Interested in learning more about security?

SANS Institute InfoSec Reading Room

HTTP Tunnels Though Proxies

A proper security policy should take into consideration both the business need to accomplish work and the need for privacy and security. HTTP tunnels are necessary for SSL web browsing. However, due to a weakness in the CONNECT method in the HTTP protocol, arbitrary connection can be made through a HTTP proxy server. Furthermore, if these simple tunnels are used in conjunction with other protocols and applications, VPNs can be created between the local and remote systems. Once a VPN is established the perimeter of the ...

GSEC Practical Assignment

(v1.4b Option 1)

HTTP Tunnels Though Proxies

By Daniel Alman

July 30, 2003

--

Abstract

Hyper Text Transport Protocol or HTTP is the protocol used for web traffic. Its specification allows the use of proxies. Proxies are used in a large number of companies and network environments to protect internal machines from attack, accelerate web browsing, filter destinations, and to authenticate users. However, due to a weakness in the CONNECT method of HTTP, the proxies are capable of blindly passing more then just HTTP traffic and can be used to check email, connect to P2P (peer to peer) networks, and even allow bidirectional VPN (virtual private networks) traffic to bypass firewalls and other security devices. All that is needed to exploit HTTP tunnels is basic web browsing privileges through a proxy. Detecting this unauthorized traffic is difficult because it is often hidden in ways that make it almost indistinguishable from normal authorized traffic. However, with proper configuration of the proxy server the risks can be minimized.

HTTP Tunneling Explained

The fundamentals of HTTP tunneling

Hyper Text Transport Protocol or HTTP web proxies are used in many network environments. A proxy can be a network device such as Network Appliance's "NetCache," appliance (http://www.netapp.com/products/#netcache) or is often part of a firewall as with Secure Computing's "Sidewinder G2" Firewall. No mater where it is implemented, a proxy makes the actual request to the web server on behalf of the client system and hides the clients IP (Internet Protocol) address from the destination web server. This behavior is often seen as increasing the overall security of the network because the proxy breaks the outbound network connection to the web server and prevents direct inbound access to the client machine from outside the network.

Proxy's do prevent direct inbound access to a protected machine. However, they can be bypassed using HTTP tunnels. Using HTTP tunnels an individual can cerate arbitrary connection into or out of a protected network. All that is needed to tunnel through a proxy is an individual inside the network with basic web browsing access.

One technology that makes bypassing HTTP proxies so effective is encryption. While encrypting network traffic offers the client and server privacy and security for their communications, the lack of inspection can reduce the over all security for the network environment it is used in. When encapsulated network traffic is encrypted and passed though a HTTP tunnel, network connections can be created that allow arbitrary, bidirectional connections to remote destinations. When encryption technologies are used in this manor, Virtual Private Networks or VPNs can be created between internal and external machines. The VPN pushes

the network perimeter of the protected network beyond the firewall, router or other network security device. This opens the protected network up to the possibility of attack or misuse.

What is a HTTP proxy?

A HTTP proxy is a network device or application that sits between the client's web browser and the web server. The proxy's main job is to make requests to web servers on behalf of a client. This allows the proxy to offers several key benefits to the client systems. Such benefits are security, caching of web pages, content filtering, web usages logs, and authentication. Proxies are often part of, or used in conjunction with, firewalls to allow internal users to access the public network without exposing the internal machines to direct attack.

What is a SSL?

Secure Sockets Layer or SSL "is a protocol developed by Netscape for transmitting private documents via the Internet. SSL works by using a private key to encrypt data that's transferred over the SSL connection"(Webopedia).

While SSL was developed and is the standard for secure web traffic, a wide range of applications can make use of SSL. Using toolkits like OpenSSL, (http://www.openssl.org/) SSL authentication and encryption can be built into almost any type of application. Furthermore, by using a SSL wrapper program like Stunnel you can encrypt arbitrary TCP connections inside SSL. Stunnel is available on both Unix and Windows and Stunnel can allow you to secure non-SSL aware daemons and protocols (like POP, IMAP, LDAP, etc) by having Stunnel provide the encryption, requiring no changes to the daemon's code (Stunnel).

What is SSH?

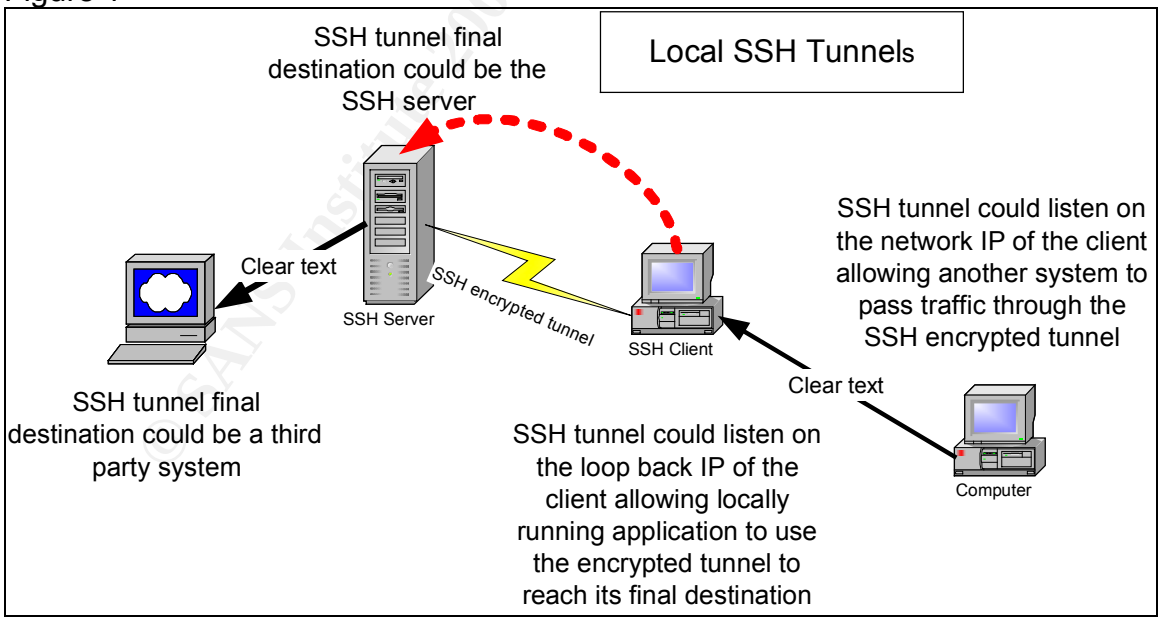
SSH (Secure Shell) was designed as a secure replacement for the UNIX "r" tools such as rsh (remote shell), rcp (remote copy), and rlogon (remote logon) (OpenBSD).

> All three "r" programs require a method for authenticating that you have permission to login or execute programs on the remote machine. They do not prompt for passwords. Rather, each system first assumes that you have the same login account name on both machines, and then verifies that your account on the remote machine is equivalent to your account on the local machine. Two methods of account verification are provided: system-to-system, or user-to-user (Farrell).

SSH was developed with the intent of keeping the “r” tool’s flexibility, ease of use, and functionality. But unlike the “r” tools SSH allows for stronger authentication methods. For example, SSH requires that the user prove his/her identity to the remote machine using public/ private key pairs, passwords, or hostname and account name information (OpenBSD). SSH main advantage is that it can “provide secure encrypted communications between two untrusted hosts over an insecure network” (OpenBSD). SSH has become a standard for administering remote systems and has largely replaced telnet and ftp where network security is needed.

One feature of SSH that goes beyond the “r” tools is its ability to create encrypted TCP tunnels between the local and remote system. SSH can create both local and remote tunnels between the systems. Local tunnels (Figure 1) listen on the local (client) machine and relay the traffic to the server. The server then delivers the traffic to its final destination. The tunnel can be set to listen on the loopback interface or 127.0.0.1(RFC 3330) of the client machine. This is convenient when configuring a locally running application to use the tunnel. However, SSH is not limited to listing on the loopback interface. The tunnel can be set to listen on a specified port on the network IP of the client system, allowing any machine on the local network to leverage the SSH tunnel. The final destination for the tunnel can be the server’s loopback interface, network IP or even a separate system that is reachable by the server system.

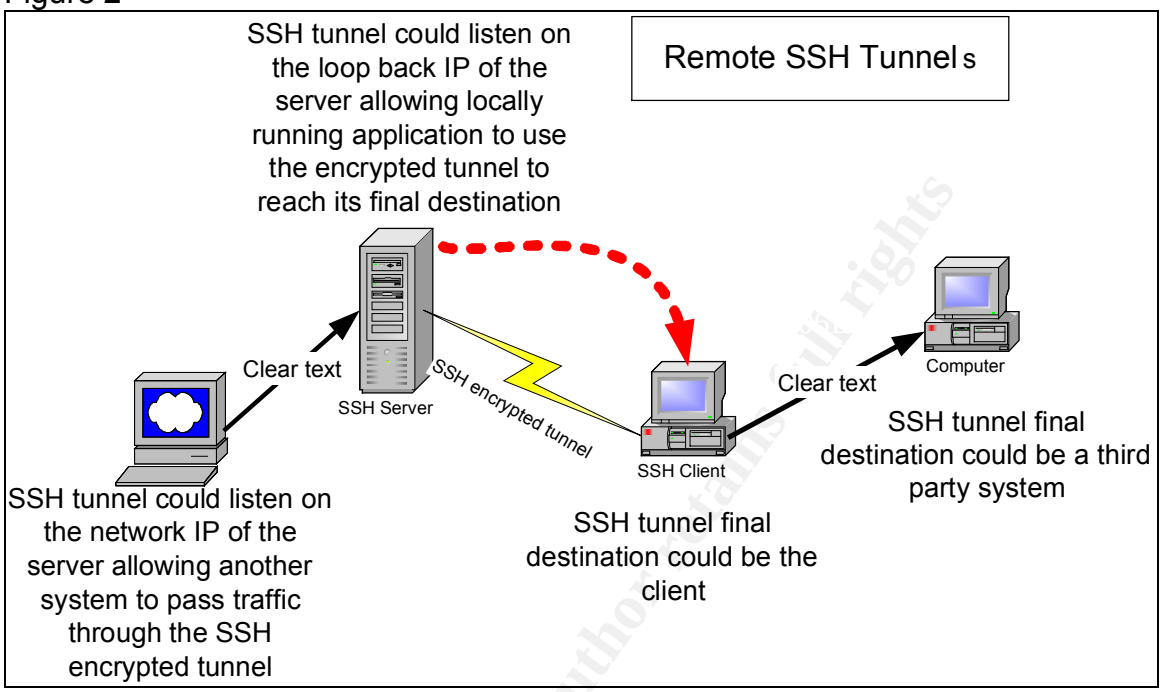
Figure 1



Remote tunnels (Figure 2) differ from local tunnels by listening on the remote (server) system. Like local tunnels, the remote system can listen on the loopback interface or the network IP and relay traffic to the client machine. The final

destination from the tunnel can be the client's loopback interface, network IP or even a separate system that is reachable by the client system.

Figure 2



What is a HTTPS tunnel?

One of the challenges to implementing HTTP security through a proxy is the proxy should not have access to any more information then is necessary to insure delivery of the data stream. As Ari Luotonen states in his expired Internet Draft "Tunneling SSL Through a WWW Proxy":

> When tunneling SSL, the proxy must not have access to the data being transferred in either direction, for sake of security. The proxy merely knows the source and destination addresses, and possibly, if the proxy supports user authentication, the name of the requesting user.

The solution was to take advantage of the HTTP CONNECT method as specified in RFC 2616 (http://www.cis.ohio-state.edu/cs/Services/rfc/rfc-text/rfc2616.txt). Unfortunately, this solution effectively allows arbitrary communication to take place through an HTTP proxy, because the CONNECT method is like an escape sequence telling the proxy not to interfere with the transaction (CERT VU#150227). Using the HTTP Connect method in this manor to pass SSL traffic is referred to as HTTPS.

For what the CONNECT is traditionally used for, tunneling SSL connections for secure web traffic, blindly passing the traffic ideal behavior. Once the proxy sees the CONNECT syntax the proxy assumes the traffic will be encrypted and blindly

forwards traffic between client and server. The blind forwarding of traffic is necessary to support the end-to-end security model of SSL. This blind passing of data through the proxy is referred to as tunneling, describing the fact that the content of the connection passes under or tunnels through the proxy.

The blind passing of traffic through the tunnels presents a security risk. Some of the security risks of the CONNECT method are detailed in CERT VU#150227. "The CONNECT method can be exploited to conduct port scanning, send unsolicited email, and even denial of service attacks if the proxy allows recursive connections." But as shown in this report, tunnels are a key weakness in HTTP proxies that can be exploited to establish VPNs or virtual private network connections, run chat applications, send and retrieve email, and bypass content filtering.

What are the risks?

Allowing any unmonitored or inappropriate traffic in or out of a network carries risk. Since HTTP tunneling can be leveraged by a wide verity of applications, the risks tunnels represent is very great. Unfortunately, because SSL is a necessary technology for conducting commerce and maintaining privacy on the Internet, some of the risks have to be accepted. But some of the risks are to great to ignore.

One risk is that someone inside the network would use Internet bandwidth for personal or inappropriate use and cause service interruption or network degradation to legitimate business traffic. For example, many of the file sharing P2P networks have clients that support HTTP proxies. These applications can utilize large quantities of bandwidth if allowed to run unchecked. Chat programs that allow file transfers can also eat up significant quantities of bandwidth.

A major risk is that employees may use tunnels to conduct behavior outside of the security or employee code of conduct polices. An employee using tunnels to bypass firewalls and IDS to access inappropriate content in violation of the security policy or code of conduct must be dealt with appropriately. Also, the use of company networks and equipment for file swapping of copyrighted martial or illegal images can open up the company to the risk of lawsuit and other legal action.

Furthermore, using a tunnel to check personal email or to conduct chat sessions can open up a potentially dangerous virus vector. Even if a well thought out security architecture is in place to prevent viruses from spreading through email and web browsing, tunnels make it possible for viruses to enter behind the cooperate network defenses and directly infect a users machine.

The greatest problem that tunnels pose is that they are very difficult to detect. This makes all the other risks listed far harder to prevent. HTTP tunnels are

difficult to detect mainly because they need to be allowed to pass HTTPS traffic, and normal HTTPS traffic is encrypted end-to-end, preventing inspection. Some steps can be taken to limit exposure to the risks posed by HTTP tunnels. But, a determined individual with administrative access to a client system inside the network environment will likely be able to exploit HTTP tunnels in some way.

Simple HTTP Tunnels

Description

Simple HTTP tunnels are an unencrypted connection through a HTTP proxy to an arbitrary destination. The tunnel takes advantage of the HTTP CONNECT method normally used for HTTPS (secure web traffic) to connect to the destination server. A typical HTTPS connection through a proxy should look like:

> CONNECT **remote-server:443** HTTP/1.0
> User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 4.0)
> Host: **remote-server**
> Content-Length: 0
> Proxy-Connection: Keep-Alive
> Pragma: no-cache

In the example above, a tunnel is established between the client and the remote-server with a destination port of 443 or the standard SSL port. If someone wanted to make a connection to **another-server** on **anyport** all that is needed is to send the following connection request instead.

> CONNECT **another-server:anyport** HTTP/1.0
> User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 4.0)
> Host: **another-server**
> Content-Length: 0
> Proxy-Connection: Keep-Alive
> Pragma: no-cache

As shown in the above example, HTTP tunnels are not restricted to web or SSL ports. Rather, HTTP tunnels are capable of passing any outbound traffic on any TCP port as long as the client warps the appropriate HTTP CONNECT header around the data stream.

Simple tunnels typically do not require control of the destination server. All that is needed is a remote server with a known listening service. Put another way, the server does not have to be modified in anyway to accept a TCP connection that passes through a simple HTTP tunnel.

While the server does not need modification, some work will need to be done on the client side to properly wrap the connection with the HTTP header. The client application may have proxy support built-in and is able to directly create the

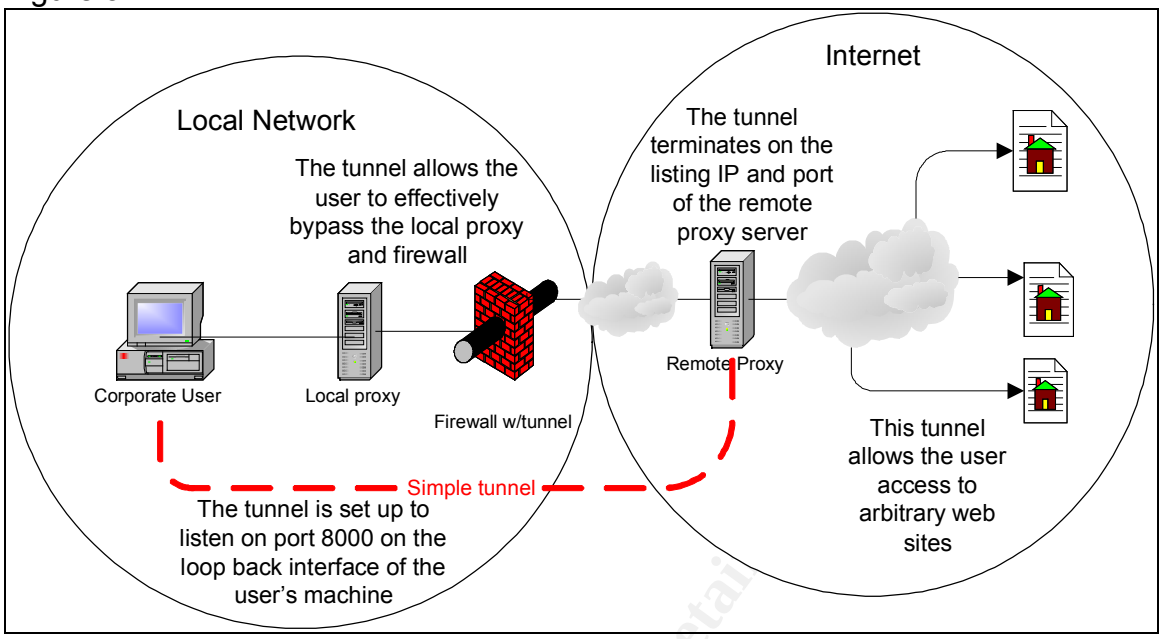
tunnel. However, a bridging application may be used to allow unmodified applications to pass through the proxy. One example of a bridging application written in PERL is connect-tunnel-0.03 (http://search.cpan.org/author/BOOK/connect-tunnel-0.03/) as detailed later in the simple tunnel example.

While simple tunnels are very useful and are the basic component of all the other more advanced tunnels, they do have their limitations. One limitation to simple tunnels is that each connection to a remote TCP port requires a separate tunnel. Another limitation is that they do not encrypt the connection and pass the data in the clear. If the data needs to be encrypted, it is up to the application to only pass encrypted data through the tunnel. For example, using a simple tunnel to check a pop mail account over the Internet would pass all mail messages in the clear over the Internet. To protect the contents of the mail messages an advanced tunnel employing some form of encryption would need to be employed.

Uses

Simple tunnels are limited to outbound TCP connections, but they still can be used for a wide variety of tasks. They can be used to check, receive, send external e-mail. IRC (internet relay chat) can also be used through a HTTP tunnel. The tunnels can be used to make FTP (File Transfer) connections to remote servers. Tunnels can even be used to have remote desktop access using programs such as VNC (Virtual Network Computing www.uk.research.att.com/vnc). If the tunnel is terminated at another HTTP proxy a user can even avoid most content filtering and destination restrictions implemented at the local proxy (Figure 3).

Figure 3



Almost any application that only uses outbound TCP connections can be passed through a HTTP proxy. Even many of the popular P2P network applications such as eDonkey (www.edonkey2000.com) and Kazaa (http://www.kazaa.com/us/index.htm) with the aid of Kazaahttp (http://www.iprisma.com/kazaahttp/index.htm) can be used through a HTTP proxy.

ICQ (www.icq.com) a popular chat program also has HTTP proxy support implemented. When applications that allow file transfer like ICQ and the P2P network clients are used from behind firewall they pose a unique threat to the local network. All an attacker would need to do is to offer a virus infected file or backdoor program to everybody in a public way and lure an unsuspecting tunnel user into downloading the file. The result is that the machine that is thought protected by network security and network antivirus devices has just been successfully infiltrated and is ready to be exploited.

Simple Tunnel Example

This example will show how a simple tunnel can be created to pass telnet traffic thought the HTTP proxy.

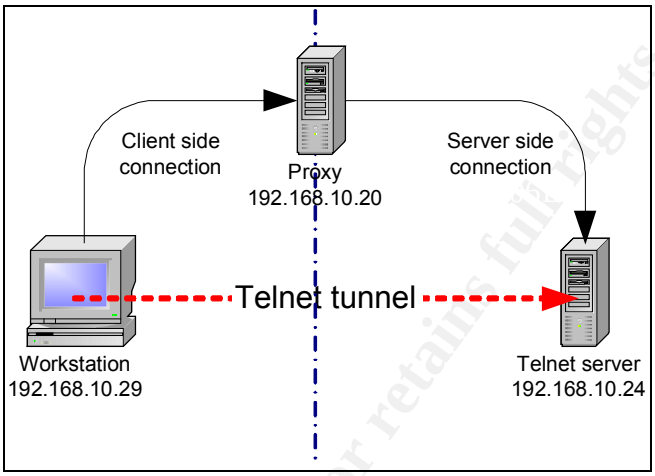
Network setup

In this example there are three systems.

1. The client system in a Windows NT system with PERL and connect-tunnel-0.03 (http://search.cpan.org/author/BOOK/connect-tunnel-0.03/) installed.

2 The Proxy server is a windows XP system running Proxy+ 3.0 (http://www.proxyplus.net/). The proxy is listing for connection on port 4480.
3 The Telnet server is a windows XP system running WAC Server 1.4 (http://www.foxitsoft.com/wac/server_intro.asp).

Network Diagram



Description

To initialize the tunnel the PERL program connect-tunnel was executed with the following command.

```
perl connect-tunnel --proxy 192.168.10.20:4480 --tunnel 2323:192.168.10.24:23
```

This prepares the HTTP tunnel on the client to accept connection on TCP port 2323 on the loopback interface on the client machine and relay though the proxy (192.168.10.20) on TCP port 4480. The tunnel ends at the telnet server (192.168.10.24) on TCP port 23 (the standard telnet port).

Once the HTTP tunnel is initialized, all that is left to do is open up a telnet connection through the tunnel. To do this, launch telnet on the client system and connect to localhost on TCP port 2323 (the listing port of the tunnel). Connect-tunnel then transparently wraps the telnet traffic in the HTTP header and passes the traffic to the proxy server.

The traffic between the client and proxy looks like this.

```
[client]
        CONNECT 192.168.10.24:23 HTTP/1.0
        Host: 192.168.10.24:23
        User-Agent: libwww-perl/5.51
[proxy]
        HTTP/1.0 200 Connection established
        Proxy-agent: Proxy+ 3.00
```

```
        Welcome to WAC Server 1.4 Build 0725. (C) Foxit Software, 2002-2003

        Evaluation Version. Maximum 2 users.
        Host: TELNETSERVER, OS: Windows XP

        Please use your Windows username and password to logon.
        Username:username
[client]
        username
[proxy]
        Password: ********
[client]
        password
[proxy]
        Domain:
[client]

[proxy]
        c:>
```

As shown in the above example, the client makes the CONNECT request to the proxy server and asks it to make a connection to the Telnet server on TCP port 23. The client also reports its user agent as libwww-perl/5.51. The proxy then responds that the connection is allowed. Then makes the connection to the telnet server and relays the telnet welcome message to the client. The client then passes username, password, and domain information to the telnet server. The telnet server grants access and gives a command prompt.

While the network traffic on the client side of the connection has the HTTP header, the server side of the connection looks like an ordinary telnet connection. The following is the capture of the traffic on the server side of the proxy.

```
        Welcome to WAC Server 1.4 Build 0725. (C) Foxit Software, 2002-2003

        Evaluation Version. Maximum 2 users.
        Host: TELNETSERVER, OS: Windows XP

        Please use your Windows username and password to logon.
        Username:username
[client]
        username
[proxy]
        Password: ********
[client]
        password
[proxy]
        Domain:
[client]

[proxy]
        c:>
```

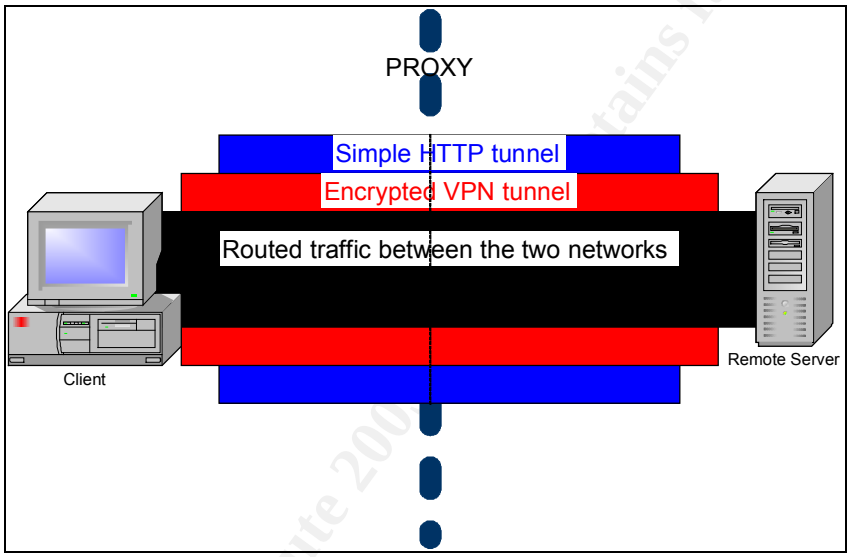
To any network sniffing device on the server side of the network, it looks like the proxy, not the client, is initiating the telnet session.

HTTP VPN Tunnels

Description

VPNs or Virtual Private Networks are connection between systems over public networks, such as the Internet, that uses encryption methods to ensure privacy. VPNs can refer to single port to port encrypted communications or to protocols such as IPSEC that are designed to encapsulate all traffic between the two systems. A HTTP VPN tunnel is where any VPN technique is uses in conjunction with a simple HTTP tunnel as describe earlier (Figure 4).

Figure 4



VPN tunnels have several advantages over simple tunnels. The biggest advantage is that the communications between the systems are encrypted. This prevents anyone whom intercepts the commutations along the network path from being able to decipher the contents. This also protects the traffic from any intrusion detection systems. Using AppGate's Mindterm (http://www.appgate.com/mindterm/), for example, a user can easily leverage the encryption features of SSH to remotely manage a system and transfer files through a HTTP proxy.

Another advantage of HTTP VPN tunnels, and the largest risk, is that with VPN tunnels it is possible pass any protocol in either direction, creating a full VPN connection with a remote site. One example is to wrap a PPTP or point–to-point tunneling protocol connection inside a simple HTTP tunnel. PPTP is relatively easy to configure and install on modern Microsoft Windows (http://www.microsoft.com) operating systems such as windows 98/NT/2000/xp. And, PPTP support is shipped with all theses operating systems. If the user has

access to a Linux system, the users could install Amrita VPN (http://amvpn.sourceforge.net/). “Amrita VPN is an easy-to-use open source VPN solution that runs on the GNU/Linux platform. The implementation is fully in userspace and requires no kernel patches or enhancements. It uses openssl library for strong encryption and authentication through SSLv3”(Amrita Institutions - Amrita VPN).

“The amvpn client can connect to amvpn server through a proxy. This option may be required if the amvpn server is sitting behind a firewall. AmritaVPN currently supports proxy authentication using Basic HTTP authentication” (Amrita Institutions - Essential Features)

With any HTTP VPN tunnel the network’s perimeter is pushed beyond the firewalls, IDS, routers and any other network security measure in place. The security of the local network depends on the security the client’s system and the remote network it is connected to.

Uses

There are few limitations on what VPN tunnels can be used for. If a full VPN is established to a remote network all network traffic can freely flow between the connected networks. A user may set up a VPN to remotely access a home network, or access the company’s network from home. A malicious user could set up a VPN to avoid detection while accessing forbidden or illegal remote network resources like music, movies, and images.

Limiting the risk of HTTP tunnels

HTTP tunnels present a difficult challenge to network security. HTTP tunnels are a necessary component of the ordinary and useful SSL connections. The easiest way to reduce the risk of HTTP tunnels being used maliciously is to prevent them completely. Blocking all HTTPS tulles would prevent legitimate HTTPS traffic as well. This, however, is not a very good solution. It does not attempt to balance business need with the security need.

A better solution is to maintain a list of blocked destination at the proxy. Many proxies are designed with this functionality, and it is likely that destination restrictions are already maintained or other reasons. Sites that could be added to the list of non-allowed destinations include to lower risks of HTTP tunnels:

Chat
- Logon servers for ICQ
- Logon servers for MSN messenger
- Logon servers for AOL instant messenger

Home networks
- Locally available DSL ISP IP ranges

Locally available Cable ISP IP ranges
Locally available dialup ISP IP ranges

Remote proxies
Open proxies list such as the ones maintained by at
http://openproxies.com/,
http://www.monkeys.com/upl/index.html, and
http://darkwing.uoregon.edu/%7Ejoe/open-proxies-used-to-send-spam.html

Along with destination restrictions at the proxy, if possible, port restrictions should be considered for the proxy server. The simplest way is to only allow connection to remote sights on ports 80(HTTP) and 443(HTTPS). However, many legitimate sites uses ports other then the standard ports. So if this solution is implemented some administration is necessary to add site destinations back to the allowed list as needed. A method that often requires less administration is to block a list of well known destination ports (IANA – port numbers).

Block

20:	FTP (file transfer protocol)
21:	FTP (file transfer protocol)
22:	SSH (secure shell)
23:	telnet
25:	SMTP (mail)
110:	POP3
119:	NNTP (network news protocol)
137:	netbios-ns (windows networking)
138:	netbios-dgm (windows networking)
139:	netbios-ssn (windows networking)
143:	imap (email)
220:	imap v3 (email)
445:	Microsoft-ds (windows networking)
1214:	default Kazaa port
1723:	PPTP (point-to-point tunneling protocol)
1080:	socks proxy
5190:	ICQ
5800:	VNC (remote desktop)
5900:	VNC (remote desktop)
6665-6669:	IRC
8000:	common proxy port
8080:	common proxy port

This list of port number represents the services that pose the largest threats to network security. But, theses services are not restricted to running on its assigned or common port. The administrator of the remote server can run the servers on any port desired. The remote admin could run a service like SSH on

port 443, the normal HTTPS port. This would allow users behind a proxy to connect as though they were going to a normal SSL enabled site, avoiding all the port restrictions described.

Another way to limit the risk of HTTP tunnels is to monitor proxy and firewall logs. One signature that a HTTP tunnel being used for nonstandard uses is if the tunnels have abnormally long connection times. After monitoring connection lengths at a fortune 500 company, it was found that most SSL web sights have connection lengths of less than 1 hour. Therefore, watching the logs for connection lengths greater then 1 or 2 hours and watching for repeat offenders can flag users who are using HTTP tunnels inappropriately.

Another way monitoring logs can help to limit the use of tunnels is to monitor the user agent of the client applications that access the proxy. While it is often easy to configure the user agent of applications like connect-tunnel, since it is a PERL script, changing the user agent in some application is quite difficult. Once again this is another way to flag users that might be abusing network privileges.

Conclusion

A proper security policy should take into consideration both the business need to accomplish work and the need for privacy and security. HTTP tunnels are necessary for SSL web browsing. However, due to a weakness in the CONNECT method in the HTTP protocol, arbitrary connection can be made through a HTTP proxy server. Furthermore, if these simple tunnels are used in conjunction with other protocols and applications, VPNs can be created between the local and remote systems. Once a VPN is established the perimeter of the local network is push to the remote system. The risks to the local machine and network depend on what applications being used and on the security of the remote systems. This poses significant risk to the owners of the local network. Steps can be taken to limit the risk of HTTP tunnels being exploited and still allow appropriate SSL web traffic. The local proxy administrator can limit destinations by web site and port number. They can also monitor connection times and flag users that make repeat long duration HTTP connections. So, while HTTP tunnels poses risks, they can be limited with proper administration and the business need of secure web traffic can still be allowed.

Works Cited

Amrita Innovative Technology Foundation Labs, “Amrita VPN” URL: http://amvpn.sourceforge.net/ (July 2003)

Amrita Innovative Technology Foundation Labs, “AmritaVPN - A Virtual Private Networking tool for GNU/Linux.: Amrita VPN Configuration - Essential Features” URL:http://amvpn.sourceforge.net/amvpn-4.html (July 2003)

Bruhat, Philippe, “Search.cpan.org:Philippe ‘Book’ Bruhat / connect-tunnel-0.03” March 20, 2003 URL: http://search.cpan.org/author/BOOK/connect-tunnel-0.03/ (July 2003)

CERT Coordination Center, “Vulnerability Note VU#150227: Multiple vendor’s HTTP proxy default configuration allows arbitrary TCP connections via HTTP CONNECT method” June 24, 2003 URL:http://www.kb.cert.org/vuls/id/150227 (July 2003)

Farrell, Phillip, “rlogin, rcp, and rsh network programs” URL: http://pangea.stanford.edu/computerinfo/internet/rcommands.shtml (June 2003)

The Internet Assigned Numbers Authority, “Internet Protocol V4 Address Space” URL:http://www.iana.org/assignments/ipv4-address-space (June 2003)

The Internet Assigned Numbers Authority, “Port numbers” URL: http://www.iana.org/assignments/port-numbers (June 2003)

Luotonen, Ari, “Tunneling SSL Through a WWW Proxy” December 14, 1995 URL:http://muffin.doit.org/docs/rfc/tunneling_ssl.html (July 2003)

Network Working Group, “RFC 1918 Address Allocation for Private Internets” February 1996 URL:http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1918.html (June 2003)

Network Working Group, “RFC 2616 Hypertext Transfer Protocol – HTTP/1.1” June 1999 URL: http://www.cis.ohio-state.edu/cs/Services/rfc/rfc-text/rfc2616.txt (June 2003)

Network Working Group, “RFC 3330 Special-Use IPv4 Addresses” September 2002 URL:http://www.faqs.org/rfcs/rfc3330.html (July 2003)

OpenBSD, “Manual Pages: ssh(1)” September 25, 1999 URL: http://www.openbsd.org/cgi-bin/man.cgi?query=ssh&sektion=1 (July 2003)

Stunnel.org, “Stunnel.org” URL:http://www.stunnel.org/ (June 2003)

Webopedia, “SSL-Webopedia”
URL:http://www.webopedia.com/TERM/S/SSL.html (June 2003)

Last Updated: December 11th, 2017

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS SEC460: Enterprise Threat Beta	San Diego, CAUS	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS Security East 2018	New Orleans, LAUS	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS Amsterdam January 2018	Amsterdam, NL	Jan 15, 2018 - Jan 20, 2018	Live Event
Northern VA Winter - Reston 2018	Reston, VAUS	Jan 15, 2018 - Jan 20, 2018	Live Event
SEC599: Defeat Advanced Adversaries	San Francisco, CAUS	Jan 15, 2018 - Jan 20, 2018	Live Event
SANS Dubai 2018	Dubai, AE	Jan 27, 2018 - Feb 01, 2018	Live Event
SANS Las Vegas 2018	Las Vegas, NVUS	Jan 28, 2018 - Feb 02, 2018	Live Event
SANS Miami 2018	Miami, FLUS	Jan 29, 2018 - Feb 03, 2018	Live Event
Cyber Threat Intelligence Summit & Training 2018	Bethesda, MDUS	Jan 29, 2018 - Feb 05, 2018	Live Event
SANS Scottsdale 2018	Scottsdale, AZUS	Feb 05, 2018 - Feb 10, 2018	Live Event
SANS London February 2018	London, GB	Feb 05, 2018 - Feb 10, 2018	Live Event
SANS Southern California- Anaheim 2018	Anaheim, CAUS	Feb 12, 2018 - Feb 17, 2018	Live Event
SANS Secure India 2018	Bangalore, IN	Feb 12, 2018 - Feb 17, 2018	Live Event
Cloud Security Summit & Training 2018	San Diego, CAUS	Feb 19, 2018 - Feb 26, 2018	Live Event
SANS Dallas 2018	Dallas, TXUS	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS Brussels February 2018	Brussels, BE	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS Secure Japan 2018	Tokyo, JP	Feb 19, 2018 - Mar 03, 2018	Live Event
SANS New York City Winter 2018	New York, NYUS	Feb 26, 2018 - Mar 03, 2018	Live Event
CyberThreat Summit 2018	London, GB	Feb 27, 2018 - Feb 28, 2018	Live Event
SANS London March 2018	London, GB	Mar 05, 2018 - Mar 10, 2018	Live Event
SANS Cyber Defense Initiative 2017	OnlineDCUS	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced