

Data Driven Attacks Using HTTP Tunneling

Ido Dubrawsky 2004-08-02

As more traffic across the Internet is coming under scrutiny and network administrators are making efforts to limit the traffic in and out of their networks, the one port that no one is willing to block en-masse is port 80. Users (and administrators) browse the web constantly, whether it is for work purposes or not. The lifeblood of a company's existence on the Internet requires a web presence in one fashion or another and this requires a web server, whether it is hosted by a service provider or located on a company's network. With every new worm, bug, or vulnerability found in IIS and Apache servers, network and secop administrators are trying to lock down these systems further at the router or firewall. To identify attacks many are turning to IDS and IPS.

In this article we will look at a means to bypass the access control restrictions of a company's router or firewall. This information is intended to provide help for those who are legitimately testing the security of a network (whether they are in-house expertise or outside consultants). This article, by no means, condones the use of this information for the purpose of unauthorized access to a network or a system. Finally, this article will provide some pointers on how to defend against this attack.

What is HTTP Tunneling?

Tunneling is nothing new. IPSec is perhaps the most widely known version of tunneling followed by SSH tunneling as a close second. However, an attacker is less likely to choose IPSec or SSH as a possible port of entry unless they have a fairly high chance of success. Not every company provides remote access through IPSec or SSH and if they do, the server may have already been hardened. Additionally, the device is more than likely either a custom appliance such as Cisco's VPN 3000 series concentrators or perhaps a UNIX/Linux system.

Due to the ubiquitous nature of web servers they represent an excellent high data value (HDV) target for attackers. In addition, given that they are probably behind a firewall they represent a single but important step into the target network. This is where HTTP tunneling comes in. As security awareness has increased, companies have added additional security systems between the web server and the Internet as well as on the web server itself. Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) are used to identify and alert administrators when an attack is detected against a system or a network. However, even these have their limitations and HTTP tunneling can be used to circumvent them.

HTTP Tunneling works by utilizing a client to encapsulate traffic within HTTP headers. The traffic is then directed to a server at the other end of the communication channel that takes the packets, strips the HTTP encapsulation headers and redirects the packet to its final destination. Both UDP and TCP traffic can be accommodated and encapsulated. This is due to the nature of the tunnel which, like an IPsec tunnel, sees the packets as the data payload only.

Currently there are only two HTTP Tunneling software packages available. One is Open Source in nature, the other is a commercial product. The first HTTP Tunneling package is the GNU HTTPtunnel available at <http://www.nocrew.org/software/httpunnel.html>. The other HTTP Tunneling software is [HTTP Tunnel](#) from HTTP-Tunnel Corporation. Both are designed to provide communications to various ports by directing the communication through TCP port 80. The rest of this article will focus on the free GNU HTTPtunnel software package and its use in pen-testing firewalls and other traffic filters.

HTTP Tunneling Example

HTTP tunneling can be used to access ports that are normally inaccessible from a network. Consider Figure 1 below. The attacker's host is shown on the left with the target systems on the right. The router at the edge has the following policies:

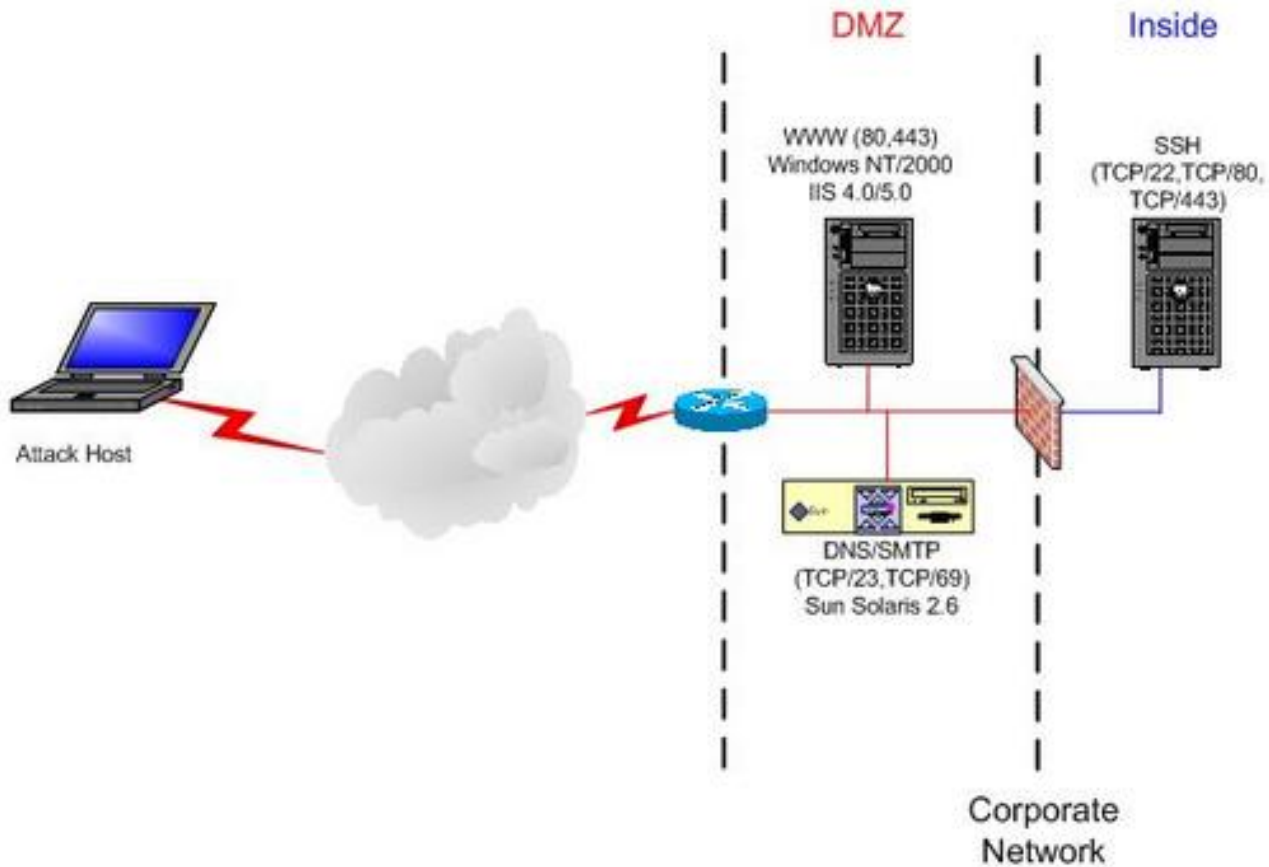


Figure 1: HTTP Tunnel Example Setup

inbound:

```
permit tcp any host WWW port 80
permit tcp any host WWW port 443
permit tcp any host DNS/SMTP port 25
permit udp any host DNS/SMTP port 53
```

outbound:

```
permit ip any any
```

The firewall has the following security policy enforced:

inbound:

```
permit ip host DNS/SMTP host SSH eq 22
permit ip host DNS/SMTP host SSH eq 80
permit ip host DNS/SMTP host SSH eq 443
```

outbound:

```
permit ip any any
```

While this is extremely simplistic in its scope it is sufficient for demonstration purposes. The attacker exploits the WWW server running IIS. It doesn't matter which particular exploit he uses; rather, that he is able to exploit the server and gain a command line access to the system. Once he has established that access he uploads a precompiled version of the HTTP tunnel server, *hts*. The syntax of the HTTP tunnel server is as follows:

```
hts.exe -F (SRC PORT) (TARGET):(DST PORT)
```

(SRC PORT) is the port that will be forwarded, the (TARGET) value is the IP address of the destination host, and (DST PORT) is the target port that will accept the forwarded traffic. When the client sends traffic to the (SRC PORT) the server will automatically forward it to the (DST PORT) on the (TARGET). (TARGET) can be the IP address of the system that the *hts* server is running on. With the *hts* server set up the attacker then starts a client on his system and directs its traffic to the (SRC PORT) of the system running the *hts* server. The syntax of the HTTP tunnel client is the same as the server:

```
htc -F (SRC PORT) (TARGET):(DST PORT)
```

The client forwards traffic from the port it is listening on - the <SRC PORT> -- and forwards it to the <DST PORT> of the <TARGET>.

Consider one further example of the exploitation of the system in Figure 1. In Figure 2, below, the attacker has setup the *hts* software on the WWW host (step 1) and started *htc* on his system (step 2). The *hts* process listens on port 80 of the host WWW and can redirect traffic. In this example the attacker has the *hts* process redirecting traffic to the telnet port (TCP/23) of the DNS/SMTP server running Solaris 2.6.

When the attacker connects to port 1025 on his system (step 3) the *htc* process encapsulates the traffic in HTTP headers and forwards it to the WWW server port 80 (step 4). This server then takes the traffic, strips the HTTP headers and forwards the traffic on to the DNS/SMTP server port

23 (step 5). In this way, while the telnet port on the DNS/SMTP is not directly accessible beyond the corporate router the attacker has now managed to setup a covert channel that does provide access to this port and he can now try to brute force a login into the system.

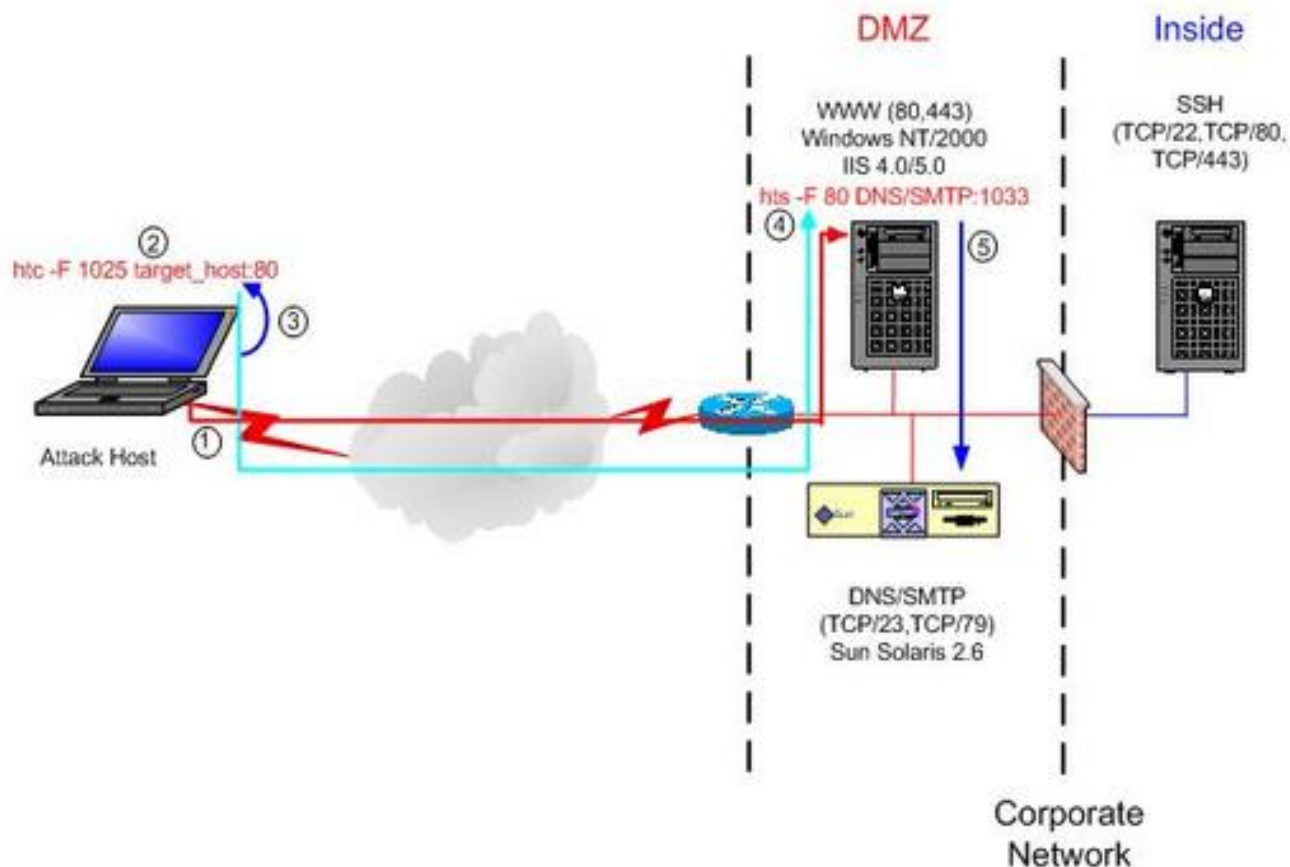


Figure 2: HTTP Tunnel Initial Exploit

Another possibility is to point the remote end of the *hts* tunnel to the finger port (TCP/79) of the SMTP/DNS host and list out all the users on the system. This provides a first glance as to possible account names. Once access to the system is gained (whether through a brute force login or through an exploitation attack such as the *sadmind* buffer overflow, the attacker now has access to a system behind the router that they would not have access to normally. By gaining command line access to the SMTP/DNS host behind the router the attacker can then setup additional tools to further exploit into the network. For example, the attacker can monitor connections on the SMTP/DNS host and identify potential services running on systems behind the firewall. This would

be a stealthier method of reconnaissance than outright scanning of IPs within the DMZ. If the attacker identifies a host running an SSH server behind the firewall they can then use the SMTP/DNS host platform to attempt to connect to the new target and use any guessed or cracked account information to gain entry as shown in Figure 3, below.

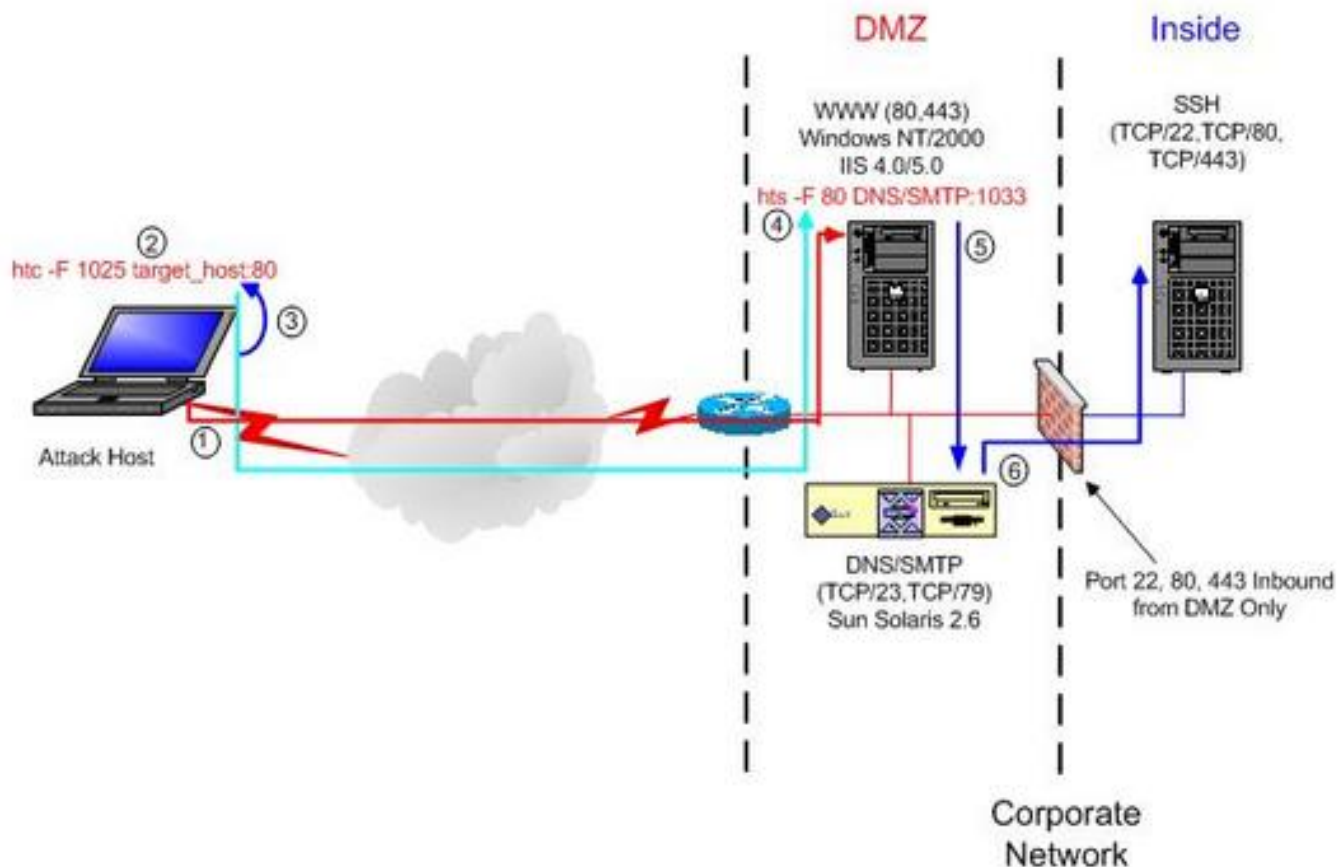


Figure 3: Further Exploiting the Network

HTTP Tunneling in Penetration Testing

One of the best uses of the HTTP tunneling method is to test the limits and capabilities of a network's perimeter security. While almost all networks limit the inbound traffic at the edge, many of these networks do not restrict outbound traffic. By using HTTP tunneling during a penetration test, a security consultant can identify additional security risks that may not be readily visible by conducting simple network and vulnerability scanning. Additionally, HTTP

tunneling can be useful in determining the extent to which a network IDS can identify malicious traffic within a communication channel. HTTP tunneling can provide significant capabilities to a penetration test that may not readily be visible.

More articles

View [more articles](#) by Ido Dubrawsky on SecurityFocus.

Comments or reprint requests can be sent to the [editor](#).

[Privacy Statement](#)

Copyright 2006, SecurityFocus