R. Roman, J. Zhou, and J. Lopez, "Casual Virtual Private Network", *International Journal of Computer Systems Science & Engineering*, vol. 3, pp. 185-192, 2005.
NICS Lab. Publications: https://www.nics.uma.es/publications

Casual Virtual Private Networks

Rodrigo Roman[1], Javier Lopez[2], Jianying Zhou[1]
[1] Institute for Infocomm Research
21 Heng Mui Keng Terrace
Singapore 119613
stucrr@i2r.a-star.edu.sg, jyzhou@i2r.a-star.edu.sg
[2] Computer Science Department, E.T.S. Ingenieria Informatica
University of Malaga, 29071 - Malaga, Spain
jlm@lcc.uma.es

Abstract. Virtual Private Networks (VPNs) provide a cost-effective way for securing communications using public and insecure networks like the Internet. The main purpose of a VPN is to securely and transparently connect two or more remote networks to form virtually a single network, using centralized security policies for better management and protection. However, in certain scenarios, users may not require such a transparent access to the resources within their networks, but only want temporary secure access to internal services based on their own demands. We call the network architecture with such a feature as *Casual VPN*. In this paper, we present the notion of Casual VPN, and explain why traditional VPN architectures and protocols are unable to offer Casual VPN services. We also propose and define the operation of a particular Casual VPN architecture, *C-VPN*, which additionally allows the management of TCP and UDP-based protocols.

1. Introduction

Virtual Private Networks (VPNs) [1] allow individuals and business to create and maintain secure communication channels between their own local networks using public and insecure networks, like Internet, instead of private and leased lines. The main purpose of a VPN is to securely and transparently connect two or more remote networks to form virtually a single network.

Network architectures defined by VPNs are inherently more scalable and flexible than pure private networks because they allow organizations to add and remove branch offices in an easier way. Other benefits of VPNs include obtaining almost the same capabilities of private or leased lines at much lower cost, and providing roaming users (or "Road-Warriors") with secure connections to their corporate or personal networks whenever they need them.

VPNs achieve the same level of protection as private networks using security mechanisms like encryption and authentication schemes. Those security mechanisms are centrally managed by a pre-defined security policy, which controls all communications inside the VPN by dictating when and how the protection is applied. In most cases, for security reasons, VPN users cannot change the security policy, and they cannot dynamically choose whether their communications are going to be protected or not.

This high level of protection and connectivity achieved by VPNs, although desirable and necessary, is not useful for networks which do not want to provide transparent access to their resources within their networks, but only want to provide temporary secure access to internal services based on users' demands. Although it is possible to provide protection for every service using modified clients and protocols, it could be useful to have an architecture that provides a user-initiated, transparent and secure connection between networks without securing every individual service. We call the network architecture with such a feature as *Casual VPN*.

In this paper, we present the notion of Casual VPN, and explain why traditional VPN architectures and protocols are unable to offer Casual VPN services. We also propose and

define the operation of a particular Casual VPN architecture, *C-VPN*, which additionally allows the management of TCP and UDP-based protocols.

2. Casual VPN vs Traditional VPN

We define Casual VPN as a VPN architecture whose purpose is not to relay network messages in a transparent and secure way between two or more distant networks, but to protect the application-level information flow between those networks when users decide to do so. The security policy in Casual VPNs is defined implicitly: every single flow of information sent through a Casual VPN is protected following the users' preferences, but it is not mandatory to use the Casual VPN services to send information between two networks.

Casual VPNs are very useful for organizations (such as university laboratories) that work in collaborative scenarios. The users of these scenarios do not need to setup a permanent private network because they do not need to continuously share the resources available on their networks with other peers, and the information shared between them is usually not sensitive and does not need protection. However, in some cases, they may need to access to the services provided by the other peers (such as telnet connections or voice conferences) in a secure way, in order to protect the information.

An intuitive solution might be implementing a Casual VPN based on any of the traditional VPN architectures. However, this approach is not possible. For a better understanding of the reason behind it, let us first investigate the existing VPN schemes and their protocols.

The architectures that provide VPN solutions can be classified into three categories [1]: secure VPNs, trusted VPNs, and hybrid VPNs.

- *Secure VPN*: Information packets travel through the network with cryptographic protection. The VPN is within the organization or company, and local personnel manage the internal security policy. Technologies used in this type of VPN architecture are IPsec [2], SSL [4,5], and SOCKSv5 [6].
- *Trusted VPN*: Information packets travel through the network over a pre-defined and protected path. In this architecture, management tasks rely on the service provider. Technologies used to provide this type of VPN service are MPLS [7], ATM circuits, and frame-relay.
- *Hybrid VPN*: This architecture combines the security properties of secure VPN and trusted VPN.

Obviously, neither trusted VPN nor hybrid VPN can provide what a Casual VPN needs because security services are managed by a service provider. Consequently, users do not have the possibility of choosing whether to apply security to their communications. Hence, we only need to investigate the Secure VPN solutions in order to learn if they are suitable for the design and development of a Casual VPN.

- IPsec-based Secure VPNs

  IPsec [2] is a protocol suite that operates at the network layer and is designed to protect data confidentiality, authentication and integrity for communications among corporate LANs as well as between corporate LANs and Road Warriors [1]. IPsec is an obligatory part of IPv6.

[1] Road Warriors refer to employees who are working outside the organization's facilities and want to access to the corporate networks and resources in a secure way.

In order to open a secure communication channel, each pair of hosts using IPsec must configure a set of *Security Associations* (SAs) [2]. Those SAs define the peers involved in a secure communication (hosts, transport-level ports etc.), the type of protection applied to the secure channel, the algorithms used for encryption and/or authentication, and the keys needed.

IPsec is becoming a success as an industry-wide standard. Thus, it could be interesting and advantageous to reuse traditional IPsec-based VPN schemes to provide Casual VPN services. Unfortunately, this is not possible. The main reason is that a user cannot open a communication channel to a single (*ip address, port)* combination (specified by a security association) and send both secured and non-secured packets through it.

We further review two basic scenarios of IPsec to verify our previous statement. In the first scenario, IPsec functionality is provided by the gateways at the boundary between the LAN and the external Internet. In this case, the security policy is enforced to all LAN users since the secure channel is created between the gateways. Hence, users cannot choose when to apply the protection to the channel.

In the second scenario, IPsec functionality is provided by users' computers. In this case, users can download the security policy from a centralized manager or change the security policy by themselves. In any of the two possibilities, a user cannot open a communication channel that provides security for some packets while other packets are sent unprotected to the same destination. This issue could be solved using multiple SAs and content-based routing, but only if all the protected traffic can be distinguished "a priori" from the unprotected traffic.

- SSL and SOCKSv5-based Secure VPNs

The SSL protocol [4] was firstly developed by Netscape Communications Corporation, and later evolved as a widely adopted standard for e-commerce communications. SSL works over the transport layer and below the application layer, and its main task is to provide server and client authentication and channel encryption. This protocol has evolved to TLS [5].

The Internet Engineering Task Force (IETF) originally approved SOCKSv5 [6] as a standard protocol for authenticated firewall transversal. The SOCKSv5 protocol works over the transport layer, setting up a virtual circuit between a client and a host on a session-by-session basis, providing monitoring and strong access control based on user authentication.

VPN architectures based only on SSL [8] may provide users with a Casual VPN solution since the users can choose when the channel must be secured for a specific application. However, this solution cannot be applied to all scenarios since applications based on UDP cannot be protected. In addition, this solution is more oriented to serve Road Warriors, rather than creating a secure infrastructure between networks.

VPN architectures based on both SSL and SOCKSv5 protocols [9,10] cannot offer a Casual VPN solution because SOCKSv5 manages the outgoing packets with a central security policy. When an application wants to send packets to a specific destination, every packet is automatically intercepted and redirected to a proxy server, which then forwards all packets depending on the central security policy. Therefore, users have no choice regarding the security features of their communications.

[2] Note that IPsec does not require automatic negotiation of security associations: they can be configured by other means (e.g., IKE, manual keying, etc.).

In summary, existing VPN solutions cannot be applied to Casual VPNs, because of two main reasons.

1. *Centralized Security Policies*. Security policies require that all users and systems of an organization must follow certain procedures in order to avoid security breaches. All existing VPN solutions are primarily designed to be administered using a centralized security policy, which provides an efficient and homogeneous management, that users cannot override.
2. *Design of Security Protocols*. Security protocols, like IPsec, are not designed to simultaneously allow a secure and a non-secure communication to the same destination. They assume that whenever a security association between two peers is established, all subsequent communications must conform the procedures established.

The two previous reasons are absolutely essential for creating a centralized VPN infrastructure, since they enforce the application of security policies. However, Casual VPN needs more flexible security policies since users may configure their own security policies.

3. Casual VPN Architecture

Since existing VPN architectures cannot completely fulfill the requirements of a Casual VPN, it would be important to provide such a technology. We have developed a gateway-to-gateway Casual VPN solution, *C-VPN*. C-VPN can be seen as a security service. If a user wants to protect a communication channel, he/she will explicitly connect this channel to the VPN service. This VPN service receives the packets from the user, and sends them through a secure channel that fulfills the requirements of confidentiality, authentication and integrity. The information that, according to the user, does not need to be secured is sent unprotected through the public network.

The architecture of C-VPN can be seen in Figure 1. There are several LANs, and there can be more than one computer providing the VPN service per LAN. Clients of the LANs use those services to establish secure connections between LANs, where service providers act as proxies. This architecture is similar to Detour [3], although C-VPN provides security between peers, not routing.

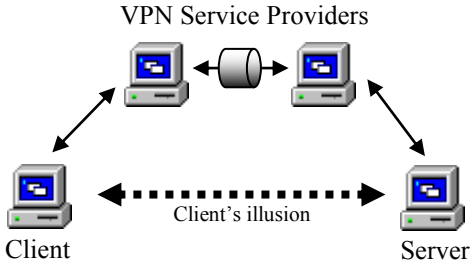

Figure 1. Basic C-VPN Scheme

When a client wants to establish a secure communication channel with a remote server located in another LAN, it must first send a notification to the local VPN provider in its own LAN. Then, it must launch the application, which will open a connection with the selected VPN provider. At that moment, the local VPN provider will contact and negotiate a secure channel with another VPN provider in the target LAN. This remote VPN provider will open a connection with the remote server.

When all channels are established, the information packets sent by both peers will pass through the secure channel between the VPN providers. In this way, all communications between peers in that session are secured.

Since each LAN is isolated – they do not know how to discover the VPN provider of other LANs located inside the VPN – there must be a manager, known to all the components of the VPN. That manager acts like a certification authority, providing information about the computers inside the VPN (e.g., their location and their public keys) and information about the master protocol list (i.e., application protocols supported by the network). The manager can operate from any of the LANs of the VPN, although it may rather reside in the headquarters of an organization.

3.1 C-VPN Components

Next we explain in more detail the components of C-VPN (Figure 2), specifying their location and functionality:

- **Main Server (S1)**: This process is the manager mentioned above. There is only one main server in the whole VPN. Preferably, it should run in the main LAN of the organization. Its tasks include controlling other types of server processes and providing information about other servers (secondary and secondary masters) in the VPN.
- **Secondary Server (S2)**: This process is the service provider of a LAN. There is (at least) one S2 per LAN in the organization. It attends the connection requests from the clients, carrying out some of the negotiations required for setting up secure channels between two peers.
- **Secondary Master Server (S2M)**: As mentioned, there can be more than one secondary server in one LAN. Thus, the S2M manages all S2s in the LAN and regulate their workload according to users' needs. S2M also controls the local users (i.e., the users inside its LAN) and decides which of them can login into the VPN. There exists only one S2M per LAN in the organization.
- **Auxiliary Servers (SAux)**: Auxiliary servers are created in real-time by one S2. These are the processes that really run the communication through the secure channel. There is one SAux process per peer client connection.
- **VPN Clients (C)**: These are the client applications run by the LAN users. Each client uses the services provided by the C-VPN processes to open the communication channel from the application process to the service provider process (S2).
- **Road Warrior Clients (RW-C)**: They represent the client software of those users that belong to the organization but are physically located outside of the LANs of an organization.

Every server in the C-VPN has a public/private key pair and a public-key certificate. Those keys and certificates are used for authenticating themselves against other VPN entities [3] and to create secure channels. Since C-VPN is oriented to collaborative scenarios, the Main Server (S1) can be used as a *Certification Authority* (CA).

For the registration process in the VPN, the servers use their own certificates for authenticating themselves. The certificate of the CA is required for validating the certificates, implying that all servers must store the CA's certificate.

When creating a secure connection on the request of a VPN client, not only in-house protocols but also standard protocols (e.g., SSL, or even IPsec) can be used for securing those gateway-to-gateway communications.

[3] "VPN entities" is a general term referring to different servers of the VPN service (see the list above).

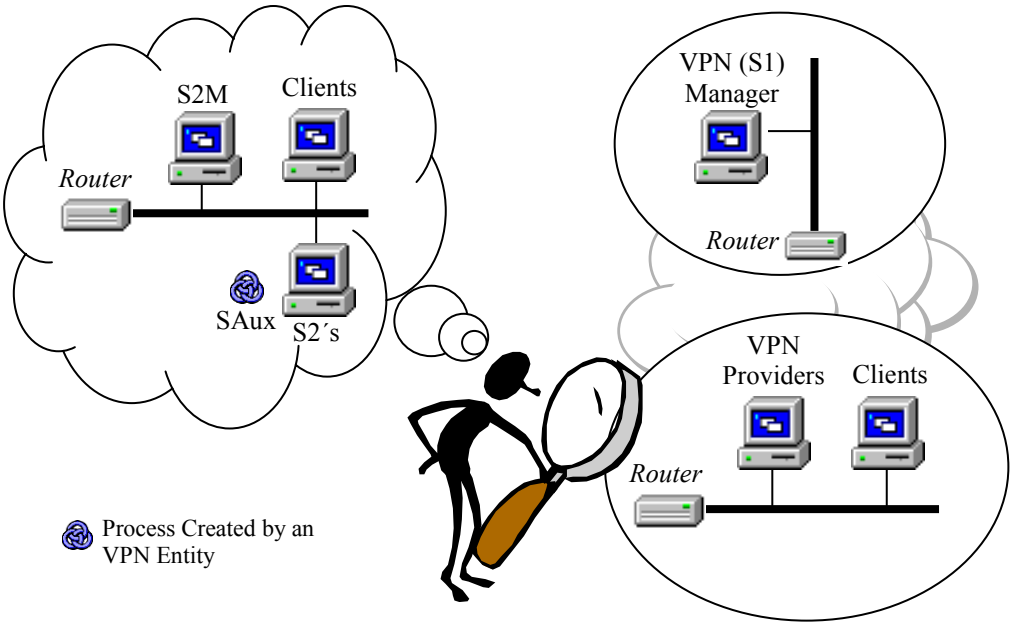

Figure 2. C-VPN Architecture

3.2 Packet Capture

C-VPN service providers receive the packets directly from the client applications. For this reason, Secondary Servers (S2) must bind all possible ports used by all the potential protocols and services that will be used by VPN clients (e.g., TCP port 80 for HTTP, TCP port 21 for FTP, UDP port 554 for RTSP...). When an application connects to a S2, this one receives all the information packets transmitted by that application, and forwards them to the appropriate Auxiliary Server (which manages the secure channels between the peer gateways).

Nevertheless, this form of operation is not enough to manage correctly all the protocols that are based on TCP and/or UDP. Every protocol has some special features that must be controlled explicitly, but such features are never known at the transport layer. For instance, the HTTP protocol makes more than one connection to the port 80 for retrieving one single web page. Consequently, if the VPN entities are not aware of this fact, packets cannot be forwarded correctly. This is the problem that transparent proxies face when relaying packets over the network.

We have realized that the most popular Internet protocols share common behaviors when dealing with incoming and outgoing packets. The results can be read in the next subsection, "*Common Characteristics of Internet Protocols*". All C-VPN entities must be aware of the characteristics of all protocols managed by the VPN. Adding a new protocol to the architecture only means pointing out its characteristics. This is achieved by defining them inside a XML configuration file called *Master Protocol List*, stored in S1. When the entities of the VPN register in their respective servers, they get the Master Protocol List, and proceed with the necessary arrangements to manage the corresponding protocols. The following is an extract of configuration file example:

```
<protocol id="2">
        <application>
                <name>FTP</name>
```

```
                <exe>mozilla</exe>
                <param>ftp://%%IPD%%PARAM</param>
        </application>
        <architecture>
                <type>clientserver</type>
                <proto>tcp</proto>
                <port>21</port>
        </architecture>
        <multiplerequest is="no"/>
        <channels>
                <newchannels cani="yes"/>
                <default>
                </default>
        </channels>
        <timeout>
                <call>2000</call>
                <idle>20000</idle>
        </timeout>
</protocol>
```

3.3 Common Characteristics of Internet Protocols

The most important Internet protocols (i.e., hypertext transmission [11], file transfer [12], remote connection [13], multimedia transmission [14,15]) share certain characteristics that define some aspect of the behavior of the protocols. Those characteristics can be mandatory (existing in all the protocols) or optional (only some protocols have them), and outlined as follows.

- *Transport Protocol (mandatory)*: All Internet protocols have a main transmission channel, and that channel works over TCP or UDP in the transport layer [4].
- *Protocol Paradigm (mandatory)*: Internet protocols works following one of two paradigms, client/server or client/client:
  - In the *client/server* paradigm, a client sends a request to a remote server (in a well-defined port) in order to receive a specific service. The flow of data between client and server will be bidirectional. An example of this paradigm is the HTTP communication between a web browser and a web server.
  - In the *client/client* paradigm, communication can be started by either client, and both can act either as client or as server. The flow of data between client and server will be unidirectional, and in most cases the service will start properly only when both clients are connected. An example of this paradigm is the two-way transmission of voice and video using RTP and RTSP.
- *Multiple Requests (optional)*: Some Internet protocols make multiple requests to the same server port during the same session in order to create many communication channels. This feature is not present in UDP-based protocols since the notion of channel does not exist. An example can be the HTTP protocol.
- *Multiple Channels (optional)*: Some Internet protocols have one or more secondary transmission channels, based either on TCP or UDP. There are two types of secondary channels:
  - *Static*: The application connects to well-defined ports that the server has bound "a priori". Example: RTP protocol (ports for the RTP protocol and the RTCP protocol).
  - *Dynamic*: The channels are created during the session, and not before. The port of the channel is usually negotiated between the two peers. Example: Protocols RTSP and FTP.

Therefore, and provided that the software working at the transport layer knows the characteristics of the protocols, it is feasible to receive and handle the packets corresponding to

[4] This *Transport Protocol* field can also refer to other special protocols, sucha as SCTP.

those protocols. That software must have a library for packet analysis in order to find out control packets that carry information about creation of dynamic channels.

4. C-VPN Operation

4.1 Setup

Every VPN provider has to be installed in its respective computer, and must have a certificate signed by the Main Server (S1). All the servers must know where to locate the Main Server. Additionally, the Secondary Servers (S2) and VPN Clients of a LAN must know where to locate the Secondary Master Server (S2M) that manages that particular LAN.

In order to initialize the C-VPN services, S1 must be run at the very beginning. After this, all S2Ms can be initialized. They register themselves in the S1 and receive the Master Protocol List. Finally, all S2s are run and registered in their respective S2M. By using the Master Protocol List (received from the S2M), S2s can proceed with an auto-configuration.

The registration and removal of S2M and S2 entities only requires an information interchange between those entities and their managers (S1 and S2M respectively), and the subsequent update of certain data structures. Thus, the VPN can evolve over time including and deleting peers without rebooting itself, and with no manual configuration in the managers' side.

4.2 Normal Operation

The most basic protocol that C-VPN manages is a client/server TCP protocol, like *telnet* or *ftp*. C-VPN sets up and provides a secure channel between the client and the server in two well-differentiated phases: *connection* and *post-connection*.

Prior to the connection phase, the user selects, through the VPN client interface, the specific communication application to run. The connection phase is then started. In this phase, the VPN client contacts the appropriate VPN entities and requests a connection to the remote server. The goal of this phase is to create a secure channel, controlled by the entities of the VPN, between the two peers.

Next we detail the operation of the connection phase (Figure 3). For simplicity, we use the *telnet* service as an example.

1. The VPN client contacts the Secondary Master Server (S2M) in LAN *A*. S2M chooses a Secondary Server (S2) depending on the S2's workload and historical statistical information, and returns its address to the client.
2. The VPN client opens a connection with the S2 assigned, and requests a *telnet* connection. S2 stores a "ticket" containing what client is going to open a connection, the application/protocol which is going to be used in the connection, and the address of the remote server. Then, S2 informs the client that it can start its application.
3. The application (*telnet*) opens a communication channel with S2. S2 verifies if this connection corresponds to a ticket reserved in step 2. If this is the case, S2 creates an Auxiliary Server (SAux) process, and provides it with the information contained in the ticket. It is important to note that SAux inherits the communication channel opened with the *telnet* application. From this moment, SAux will receive all packets sent out by the *telnet* application.
4. SAux needs to locate the remote S2M in LAN *B*. Thus, SAux queries S1. Afterwards, SAux contacts the remote S2M to get the address of certain S2 in LAN *B*.

5. S2 in LAN *B* creates SAux', and helps to establish the communication between SAux' and SAux in LAN *A*. Finally, SAux' connects to the *telnet* server. At this stage, the *telnet* client (in LAN *A*) and the *telnet* server (in LAN *B*) are connected through the secure channel established between SAux and SAux'.

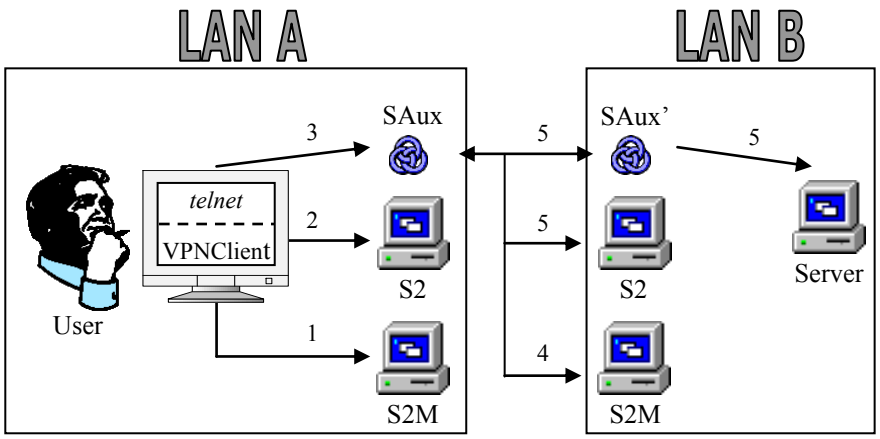

Figure 3. Connection Phase

In the post-connection phase, both client and server applications interchange information through the secure channel. The entities of the VPN will guarantee that the communication is carried out transparently and correctly. The detailed operation of the post-connection phase is as follows.

6. As previously stated, whenever the *telnet* client in LAN *A* sends out a packet, this is received by SAux. SAux forwards the packet to its partner SAux' in LAN *B* through the secure channel. Similarly, SAux' forwards the packet to the *telnet* server.
7. If the remote *telnet* server responses, the reply is sent back to SAux', which forwards the packet to SAux in LAN *A*, and SAux sends the packet to the *telnet* client.
8. Once the *telnet* client closes the session, both SAux and SAux' close the secure channel, and are self-destroyed.

4.3 Advanced Operations

C-VPN manages an incoming packet according to the characteristics of the communication protocol used. As we stated in the previous subsection, the simplest protocols that can be managed by C-VPN are based on a client-server paradigm using TCP.

In this section we show how C-VPN can handle other characteristics that are more difficult to manage than those in client-server paradigm. The management is based on the operation procedure studied before. However, some differences must be pointed out. We explain now the different cases.

UDP-Based Client-Server. For a better understanding about how C-VPN manages this situation, an intrinsic characteristic of the UDP protocol must be reminded. When a TCP-based server receives a connection request, the system deviates the communication to a new port, and a new channel is thus created for that request. However, in UDP there are no connection requests since the same channel receives all communications.

Therefore, when the S2 receives a communication request from a UDP application that follows the client-server paradigm, SAux cannot inherit the channel from S2 because, basically, there is no channel to assign. Thus, S2 must receive all the packets from the client application and forward them to an appropriate Auxiliary Server.

Client-Client Paradigm. In this paradigm, any peer can start the communication (e.g., audio conference) at any moment since both peers are clients and servers at the same time. What C-VPN does to handle this situation is to simulate a client-server scenario. The S2 that tries to open the communication channel becomes a "transient server", and the other S2 becomes a "transient client".

The transient client receives a message from the transient server and waits for the connection of the peer located in its LAN. Once it is established, the transient client executes the connection phase to a predefined SAux created before by the transient server. The reason of this behavior is the interactive nature of the client-client applications (e.g., voice applications) - maybe one peer is disconnected, or a user is unable to answer the call.

Multiple Requests. Since some protocols send multiple requests to the same port in the same session (e.g., HTTP), the Secondary Server (S2) does not destroy the "ticket" used in the connection phase. Hence, when a client application makes a new connection to S2, there will be always a ticket to create Auxiliary Servers for every request. In other words, there is one SAux per request between the client and the remote server.

Static Multiple Channels. Some protocols (e.g., RTP) connect to more than one port simultaneously during the same session. Since these ports are bound by the Secondary Servers (S2), S2 creates one SAux (thus, one secure channel) per port.

Dynamic Multiple Channels. Some protocols can create new communication channels during an on-going session (e.g., FTP). Therefore, C-VPN provides a mechanism for the analysis of the control packets of those protocols and further negotiates one or more secure channels to manage such communications.

Packet analyzers are the only external modules required in C-VPN. C-VPN uses one module per protocol. The module inspects every control packet sent and received through particular secure channels in order to discover and modify certain commands used by the protocols for the eventual creation of new channels. The interface for all these modules is standard, receiving the status of the channel at that time and the contents of the packet, and returning a modified packet and information about how to create new channels.

The information regarding the new channels to be created is used by SAux processes in order to open new secure connections, with or without negotiations with the SAux in the remote LAN. The effect of packet analyzers in C-VPN is that, whenever the applications receive modified control packets, new channels are set up. Those channels will be transparently managed by the corresponding SAux (Figure 4).

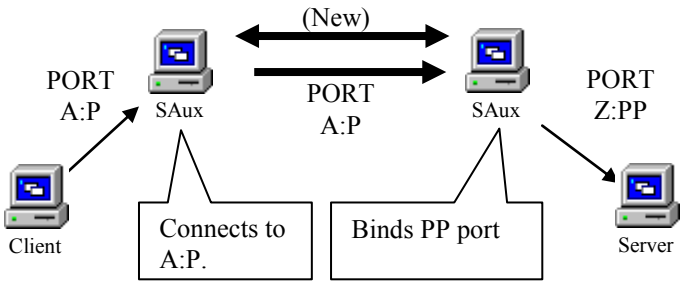

Figure 4. Dynamic Multiple Channel Protocols (FTP as example)

Packet analyzer modules can be stored in dynamic link libraries (DLLs) or, alternatively, in script languages [16]. Script languages are more flexible and easy to maintain than DLLs, but since script languages are interpreted languages, their speed is considerably low.

5. Other Considerations

5.1 C-VPN in Road Warriors

Road Warriors (in the following, R-W) are the users of a VPN that try to access to the private network resources using a portable device (e.g., laptop, PDA) that is not physically connected to any LAN of the VPN. An example of a R-W is the employee who tries to access the resources of a company from a hotel or home using a laptop.

Since a R-W is not inside any LAN of the VPN, there are no Secondary Master Server (S2M) and Secondary Servers (S2) to receive the communication requests from the clients. In addition, R-W cannot receive connections from any of the VPN entities, because the Main Server (S1) does not know how to find the device used by the R-W. For that reason, R-W cannot be used as a normal C-VPN client.

A possible solution for R-W being a special C-VPN client is to set up a LAN with only one device that have all the necessary VPN entities (including a Secondary Master Server and a Secondary Server) implemented on it. The R-W registers itself inside the VPN as a virtual LAN. However, this approach cannot be used in low-powered devices (e.g., PDAs), since the amount of resources of these devices is small.

5.2 Possible Limitations

The design of the C-VPN architecture has some features that are inherently associated with the way it manages the entities and the connections in the VPN.

Inside one LAN, we can find there is an intrinsic lack of security in the connections between the clients and the entities of the VPN. However, this is the same situation that we can find in the scenario of an IPsec-based gateway-to-gateway VPN.

On the other hand, R-W cannot play the roles of client and server (e.g., use a web browser and a http server) at the same time. Since the Secondary Server (S2) tries to bind the ports (e.g., port 80 in the example) used by the servers, both cannot coexist in the same device at the same time. Therefore, a R-W device must be configured to work as “client only” (that has no servers) or “server only” (that has no client applications). However, this is not a major issue because R-W always works as clients.

Another issue to consider is related to the number of instances of client applications. Due to the design of C-VPN, Secondary Servers (S2) must distinguish between connections from the same client that comes from different instances of the same application. S2 uses the "tickets" for this purpose. However, in certain protocols with specific characteristics (e.g., protocols with multiple requests) it is not possible to use those tickets to distinguish each instance of a client application from the same machine. (If two instances of a certain application in the same client machine open a connection, S2 will not be able to know which secure channel to be used to send packets.) Only a few protocols are affected by this issue; therefore, it is not a major inconvenience.

Finally, and regarding the efficiency of the scheme, it is very important to point out that the task of SAux processes is just to send and receive the information through the secure channels. This creates a minimal overhead in the analysis of control packets and in the management of a few protocols. Therefore, the efficiency of our solution is comparable to other software VPN solutions.

6. Conclusion

Traditional Virtual Private Networks protect the information flow of a company using centralized security policies. Using those policies, the management of security is efficient and homogeneous, and the company can prevent and respond adequately to information leakage and security breaches.

However, not all organizations need to protect the information flow at every time, but only when sensitive information is shared and transmitted between peers. This is the case in collaborative application scenarios. Motivated by such a requirement, we proposed a novel concept of Casual VPN and explained in detail in this paper.

We also developed a gateway-to-gateway Casual VPN solution, called C-VPN. This solution is able to manage TCP and UDP-based protocols, and can be used to interconnect both LANs and Road Warriors.

There are some aspects of Casual VPN that have not been totally investigated, such as interoperability with traditional VPN solutions and multicast/P2P support, which is part of our ongoing work.

References

[1] C. Scott, P. Wolfe, M. Edwin, "Virtual Private Networks", O'Reilly & Assoc., 1998.

[2] S. Kent, R. Atkinson, "Security Architecture for the Internet Protocol", Internet Engineering Task Force, RFC 2401, November 1998.

[3] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, J. Zahorjan, "Detour: A Case for Informed Internet Routing and Transport", IEEE Micro, v 19, no 1, January 1999.

[4] A. Freier, P. Karlton, P. Kocher, "The SSL Protocol Version 3.0". http://wp.netscape.com/eng/ssl3/

[5] T. Dierks, C. Allen, "TLS Protocol Version 1.0", Internet Engineering Task Force, RFC 2246, January 1999.

[6] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, L. Jones, "SOCKS v5", Internet Engineering Task Force, RFC 1928, March 1996.

[7] Multiprotocol Label Switching (MPLS) Charter, http://www.ietf.org/html.charters/mpls-charter.html

[8] Aventail, http://www.aventail.com

[9] Permeo Technologies Inc., http://www.permeo.com
[10] Nortel Networks, http://www.nortelnetworks.com
[11] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1", Internet Engineering Task Force, RFC 2616, June 1999.
[12] J. Postel, J. Reynolds, "File Transfer Protocol", Internet Engineering Task Force, RFC 959, October 1985.
[13] J. Postel, J. Reynolds, "Telnet Protocol Specification", Internet Engineering Task Force, RFC 854, May 1983.
[14] H. Schulzrinne, S. L. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", Internet Engineering Task Force, RFC 1889, January 1996.
[15] H. Schulzrinne, A. Rao, R. Lanphier, "Real Time Streaming Protocol (RTSP)", Internet Engineering Task Force, RFC 2326, April 1998.
[16] Department of Computer Science, Pontifical Catholic University of Rio de Janeiro, "LUA", http://www.lua.org