module is responsible for monitoring the workload of the application servers present in the server farm, and determines the processing load and idle time of the each application servers. The information so determined are stored in the VPN server in hash based linked list which enables fast identification of the application server during the demand. Parameters such numbers of jobs currently running, main memory usage, percentage utilization of the processor, number of zombie process, number of sleeping process, etc., are monitored by this module and they are stored.

TABLE I HARDWARE DETAILS OF THE SYSTEM USED FOR OUR TEST BED

System Name	CPU	Memory	OS	Quantity
VPN Server	Pentium IV 2.4 GHz	2 GB	Linux 2.4	1
Client Gateway	Pentium IV 2.4 GHz	2 GB	Linux 2.4	1
Application Server (FTP)	Pentium IV 2.4 GHz	1 GB	Linux 2.4	2
VPN Client	Pentium IV 1.8 GHz	512 MB	Windows Xp, Red Hat	2

C. Capturing Module at Application Server

As the communication with the application server is done using Data Link Layer protocol, a module is developed which captures the frames sent from the VPN server, which then analyzes these frames using cipher spec and other details forwards them to appropriate Application servers, depending upon the client info base, present at the VPN server. The module determines the Application server to be used for a particular VPN client, does all the encapsulation and decapsulation till the connection between the VPN server and VPN client is broken. which is used for the communication between the VPN server and Application server as shown in Fig. 2. The message after being decapsulated and decrypted are sent to destination where they have to reach by the application server. The destination machine instead forwarding the replies to VPN server, are forwarded to Application server which was involved in decapsulation and encapsulation. Thus the replies to the VPN client are decapsulated at the application server itself, hence the VPN server acts as forwarding agent at most of times.

When the VPN client wants to communicate with VPN server, usual TCP connection are established, cipher spec is shared between them. Whenever packet arrives from VPN client machine, the VPN server capture module captures it and compares it against the VPN clients address stored in hash based table, if match is found, those packets are authenticated by the module. Authenticated packets are transmitted to application servers by daemon which keeps track of idleness of application servers; we have developed API for the monitoring module to use. These API returns data structure containing the application server details such IP address, MAC Address, Subnet mask, etc..

A test bed was created using the system of the following configuration and as shown in the Fig. 3. It consists of a VPN server, two VPN clients, one Client Gateway and two Application servers configuration of which are shown in Table I.

Modules were developed using C language and other low level function calls for handling the packet buffer, inserting the packet into NIC buffer etc. The modules were developed from the scratch from packet capturing, transmission of packet, encryption, decryption were also done using our own encryption and decryption algorithm for our experimental test bed. FTP service as used [base] was used for testing our test bed setup.

V. CONCLUSION AND FURTHER DISCUSSION

Hybrid Tunnel solution improves the VPN throughput considerably, however currently we have not encrypted the transmission of the messages at layer 2 level as we assume the server farm will be secured one. On larger server network where many administrators controlling each of the servers, security is vulnerable, hence layer 2 transmission can also be encrypted. The performance of throughput may decrease at cost of this encryption.

REFERENCES

- [1] Jingli Zhou, Hongtao Xia, Xiaofeng Wang, Jifeng Yu, "A New VPN Solution Based on Asymmetrical SSL Tunnels", IEEE Proceeding 2006
- [2] Alan O.Freier, Philip Karlton, "The SSL Protocol Version 3.0 [EB/OL]". Oct.2004.
- [3] Apostolopoulos, G.; Peris, V.; Saha, D.; "Transport layer security: how much does it really cost?", Proceedings. IEEE. Volume 2, 21-25 March 1999, vol.2 pp.717 - 725
- [4] T. Dierks and C. Allen, "RFC2246: The TLS Protocol Version 1.0", <http://www.ietf.org/rfc/rfc2246.txt>, Jan. 1999.
- [5] Gartner Company. <http://www3.gartner.com/>.